SIAM J. SCI. STAT. COMPUT.
Vol. 7, No. 1, January 1986

SIMPLE GENERAL APPROXIMATIONS FOR A RANDOM VARIABLE AND ITS INVERSE DISTRIBUTION FUNCTION BASED ON LINEAR TRANSFORMATIONS OF A NONSKEWED VARIATE*

HAIM SHORE†

Abstract. Linear transformations of a nonskewed random variable are employed to derive simple general approximations for a random variable having known cumulants. Introducing the unit normal variate, these become linear normal approximations.

Some nonskewed variates with explicit inverse cumulative density function are then used to derive general approximations for the inverse DF of the approximated variable.

The approximations are applied to the binomial, Poisson, Fisher's z and F, gamma (chi-square in particular) and the t distributions and their accuracy examined.

Simple general approximations for the loss function of a random variable either continuous or discrete are developed. A simple approximation for the loss function of the Poisson distribution is then derived and demonstrated by an example from inventory analysis.

Two further examples from interval estimation and from hypothesis testing highlight the usefulness of the new approximations.

Key words. approximations, binomial, chi-square, distributions, F distribution, gamma distribution, hypothesis testing, inverse distribution function, loss function, normal approximation, Poisson, t distribution

1. Introduction. Many of the statistical problems that a practitioner encounters are difficult to solve due to their inherent mathematical nontractability. If, for instance, he wishes to find a sample size needed in estimating the ratio of the variances of two normal populations for a nonstandard confidence or precision level he will find it mathematically impossible and otherwise difficult to accomplish since available tables are confined to standard values only. Sensitivity analysis of optimal solutions in inventory analysis, to take another example, is rarely possible even though derivation of the optimal solution with today's computing facilities is easy to attain.

A need thus arises for approximations which while accurate enough will preserve that degree of simplicity required to derive closed form expressions for various decision variables incorporated in statistical models.

With regard to the majority of existing approximations this simplicity requirement is rarely met. For example, while most approximations based on transformations of the normal deviate (general ones like the Cornish–Fisher expansion (1937) or approximations aimed for individual distributions like Bailey's (1980) are highly accurate their algebraic structure is too complex for the aforementioned objective.

In this paper we develop a series of simple general approximations based on linear transformations of a standardized nonskewed random variable.

In § 2 we show that by a proper choice of the linear transformation any random variable (either nonskewed or skewed) with known cumulants may be approximated by a random variable with a symmetrical distribution, where accuracy is determined by the ensuing approximate equality of the first three or four cumulants. The structure these approximations assume when the standard normal variate serves as the approximating variable is shown in § 3.

In § 4 several symmetrical random variables having explicit inverse DF are employed to derive simple general approximations for the inverse of a skewed variate. Section 5 repeats the latter for a nonskewed variable.

* Received by the editors January 31, 1984, and in final form December 10, 1984.
† 16 Yeguada Hanassi Street, Tel Aviv 69200, Israel.

The above linear transformations are applied in § 6 to the binomial, the Poisson, Fisher's z and F, gamma (chi-square in particular) and the t distributions, and their accuracy demonstrated.

Simple general approximations for the loss function of a random variable expressed in terms of the distribution function are derived in § 7, and demonstrated for the Poisson. A problem from inventory analysis exemplifies their usefulness.

Eventually, by employing some of the above approximations we arrive in § 8 at approximate explicit expressions for the decision variables of two problems in statistical inference commonly encountered by applicants.

Comparison of the new approximations with existing ones in terms of both algebraic tractability and accuracy is deferred to a projected paper currently under preparation.

2. Presentation of general approximations. Let X be a standardized random variable, the distribution of which depends on a certain parameter, n, so that as n tends to infinity X approaches normality.

Assume that l_r, the rth cumulant of X, is of order $n^{1-r/2}$, and denote the cumulative density function of X by $F(x)$.

Let Z be a standardized random variable with a symmetrical distribution and known partial moments, M_i, where

$$M_i = \int_{1/2}^{1} z^i \, dG(z)$$

and $G(\cdot)$ is the cumulative density function. Let (x, z) be a pair of values of the respective variates related by $F(x) = G(z) = P$.

In this paper we examine a linear transformation of z that approximates x, where the transformation, denoted $\hat{x}$, has the general algebraic structure

$$\hat{x} = \begin{cases} A_1 z + B_1, & P < \frac{1}{2}, \\ A_2 z + B_2, & P > \frac{1}{2}. \end{cases} \tag{1}$$

$\hat{X}$ has mean, variance, third and fourth cumulants equal to, respectively,

$$E(\hat{X}) = (A_2 - A_1) M_1 + \tfrac{1}{2}(B_2 + B_1), \tag{2}$$

$$\begin{aligned} V(\hat{X}) = {} & (A_2^2 + A_1^2)(M_2 - M_1^2) + 2A_2 A_1 M_1^2 \\ & + (B_2 - B_1)[\tfrac{1}{4}(B_2 - B_1) + M_1(A_2 + A_1)], \end{aligned} \tag{3}$$

$$\begin{aligned} \hat{l}_3 = {} & (A_2^3 - A_1^3)(M_3 - 3M_2 M_1 + 2M_1^3) \\ & + 3(A_2^2 A_1 - A_2 A_1^2)(M_2 M_1 - 2M_1^3) + 3(A_2^2 - A_1^2)(B_2 - B_1)(\tfrac{1}{2} M_2 - M_1^2) \\ & + \tfrac{3}{2}(A_2 - A_1)(B_2 - B_1)^2 M_1, \end{aligned} \tag{4}$$

$$\begin{aligned} \hat{l}_4 = {} & (A_2^4 + A_1^4)(M_4 - 4M_3 M_1 + 6M_2 M_1^2 - 3M_1^4) \\ & + 4(A_2^3 A_1 + A_2 A_1^3)(M_3 M_1 - 3M_2 M_1^2 + 3M_1^4) + 6A_2^2 A_1^2 (2M_2 M_1^2 - 3M_1^4) \\ & + 2(B_2 - B_1)[(A_2^3 + A_1^3)(M_3 - 3M_2 M_1 + 3M_1^3) \\ & \qquad + (A_2^2 A_1 + A_2 A_1^2) 3M_1 (M_2 - M_1^2)] \\ & + \tfrac{3}{2}(B_2 - B_1)^2 [(A_2^2 + A_1^2)(M_2 - M_1^2) + 2A_2 A_1 M_1^2] \\ & + \tfrac{1}{2}(B_2 - B_1)^3 (A_2 + A_1) M_1 + \tfrac{1}{16}(B_2 - B_1)^4 - 3. \end{aligned} \tag{5}$$

Let us now consider a few special cases.

First, if X has a symmetrical distribution we should have $\hat{l}_3 = 0$. This we obtain by putting $A_1 = A_2 = A$, which necessarily leads to $B_1 = -B_2 = B$ since then $E(\hat{X}) = 0$.

The solution for this case is (see Appendix A for details)

$$A = [1 + M_1(2/M_2)^{1/2}B]/(2M_2)^{1/2},$$
$$B = [l_4 + 3 - 0.5M_4/M_2^2]/[(2/M_2)^{3/2}(M_1 M_4/M_2 - M_3)]. \tag{6}$$

This approximation has its variance and fourth cumulant identical with those of X to $O(B^2)$. If $M_4 = 3/2$ then B is $O(n^{-1})$ and we achieve identity to $O(n^{-2})$.

Second, let $A_1 = 1 - C$, $A_2 = 1 + C$ and $B_1 = -hC$, $B_2 = -hC$. Then

$$E(\hat{X}) = C(2M_1 - h), \tag{2a}$$

$$V(\hat{X}) = 2M_2 + C^2(2M_2 - 4M_1^2), \tag{3a}$$

$$\hat{l}_3 = 6C(M_3 - 2M_2 M_1) + 2C^3(M_3 - 6M_2 M_1 + 8M_1^3), \tag{4a}$$

$$\hat{l}_4 = 2(1 + 6C^2 + C^4)(M_4 - 4M_3 M_1 + 6M_2 M_1^2 - 3M_1^4)$$
$$+ 8(1 - C^4)(M_3 M_1 - 3M_2 M_1^2 + 3M_1^4)$$
$$+ 6(1 - C^2)^2(2M_2 M_1^2 - 3M_1^4) - 3. \tag{5a}$$

To have $E(\hat{X}) = 0$ we first put $h = 2M_1$. To find C we equate $\hat{l}_3 = l_3$ to obtain a cubic equation,

$$C^3 + 3C(U/V) - (l_3/2)/V = 0, \tag{7}$$

where

$$U = M_3 - 2M_2 M_1,$$
$$V = M_3 - 6M_2 M_1 + 8M_1^3 = U - 4M_1(M_2 - 2M_1^2).$$

On condition that $l_3^2/(16V^2) + (U/V)^3 \geqq 0$, the only real root of (7) is

$$C_a = \{l_3/(4V) + [l_3^2/(16V^2) + (U/V)^3]^{1/2}\}^{1/3}$$
$$+ \{l_3/(4V) - [l_3^2/(16V^2) + (U/V)^3]^{1/2}\}^{1/3}. \tag{8}$$

However, equality to $O(n^{-3/2})$ is obtained for the third cumulant if the term $O(C^3)$ in (7) is neglected and we obtain the simpler

$$C_b = l_3/(6U). \tag{9}$$

For $U \neq 0$ this solution yields equality of the variance and the third cumulant to $O(n^{-1})$ and $O(n^{-3/2})$, respectively. For the majority of special cases we have examined, the more complex C_a does not add much to the accuracy of the approximation, so the simpler C_b should be preferred.

Third, let

$$A_1 = 1 - C_1, \quad A_2 = 1 + C_2, \quad B_1 = -hC_1, \quad B_2 = -hC_2, \tag{10}$$

where $C_1 = C - D$, $C_2 = C + D$, and h, C and D have yet to be determined.

Introducing this solution into (2)-(4), we have

$$E(\hat{X}) = (C_1 + C_2)(M_1 - h/2) = C(2M_1 - h), \tag{2b}$$

$$V(\hat{X}) = 2M_2 + (C_2^2 + C_1^2)(M_2 - M_1^2 - hM_1 + h^2/4)$$

(3b)
$$-2C_2C_1(M_1^2 - hM_1 + h^2/4) + 2(C_2 - C_1)(M_2 - hM_1)$$
$$= 2M_2 + D^2(2M_2 - 4M_1h + h^2) + C^2(2M_2 - 4M_1^2) + 4D(M_2 - M_1h),$$

$$\hat{l}_3 = [(C_2^3 + C_1^3) + 3(C_2^2 - C_1^2) + 3(C_2 + C_1)](M_3 - 3M_2M_1 + 2M_1^3)$$
$$+ 3[(C_2^2 - C_1^2) - C_2C_1(C_2 + C_1) + (C_2 + C_1)](M_2M_1 - 2M_1^3)$$
$$- 3h(C_2^2 - C_1^2)(C_2 - C_1 + 2)(\tfrac{1}{2}M_2 - M_1^2)$$
(4b)
$$+ \tfrac{3}{2}h^2(C_2 - C_1)^2(C_2 + C_1)M_1$$
$$= 2C(C^2 + D^2 + 6D + 3)(M_3 - 3M_2M_1 + 2M_1^3)$$
$$+ 6C(-C^2 + D^2 + 2D + 1)(M_2M_1 - 2M_1^3)$$
$$+ 24hCD(D + 1)(\tfrac{1}{2}M_2 - M_1^2) + 12CD^2h^2M_1.$$

The expression for $\hat{l}_4$ is cumbersome and will not be given here.

From (2b) in order to have zero mean, we first have $h = 2M_1$.

Now assume that C is $O(n^{-1/2})$ and D is $O(n^{-1})$. Neglecting terms which are $O(n^{-1})$ in (4b) we obtain thereof

(11)
$$C = (l_3/6)/U = O(n^{-1/2}),$$

which is identical with C_b (9).

Introducing this solution into the expression for $\hat{l}_4$ and neglecting terms which are $O(n^{-2})$ we obtain for D

(12)
$$D = \{l_4 - (2M_4 - 3) - 12C^2[M_4 - 4M_3M_1 + 4M_2M_1^2]\}/[8(M_4 - hM_3)].$$

Under the above assumption this solution yields equality of the variance, the third and the fourth cumulants to $O(n^{-1})$, $O(n^{-1^{1/2}})$ and $O(n^{-2})$, respectively. Note that for D to be $O(n^{-1})$ we should have $M_4 = \frac{3}{2}$.

For a nonskewed X we have $C = 0$, and $D > 0$. However D is not equal to B (6) and the latter should be preferred when approximating nonskewed X since it results in an approximation identical with X to $O(B^2)$ in both its variance and fourth cumulant.

3. General approximations based on linear transformations of the unit normal variate. The partial moments of the standard normal variate, hereafter denoted z_1, are

$$M_1 = 1/(2\pi)^{1/2}, \quad M_2 = \tfrac{1}{2}, \quad M_3 = (2/\pi)^{1/2}, \quad M_4 = \tfrac{3}{2}.$$

Introducing first into (6) we obtain a normal approximation for a nonskewed variate

$$\hat{x} = \begin{cases} Az_1 + B, & z_1 < 0, \\ Az_1 - B, & z_1 > 0, \end{cases}$$

where

$$A = 1 + (2/\pi)^{1/2}B = 1 + (\tfrac{1}{4})l_4, \quad B = (\pi/32)^{1/2}l_4.$$

For a skewed variate in order to solve (7) we have

$$U = M_3 - 2M_2M_1 = \tfrac{1}{2}(2/\pi)^{1/2} = M_1 = 0.3989,$$
$$V = M_3 - 6M_2M_1 + 8M_1^3 = [(4 - \pi)/(2\pi)](2/\pi)^{1/2} = 0.1090,$$

hence from (8)

$$C_a = \{l_3/0.4360 + [l_3^2/0.1901 + 49.0132]^{1/2}\}^{1/3} + \{l_3/0.4360 - [l_3^2/0.1901 + 49.0132]^{1/2}\}^{1/3},$$

and from (9) and (11)

$$C_b = (\pi/18)^{1/2} l_3 = 0.4178 l_3 = C.$$

Also we have $h = 2M_1 = M_3 = 0.7979$.

Finally from (12) we have

$$D = (l_4 - 6.5408C^2)/6.9070 = 0.1448(l_4 - 1.1417 l_3^2).$$

In § 6 we apply these approximations to some commonly used distributions.

4. General approximations for the inverse cumulative function of a skewed variate. In this section we introduce for z several random variables with known inverse distribution function so that general approximations for the inverse DF of X can be derived. The headings of the subsections specify the respective z. The concluding subsection notes some of the considerations relevant to the choice of the appropriate approximation in applications.

4.1. Shore's approximations for the inverse unit normal distribution function. Recently we have presented several approximations for the inverse of the standard normal distribution function (Shore (1982)), among which the most accurate is

$$z_2 = -5.5310\{[(1-P)/P]^{0.1193} - 1\}, \qquad P \geqq \tfrac{1}{2} \tag{13}$$

and the simplest is

$$z_3 = -0.4506 \ln [(1-P)/P] + 0.2253 = 0.2253 \ln \{e[P/(1-P)]^2\}, \qquad P > \tfrac{1}{2}. \tag{14}$$

These approximations have $M_2 = \frac{1}{2}$, $M_4 = \frac{3}{2}$, identical with those of the unit normal.

The partial moments of (13) are (see detailed derivation in Appendix B)

$$M_1 = 0.4002, \quad M_2 = \tfrac{1}{2}, \quad M_3 = 0.7990, \quad M_4 = \tfrac{3}{2}.$$

These are very close to those of the standard normal distribution.

For (13) we have (see (7)) $U = 0.3988$, $V = 0.1112$; hence from (8),

$$C_a = \{l_3/0.4448 + [l_3^2/0.1978 + 46.1266]^{1/2}\}^{1/3} + \{l_3/0.4448 - [l_3^2/0.1978 + 46.1266]^{1/2}\}^{1/3},$$

and from (9) and (11)

$$C_b = 0.4179 l_3 = C.$$

Also we have $h = 2M_1 = 0.8004$. Finally from (12)

$$D = 0.1453(l_4 - 1.1344 l_3^2).$$

The partial moments of (14) are (see Appendix B)

$$M_1 = 0.4249, \quad M_2 = \tfrac{1}{2}, \quad M_3 = 0.7738, \quad M_4 = \tfrac{3}{2},$$

which lead to

$$U = 0.3489, \quad V = 0.1128, \quad h = 0.8498,$$

$$C_a = \{l_3/0.4512 + [l_3^2/0.2036 + 29.5921]^{1/2}\}^{1/3} + \{l_3/0.4512 - [l_3^2/0.2036 + 29.5921]^{1/2}\}^{1/3},$$

$$C_b = 0.4777 l_3 = C.$$

Finally

$$D = 0.1484(l_4 - 1.4949 l_3^2).$$

4.2. The logistic variate. Let

$$z_4 = 0.5513 \ln [P/(1-P)] \tag{15}$$

be a standardized logistic variate, the partial moments of which are

$$M_1 = 0.3821, \quad M_2 = \tfrac{1}{2}, \quad M_3 = 0.9064, \quad M_4 = 2.0996.$$

Introducing into (8), (9) and (12) we get

$$U = 0.5243, \quad V = 0.2064, \quad h = 0.7642,$$

$$C_a = \{l_3/0.8256 + [l_3^2/0.6816 + 16.3912]^{1/2}\}^{1/3} + \{l_3/0.8256 - [l_3^2/0.6816 + 16.3912]^{1/2}\}^{1/3},$$

$$C_b = 0.3179 l_3 = C.$$

Finally

$$D = 0.0888(l_4 - 1.2 - 1.2203 l_3^2).$$

4.3. The uniform variate. A standardized uniform variate on the interval (0, 1) is

$$z_5 = (12)^{1/2}(P - \tfrac{1}{2}), \tag{16}$$

which has partial moments

$$M_1 = 0.4330, \quad M_2 = \tfrac{1}{2}, \quad M_3 = 0.6495, \quad M_4 = 0.9.$$

We obtain

$$U = 0.2165, \quad V = 0, \quad h = 0.8660.$$

Therefore

$$C_a = C_b = C = 0.7698 l_3$$

which yields equality with the approximated variable of the third cumulant!

Finally

$$D = 0.3703(l_4 + 1.2 - 1.0670 l_3^2).$$

z_5 approximates x as a linear function of its distribution function with both mean and the third cumulant preserved. Yet the density function of the resultant approximation is a constant. Expectedly, the accuracy associated with this approximation will for the majority of approximated distributions be below acceptable standards, therefore it will not be elaborated upon any more in this paper. Notwithstanding, we chose to present the approximation here for two reasons. First, it occasionally occurs in simulation

studies that we wish to replace a constant factor by a random one that will only roughly exhibit the characteristic behaviour of the original agent. If accuracy is not of prime concern and the distribution of the simulated factor is unspecified, the above approximation may prove an easy and simple application tool for the aforementioned purpose. Second, since the uniform approximation, when differentiated, results in a constant it may well replace more complicated expressions that appear in the target function of optimization models, thus assisting in deriving closed form solutions. However, the robustness of the optimal solution to deviations from the exact values should be carefully studied in order to avoid improper application of this approximation.

4.4. Comparison of the approximations for applicability—a note. When applied to individual distributions the above general approximations may differ on accuracy. Yet even when no dominance may be noticed in terms of the latter, we hold the view that no approximation may a priori be discarded on grounds of redundancy. In attempting to derive closed form solutions to stochastic models some approximations may prove fruitful in certain cases and some in others. For example: z_2 by our experience usually results in the more accurate approximations. However z_3 obviously leads to more tractable solutions since on differentiating it with respect to $P/(1-P)$ an expression linear in this term results, from which the optimal value of P may be easily isolated (see a demonstration to inventory analysis in § 7).

Consequently, while the z_i of this section have been introduced in an increasing order of simplicity (as judged for instance by the criterion referred to above) they are most likely to be associated with a decreasing order of accuracy. A practitioner should be advised to select the approximation not only to suit the needs of accuracy but also in accordance with the simplicity requirements, as presented by the case on hand.

5. General approximations for the inverse cumulative function of a nonskewed variate. Introducing the partial moments of the z_i of § 4 into (6) we obtain

for z_2: $B=0.3112l_4$, $A=1+\frac{1}{4}l_4$,
for z_3: $B=\frac{1}{4}l_4$, $A=1+0.2124l_4$,
for z_4: $B=0.1791(l_4-1.2)$, $A=1+0.1368(l_4-1.2)$.

6. Approximations for the inverse distribution function of individual distributions.

6.1. The binomial variate. Let x be a standardized binomial with parameters (n, p), that is

$$x=(y-\mu)/\sigma=(y-np)/[np(1-p)]^{1/2},$$

where y, here and in subsequent subsections, is the unstandardized deviate.

For x we have

$$l_3=(1-2p)/\sigma, \qquad l_4=[1-6p(1-p)]/\sigma^2.$$

As is common practice in approximating the binomial by a continuous variate (see for example Benedetti (1956), we apply a continuity correction, and to approximate x substract from each of the above approximations the term $1/(2\sigma)$. To conserve space and in view of an earlier remark that solution (9) yields accuracy comparable to that of the more complex solution (8), the latter is discarded here and in subsequent subsections. The interested reader may work out this solution for himself from the respective general equations given in earlier sections.

The resultant approximations are:
For solution (9):

$$\hat{x}=[1+0.4178(1-2p)/\sigma]z_1-(5-4p)/(6\sigma), \qquad z_1>0,$$
$$\hat{x}=[1+0.4179(1-2p)/\sigma]z_2-(5-4p)/(6\sigma), \qquad z_2>0,$$
$$\hat{x}=[1+0.4777(1-2p)/\sigma]z_3-(0.9059-0.8118p)/\sigma, \qquad z_3>0,$$
$$\hat{x}=[1+0.3179(1-2p)/\sigma]z_4-(0.7429-0.4859p)/\sigma, \qquad z_4>0.$$

For solution (10):

$$\hat{x}=\begin{cases}(1-C_1)z_i-hC_1-1/(2\sigma), & z_i<0,\\ (1+C_2)z_i-hC_2-1/(2\sigma), & z_i>0,\end{cases}$$

where

$$C_1=C-D, \quad C_2=C+D, \quad h=2M_1.$$

For z_1 we have

$$C=0.4178(1-2p)/\sigma, \quad D=-(1/\sigma^2)[0.0205+0.2075p(1-p)], \quad h=0.7979,$$

for z_2 we have

$$C=0.4179(1-2p)/\sigma, \quad D=-(1/\sigma^2)[0.0195+0.2125p(1-p)], \quad h=0.8004,$$

for z_3 we have

$$C=0.4777(1-2p)/\sigma, \quad D=-(1/\sigma^2)[0.0734+0.0030p(1-p)], \quad h=0.8498,$$

for z_4 we have

$$C=0.3179(1-2p)/\sigma, \quad D=-(1/\sigma^2)[0.0196+0.1066\sigma^2+0.0993p(1-p)], \quad h=0.7642.$$

Some comparative values of $\hat{x}$ for the various approximations, together with values derived from the standard commonly used approximation

$$\hat{x}_0=z_1-1/(2\sigma),$$

are given in Table 1. Values are the unstandardized $\hat{x}$. The upper part of the table contains values derived from (9) while the lower values derived from solution (10) (same order is preserved in all subsequent tables). Values of $\hat{x}$ derived from z_1 are missing (here and in subsequent tables) since they are nearly identical with those derived from z_2. Also only $P>\frac{1}{2}$ is referred to, because the same accuracy obtains for $P<\frac{1}{2}$.

Table 1 shows no uniform dominance in terms of accuracy, although z_2 is usually the more accurate. Expectedly solution (10) yields better accuracy than solution (9) though the simpler latter results in accuracy which probably is satisfactory for most applications.

For nonextreme values (say $P<0.99$) all of the approximations constitute a vast improvement in accuracy over the traditional approximation, $\hat{x}_0$, without losing in simplicity since linearity of the approximating variable is preserved. Uniform dominance over $\hat{x}_0$ is exhibited by z_2.

Finally, note that the parameters we have chosen to show in Table 1 represent distributions which are far from normal (σ is small and p is distant from 0.5). Notwithstanding the accuracy achieved is high, far tail probabilities not excluded.

TABLE 1
Comparative values of approximations for the binomial.

	$n=5$ $p=.05$			$n=5$ $p=.25$			
P	0.7738	0.9774	0.9988	0.6328	0.8965	0.9844	0.9990
Z_1	0.75	2.00	3.04	0.34	1.26	2.15	3.08
Exact	0	1	2	1	2	3	4
App. $\hat{x}_0$	−.410	0.199	0.706	1.063	1.954	2.815	3.716
z_2	0.101	1.178	2.084	0.992	2.061	3.123	4.238
z_3	0.100	1.148	2.371	1.115	1.993	3.073	4.576
z_4	0.056	1.138	2.399	0.967	1.970	3.204	4.921
z_2	0.103	1.105	1.947	1.020	2.033	3.040	4.096
z_3	0.110	0.986	2.007	1.144	1.967	2.978	4.386
z_4	0.065	1.004	2.099	1.033	1.909	2.987	4.486

	$n=15$ $p=.15$			$n=25$ $p=.10$				
P	0.6042	0.9383	0.9964	0.7636	0.9020	0.9666	0.9905	0.9977
Z_1	0.26	1.54	2.69	0.72	1.29	1.83	2.35	2.83
Exact	2	4	6	3	4	5	6	7
App. $\hat{x}_0$	2.248	4.018	5.609	3.247	4.102	4.912	5.692	6.412
z_2	1.972	4.085	6.045	3.057	4.093	5.087	6.050	6.962
z_3	2.180	3.959	6.204	3.094	3.982	4.953	6.040	7.249
z_4	1.954	3.989	6.557	2.940	3.952	5.060	6.300	7.679
z_2	1.990	4.060	5.981	3.059	4.080	5.061	6.010	6.909
z_3	2.203	3.927	6.102	3.098	3.963	4.909	5.968	7.146
z_4	2.045	3.863	6.158	2.961	3.870	4.865	5.978	7.217

6.2. The Poisson variate. Let x be a standardized Poisson with parameter λ, that is $x=(y-\lambda)/\lambda^{1/2}$.

Since the Poisson is the limiting case of the binomial (that is $p\to 0$ and $n\to\infty$ so that $pn=\sigma^2=\text{const.}=\lambda$), the former may be approximated by setting $p=0$ and $np=\lambda$ in the approximations for the latter.

To conserve space these approximations are not given here. However, they are utilized to form Table 2, where some comparative values of the approximate unstandardized $\hat{x}$ are presented. Again we also exhibit values derived from the traditional

$$\hat{x}_0=z_1-1/(2\sigma).$$

The pattern revealed in Table 2 is similar to that of Table 1. z_2 usually gives the more accurate results, while among the logistic variates (z_3 and z_4) z_3 is better. Note the improvement in the accuracy for far tail probabilities of approximations z_3 and z_4 as we move from solution (9) to (10). This improvement, unnoticeable for z_2, is due probably to the relatively longer tail of the logistic distribution as compared with that of the standard normal (recall that solution (10) gives near equality of both l_3 and l_4).

TABLE 2
Comparative values of approximations for the Poisson.

	$\lambda = 0.5$				$\lambda = 1.0$			
P	0.607	0.910	0.986	0.998	0.736	0.920	0.981	0.996
Z_1	0.27	1.34	2.20	2.88	0.63	1.41	2.08	2.65
Exact	0	1	2	3	1	2	3	4
App. $\hat{x}_0$	−.016	0.740	1.349	1.829	1.130	1.910	2.580	3.150
z_2	−.020	1.166	2.142	2.923	1.068	2.148	3.109	3.947
z_3	0.093	1.096	2.132	3.178	1.110	2.053	3.053	4.101
z_4	0.003	1.064	2.161	3.268	1.002	2.032	3.123	4.266
z_2	−.006	1.152	2.104	2.865	1.072	2.136	3.084	3.911
z_3	0.138	1.053	1.998	2.952	1.122	2.018	2.969	3.964
z_4	0.057	1.012	1.998	2.994	1.027	1.958	2.945	3.978
	$\lambda = 2.0$			$\lambda = 3.0$				
P	0.677	0.947	0.995	0.647	0.815	0.966	0.996	
Z_1	0.46	1.62	2.57	0.38	0.90	1.82	2.65	
Exact	2	4	6	3	4	6	8	
App. $\hat{x}_0$	2.297	3.937	5.281	3.369	4.270	5.864	7.301	
z_2	2.022	4.115	5.910	2.995	4.094	6.080	7.900	
z_3	2.151	3.978	6.033	3.195	4.068	5.924	8.086	
z_4	1.964	4.010	6.312	2.942	3.933	6.039	8.493	
z_2	2.026	4.103	5.885	3.000	4.092	6.069	7.879	
z_3	2.166	3.943	5.941	3.210	4.067	5.887	8.007	
z_4	2.022	3.874	5.957	3.026	3.922	5.828	8.046	

6.3. The gamma variate (including chi-square). For the gamma distribution with parameters (α, r), the mean is r/α, the standard deviation is $(r/\alpha^2)^{1/2}$, and the kth cumulant of the standardized variate is $l_k = (k-1)!\, r^{1-k/2}$. We obtain

$$l_3 = 2r^{-1/2}, \qquad l_4 = 6r^{-1}.$$

In particular, for a χ^2 variate with ν degrees of freedom ($r = \nu/2$, $\alpha = \frac{1}{2}$) we obtain

$$\mu = \nu, \quad \sigma = (2\nu)^{1/2}, \quad l_3 = (8/\nu)^{1/2} \quad \text{and} \quad l_4 = 12/\nu.$$

Introducing into the approximations of § 3 and 4 we get the following.

For solution (9):

$$\hat{x} = [1 + 1.1817/(\nu)^{1/2}]z_1 - 0.9429/(\nu)^{1/2}, \qquad z_1 \geqq 0,$$

$$\hat{x} = [1 + 1.1820/(\nu)^{1/2}]z_2 - 0.9461/(\nu)^{1/2}, \qquad z_2 \geqq 0,$$

$$\hat{x} = [1 + 1.3511/(\nu)^{1/2}]z_3 - 1.1482/(\nu)^{1/2}, \qquad z_3 \geqq 0,$$

$$\hat{x} = [1 + 0.8992/(\nu)^{1/2}]z_4 - 0.6872/(\nu)^{1/2}, \qquad z_4 \geqq 0.$$

For solution (10):

$$\hat{x} = \begin{cases} (1 - C_1)z_i - hC_1, & z_i < 0, \\ (1 + C_2)z_i - hC_2, & z_i > 0, \end{cases}$$

where

$$C_1 = C - D, \quad C_2 = C + D, \quad h = 2M_1.$$

For z_1 we have

$$C = 1.1817/\nu^{1/2}, \quad D = 0.4151/\nu, \quad h = 0.7979,$$

for z_2 we have

$$C = 1.1820/\nu^{1/2}, \quad D = 0.4250/\nu, \quad h = 0.8004,$$

for z_3 we have

$$C = 1.3511/\nu^{1/2}, \quad D = 0.0060/\nu, \quad h = 0.8498,$$

for z_4 we have

$$C = 0.8992/\nu^{1/2}, \quad D = 0.1987/\nu - 0.1066, \quad h = 0.7642.$$

Some comparative values of these approximations, together with values derived from the simple $\hat{x}_0 = z_1$ are given in Table 3. For solution (9) the logistic approximations seem to be the more accurate, z_4 slightly better than z_3. For solution (10) it is z_3 which is usually the more accurate.

TABLE 3
Comparative values of approximations for the χ^2.

	$\nu = 10$					
P	0.7	0.8	0.9	0.95	0.99	0.995
Z_1	0.524	0.842	1.282	1.645	2.326	2.576
Exact	11.781	13.442	15.987	18.307	23.209	25.188
App. $\hat{x}_0$	12.345	13.764	15.731	17.356	20.404	21.519
z_2	11.929	13.842	16.498	18.727	23.002	24.572
z_3	12.251	13.801	16.134	18.283	23.030	25.038
z_4	11.711	13.418	15.986	18.352	23.579	25.790
z_2	11.878	13.850	16.588	18.887	23.294	24.912
z_3	12.250	13.801	16.135	18.285	23.034	25.043
z_4	11.826	13.418	15.812	18.019	22.893	24.955
	$\nu = 25$					
P	0.7	0.8	0.9	0.95	0.99	0.995
Z_1	0.524	0.842	1.282	1.645	2.326	2.576
Exact	28.172	30.675	34.382	37.652	44.314	46.928
App. $\hat{x}_0$	28.708	30.951	34.062	36.631	41.450	43.214
z_2	28.311	31.033	34.812	37.985	44.069	46.302
z_3	28.829	31.010	34.292	37.317	43.997	46.823
z_4	27.925	30.404	34.134	37.571	45.163	48.374
z_2	28.279	31.038	34.869	38.086	44.253	46.517
z_3	28.829	31.010	34.293	37.318	44.000	46.826
z_4	28.132	30.404	33.822	36.972	43.929	46.872

6.4. Fisher's $\hat{z}$ and the F distribution. Fisher's $\hat{z}$ is defined by $\hat{z}=0.5\ln F$, where F has an F distribution with ν_1 and ν_2 degrees of freedom. Let

$$r_1=1/(\nu_1-1),\quad r_2=1/(\nu_2-1),\quad \mu=0.5(r_2-r_1),\quad \sigma=[0.5(r_1+r_2)]^{1/2};$$

standardization of $\hat{z}$ is based on Fisher's normal approximation to $\hat{z}$, and r_1 and r_2 are defined in accordance with Fisher's suggestion so as to normalize the standardized $\hat{z}$ (for details see Johnson and Kotz (1970, Ch. 26, Section 4)).

From Wishart (1947), an approximate formula for the cumulants of the standardized $\hat{z}$ is

$$l_k=0.5(k-2)![r_2^{k-1}+(-1)^k r_1^{k-1}]/\sigma^k.$$

Thus

$$l_3=2\mu/\sigma,\qquad l_4=2\sigma^2+6(\mu/\sigma)^2.$$

Introducing these into the equations of §§ 3 and 4 simple approximations for $\hat{z}$ ensue. However, of more interest to applied statisticians are approximations for the inverse of the F distribution. These are easily derived from the above approximations. In particular, let us consider that based on z_3 and solution (9):

$$(\hat{z}-\mu)/\sigma=\begin{cases}(1-0.4777l_3)z_3-0.4059l_3, & z_3\leqq 0,\\ (1+0.4777l_3)z_3-0.4059l_3, & z_3\geqq 0.\end{cases}$$

TABLE 4
Comparative values of the approximation for F.

	$\nu_1=10$			$\nu_2=5$		
P	0.75	0.90	0.95	0.975	0.99	0.995
Exact	1.89	3.30	4.74	6.62	10.10	13.60
$\hat{F}$	2.08	3.39	4.72	6.48	9.80	13.34
	$\nu_1=30$			$\nu_2=30$		
P	0.75	0.90	0.95	0.975	0.99	0.995
Exact	1.28	1.61	1.84	2.07	2.39	2.63
$\hat{F}$	1.31	1.57	1.78	2.01	2.35	2.64
	$\nu_1=20$			$\nu_2=10$		
P	0.75	0.90	0.95	0.975	0.99	0.995
Exact	1.52	2.20	2.77	3.42	4.41	5.27
$\hat{F}$	1.59	2.17	2.68	3.29	4.28	5.21
	$\nu_1=40$			$\nu_2=24$		
P	0.75	0.90	0.95	0.975	0.99	0.995
Exact	1.30	1.64	1.89	2.15	2.49	2.77
$\hat{F}$	1.33	1.61	1.84	2.08	2.45	2.77

Introducing $\hat{z}$ in terms of F, a simple approximation for F in terms of its distribution function is readily derived:

$$\hat{F} = [P/(1-P)]^{0.9012\sigma+\delta_P 0.8610\mu} * \exp[\delta P 0.4506\sigma + 0.8069\mu]$$

where

$$\delta P = \begin{cases} -1, & P < \frac{1}{2}, \\ 1, & P > \frac{1}{2}. \end{cases}$$

Some comparative values of $\hat{F}$ are introduced in Table 4.

6.5. The *t* distribution. For a t variate with ν degrees of freedom we have all odd order cumulants identically zero. For the standardized t, that is $x = t(1-2/\nu)^{1/2}$ we have $l_4 = 6/(\nu-4)$.

Introducing it into the expressions of § 3 and § 5 we get:

for z_1: $A = 1+(3/2)/(\nu-4)$, $\quad B = 6(\pi/32)^{1/2}/(\nu-4)$,

for z_2: $A = 1+(3/2)/(\nu-4)$, $\quad B = 1.8672/(\nu-4)$,

for z_3: $A = 1+1.2744/(\nu-4)$, $\quad B = (3/2)/(\nu-4)$,

for z_4: $A = 0.8208/(\nu-4)+0.8358$, $\quad B = 1.0746/(\nu-4)-0.2149$.

Values of the approximate unstandardized t, together with values derived from $\hat{x}_0 = z_1$ are shown in Table 5. Though for small P z_2 performs worse than z_3 and z_4,

TABLE 5

Comparative values of the approximations for t.

	$\nu = 10$					
P	0.70	0.80	0.90	0.95	0.99	0.995
Z_1	0.524	0.842	1.282	1.645	2.326	2.576
t exact	0.542	0.879	1.372	1.812	2.764	3.169
App. $\hat{x}_0$	0.586	0.941	1.433	1.839	2.601	2.880
z_2	0.395	0.830	1.434	1.942	2.914	3.271
z_3	0.543	0.873	1.368	1.824	2.832	3.259
z_4	0.548	0.871	1.357	1.805	2.795	3.213
	$\nu = 20$					
P	0.70	0.80	0.90	0.95	0.99	0.995
Z_1	0.524	0.842	1.282	1.645	2.326	2.576
t exact	0.553	0.860	1.325	1.725	2.528	2.845
App. $\hat{x}_0$	0.553	0.887	1.351	1.734	2.452	2.715
z_2	0.490	0.849	1.347	1.766	2.568	2.863
z_3	0.592	0.868	1.284	1.667	2.514	2.872
z_4	0.593	0.870	1.288	1.674	2.525	2.885

no uniform dominance may be noticed for higher values. All three of the new approximations obviously dominate $\hat{x}_0$ in terms of accuracy, having the additional advantage of expressing x explicitly in terms of its distribution function.

7. Simple general approximations for the loss function of a random variable (continuous or discrete) and an application to the Poisson distribution. The loss function of a continuous variable, Y, is defined by

$$L(y)=\int_y^\infty (t-y)f(t)\,dt=\sigma\int_u^\infty (x-u)f(x)\,dx=\sigma\int_u^\infty [1-F(x)]\,dx, \tag{17}$$

where $f(\cdot)$ and $F(\cdot)$ are the density and cumulative distribution functions, respectively, and x and u are in standard units.

Likewise for a discrete variable we have

$$L(y)=\sum_{t=y}^{\infty}(t-y)f(t)=\sigma\sum_{u}^{\infty}(x-u)f(x)=\sigma\sum_{u}^{\infty}[1-F(x)]. \tag{17a}$$

The loss function plays a central role in many stochastic optimization models which incorporate it as a major component of their target function. Outstanding examples are inventory control problems, like the well-known "newsboy problem", and optimization problems associated with Bayesian statistics. Deriving a simple approximation for $L(y)$ may enhance the development of closed form solutions for these problems.

To start with, we first develop general approximations for the loss function of a continuous variable.

In § 2 we introduced a general approximation for x based on solution (9)

$$\hat{x}=\begin{cases}(1-c)z-hc, & z\leqq 0,\\ (1+c)z-hc, & z\geqq 0.\end{cases} \tag{18}$$

Deriving dx from (18), introducing it in terms of dz into the rightmost wing of (17) and integrating, we obtain for $P<\frac{1}{2}$

$$L(y)=\sigma\{(1-c)[L_Z(P)-L_Z(\tfrac{1}{2})]+(1+c)L_Z(\tfrac{1}{2})\}=\sigma\{(1-c)L_Z(P)+2cL_Z(\tfrac{1}{2})\} \tag{19a}$$

and for $P>\frac{1}{2}$

$$L(y)=\sigma(1+c)L_Z(P) \tag{19b}$$

where $L_Z(P)$ is the loss function of Z at $G(z)=P$.

Let us now introduce the various approximating z_i presented in §§ 3 and 4, together with the respective c.

First, approximating $L_Z(P)$ for the standard normal deviate from Shore (1982)

$$L_Z(P)=\begin{cases}0.4115P/(1-P)-z_1, & P\leqq\frac{1}{2},\\ 0.4115(1-P)/P, & P\geqq\frac{1}{2},\end{cases}$$

where $P=\Phi(z_1)$, and $\Phi(\cdot)$ is the cumulative standard normal distribution function, we obtain

$$L(y)=\begin{cases}\sigma[(1-0.4178l_3)0.4115P/(1-P)-x], & P<\frac{1}{2},\\ \sigma[(1+0.4178l_3)0.4115(1-P)/P], & P>\frac{1}{2}.\end{cases} \tag{20}$$

Note that to derive (20) we introduced from (18) $\sigma(1-0.4178l_3)z_1=\sigma(\hat{x}+hc)=\sigma[\hat{x}+(\frac{1}{3})l_3]$, and also put the exact

$$L_Z(1/2)=0.3989.$$

Next we derive the loss function of z_2, z_3 and z_4, as defined in § 4.

To do that, note that

$$L_Z(P) = \int_u^\infty [1 - G(z)]\, dz = \int_P^1 (1 - G)[\partial z / \partial G]\, dG.$$

Thus for z_2 we obtain

$$L_Z(P) = \begin{cases} 0.6598 \int_P^1 G^{-0.8807}(1-G)^{-0.1193}\, dG, & P < \frac{1}{2}, \\ 0.6598 \int_P^1 \{[(1-G)/G]^{0.1193}/G\}\, dG, & P > \frac{1}{2} \end{cases}$$

which is most unlikely to lead to any useful results in the sense expounded above.

For z_3 we have

$$L_Z(P) = 0.4506 \int_P^1 (1/G)\, dG = -0.4506 \ln P$$

which on introduction into (19) yields

$$L(y) = \begin{cases} \sigma\{-(1 - 0.4777 l_3) 0.4506 \ln P + 0.30 l_3\}, & P < \frac{1}{2}, \\ -\sigma(1 + 0.4777 l_3) 0.4506 \ln P, & P > \frac{1}{2}. \end{cases}$$

Likewise for z_4 we get

$$L(y) = \begin{cases} \sigma\{-(1 - 0.3179 l_3) 0.5513 \ln P + 0.24 l_3\}, & P < \frac{1}{2}, \\ -\sigma(1 + 0.3179 l_3) 0.5513 \ln P, & P > \frac{1}{2}. \end{cases}$$

To derive a simple approximation for the loss function of a discrete variate let us draw a graph of $1 - F(x)$ (the vertical axis) as a function of x (the horizontal axis). From simple geometric considerations it can be easily verified that

$$\begin{aligned} L(y) = \sum_{x=u}^{\infty} [1 - F(x)] &\approx \sigma \left\{ \int_u^\infty [1 - F(x)]\, dx + (1/\sigma) 0.5 \sum_{x=u+1/\sigma}^{\infty} f(x) \right\} \\ &= \sigma \left\{ \int_u^\infty [1 - F(x)]\, dx \right\} + 0.5[1 - F(x)]. \end{aligned} \tag{21}$$

The above first term on the right side is virtually the loss function of X were it regarded as a continuous variate, so that the second term may be considered a continuity correction. Introducing for the "continuous" loss function its approximation as derived above (19) we finally have for a discrete variate

$$L(y) = \begin{cases} \sigma\{(1-c) L_Z(P) + 2c L_Z(1/2)\} + 0.5(1-P), & P < \frac{1}{2}, \\ \sigma(1+c) L_Z(P) + 0.5(1-P), & P > \frac{1}{2}. \end{cases} \tag{22}$$

To demonstrate the accuracy associated with the above approximations we apply them to the Poisson loss function which is being in extensive use in the formulation of stochastic inventory control models.

Introducing the respective parameters in (22) we obtain for z_1

$$L(y) = \begin{cases} \sigma\{(1 - 0.4178/\sigma) 0.4115 P/(1-P) - x\} - 0.5P, & P < \frac{1}{2}, \\ \sigma(1 + 0.4178/\sigma) 0.4115(1-P)/P + 0.5(1-P), & P > \frac{1}{2}, \end{cases} \tag{23}$$

where we used $\sigma(1 - 0.4178/\sigma) z_1 = \sigma[x + hc + 1/(2\sigma)] = \sigma x + (5/6)$.

For far tail probabilities we may substitute

for $P<\frac{1}{2}$: $0.5P$ by $0.5P/(1-P)$,

for $P>\frac{1}{2}$: $0.5(1-P)$ by $0.5(1-P)/P$,

so that (23) simplifies to

$$L(y)=\begin{cases}\sigma\{(1-1.6329/\sigma)0.4115P/(1-P)-x\}, & P<\frac{1}{2},\\ \sigma(1+1.6329/\sigma)0.4115(1-P)/P, & P>\frac{1}{2}.\end{cases} \tag{23a}$$

Introducing for x in terms of P from one of the approximations of § 6, $L(y)$ is expressible in terms of P only, which is very useful in application to inventory control models.

For z_3 we obtain likewise

$$L(y)=\begin{cases}-\sigma(1-0.4777/\sigma)0.4506\ln P+0.8-0.5P, & P<\frac{1}{2},\\ -\sigma(1+0.4777/\sigma)0.4506\ln P+0.5(1-P), & P>\frac{1}{2},\end{cases} \tag{24}$$

and for z_4

$$L(y)=\begin{cases}-\sigma(1-0.3179/\sigma)0.5513\ln P+0.74-0.5P, & P<\frac{1}{2},\\ -\sigma(1+0.3179/\sigma)0.5513\ln P+0.5(1-P), & P>\frac{1}{2}.\end{cases} \tag{25}$$

Since (24) and (25) are of the same algebraic structure a choice between them has to be made in terms of accuracy. Comparison shows that in terms of maximum deviation (25) is uniformly dominant. Table 6 presents some values for approximations (23), (23a) and (25). In terms of maximum deviation (23) generally yields the most accurate

TABLE 6
Comparative values of the approximations for $L(y)$.

	$\sigma=2$									
y	0	1	2	3	4	5	6	7	8	9
P	0.0183	0.0916	0.2381	0.4335	0.6288	0.7851	0.8893	0.9489	0.9786	0.9919
$L(y)$ (exact)	4.0000	3.0185	2.1098	1.3481	0.7815	0.4103	0.1954	0.0848	0.0336	0.0123
App. (23)	4.0030	3.0199	2.0844	1.2815	0.7729	0.3798	0.1792	0.0791	0.0325	0.0122
App. (23a)	4.0028	3.0152	2.0472	1.1156	0.8825	0.4092	0.1861	0.0805	0.0327	0.0122
App. (25)	4.4410	2.9108	1.9517	1.2984	0.7785	0.4166	0.2053	0.0926	0.0383	0.0144
	$\sigma=4$									
y	8	10	12	14	16	18	20	22	24	
P	0.0220	0.0774	0.1931	0.3675	0.5660	0.7423	0.8682	0.9418	0.9777	
$L(y)$ (exact)	8.0159	6.0812	4.2852	2.7529	1.5872	0.8114	0.3672	0.1464	0.0504	
App. (23)	8.0222	6.0850	4.2562	2.6727	1.6110	0.7600	0.3419	0.1414	0.0526	
App. (23a)	8.0219	6.0817	4.2331	2.5659	1.7774	0.8047	0.3519	0.1432	0.0529	
App. (25)	8.4767	5.8954	3.9818	2.5883	1.5719	0.8382	0.4023	0.1718	0.0648	

results, but as expected no meaningful differences in accuracy may be noticed between (23) and (23a) for tail probabilities. Equation (25) performs best in middle range probabilities.

To demonstrate the applicability of the above approximations we take a simple standard model where inventory supply is periodical, and it is required to determine the reorder level and order quantity which simultaneously minimize the total cost per unit time. The well-known fundamental equation is

$$T = Sd/Q + (R - ld)IC + ICQ/2 + (\pi d/Q)L(R),$$

where S is the order cost; d the mean demand per unit time; l the mean lead time; C the unit cost; I the interest rate; π the loss of revenue per item in case of shortage; R the reorder level; Q the order quantity; and T the total cost per unit time. Demand in the lead time is assumed to be Poisson with mean $\sigma^2 = ld$. To find the optimal reorder level, R^*, let us introduce for $(R - ld)$ from $\hat{x}$ of the Poisson based on z_3, and for $L(R)$ from (23a) (assuming $P^* = F[(R^* - ld)/\sigma] > 0.5$), to obtain

$$\begin{aligned} T = {} & Sd/Q + IC\sigma\{(1 + 0.48/\sigma)[0.4506 \ln [P/(1-P)] + 0.2253] - 0.91/\sigma\} \\ & + ICQ/2 + (\pi d/Q)\sigma(1 + 1.63/\sigma)0.4115(1-P)/P. \end{aligned} \tag{26}$$

Differentiating with respect to $(1-P)/P$ and with respect to Q and equating to zero we obtain

$$(1 - P^*)/P^* = 1.0950(ICQ^*/\pi d)[(\sigma + 0.48)/(\sigma + 1.63)], \tag{27}$$

$$(Q^*)^2 = (2/IC)[Sd + \pi d(\sigma + 1.63)0.4115(1 - P^*)/P^*] \tag{27a}$$

from which

$$Q^* = \sigma\{0.4506(1 + 0.48/\sigma) + [0.2030(1 + 0.48/\sigma)^2 + 2Sd/(IC\sigma^2)]^{1/2}\}. \tag{28}$$

Introducing (28) into (27) and then into $\hat{x}$, we obtain the optimal value of the reorder level

$$R^* = ld - (\sigma + 0.48)\{0.4506 \ln [(1 - P^*)/P^*] - 0.2253\} - 0.91. \tag{29}$$

To demonstrate the accuracy of this solution let $S = 50$, $d = 20$ items, $l = \frac{1}{2}$ time unit, $C = 2$, $I = 0.10$, $\pi = 5$, $\sigma^2 = ld = 10$. Then $Q^* = 101.6$ (28), $(1 - P^*)/P^* = 0.1692$ (27), and from (29) $R^* = 12.83$. The exact optimal solution is $Q^* = 102$, $R^* = 13$ with an associated cost of $T = 20.9201$.

To demonstrate the usefulness of the above solution, suppose that costs incurred by a change in the current inventory policy leave us indifferent to a deviation of $\pm p\%$ in Q^*. What then is the permissible range of variation of the relevant parameters of the model (assuming σ unchanged)? From (28) we find:

$$\begin{aligned} (Sd)/(IC) &= \{[Q^*/\sigma - 0.4506(1 + 0.48/\sigma)]^2 - 0.2030(1 + 0.48/\sigma)^2\}(\sigma^2/2) \\ &= 0.5\{(Q^* - 1.6164)^2 - 2.6123\}. \end{aligned}$$

Thus for a permissible change in Q^* of say $\pm 30\%$ from its current value, the relevant Sd/IC may be allowed to vary from -51% to $+72\%$ from its current value before a change in policy needs to take place. Finally note that (28) implies Q^* being independent of π, unlike R^* which increases with π (see (29)).

8. Two approximate solutions in statistical inference.

8.1. Determining sample size in estimating the ratio of the variances of two normal populations—an approximate solution. Let σ_1^2 and σ_2^2 be the unknown variances of two

normal populations. Let $\Delta = \sigma_1^2/\sigma_2^2$ be the ratio of the two variances, which has to be estimated by two random samples of equal size n. It is required to choose n so that the positive relative sampling error will not exceed $p_1\%$ with probability $(1-\alpha_1)$, and that the relative negative sampling error will not exceed $p_2\%$ with probability $(1-\alpha_2)$.

Let $F_{1-\alpha}(n-1, n-1)$ be the $(1-\alpha)100$ percentile of an F distribution with $\nu_1 = \nu_2 = n-1$ degrees of freedom and let $\hat{\Delta} = S_1^2/S_2^2$ be the samples' ratio.

Since $\Delta/\hat{\Delta}$ is distributed as F with $\nu_1 = \nu_2 = n-1$ degrees of freedom we obtain the following conditions:

$$F_{1-\alpha_1}(n-1, n-1) \leqq 1 + p_1/100, \tag{30}$$

$$F_{\alpha_2}(n-1, n-1) \geqq 1 - p_2/100. \tag{31}$$

In order to determine a proper n we use $\hat{F}$, where $\mu = 0$, and solve (30) and (31), each separately, for the equality sign to obtain

$$n_i = \{0.4506 \ln [[(1-\alpha_i)/\alpha_i]^2 e]/[\ln (1 \pm p_i/100)]\}^2 + 2 \quad (+ \text{ for } i=1, - \text{ for } i=2). \tag{32}$$

To select the sample size, choose max (n_1, n_2).

To improve the accuracy of the above solution the numerical coefficient in (32) was readjusted for the commonly used range: $\alpha_i \leqq 0.10$ to give 0.4630.

The accuracy of this solution is demonstrated in Table 7.

TABLE 7

Approximate sample size needed to estimate Δ—some representative values.

	$\alpha_1 = 0.10$		$\alpha_1 = 0.05$		$\alpha_1 = 0.01$	
ν exact	p_1 (%)	ν app.	p_1 (%)	ν app.	p_1 (%)	ν app.
10	132	10.1	198	9.8	385	10.1
15	97	15.1	140	14.7	252	15.3
20	79	20.1	112	19.5	194	20.5
30	61	29.5	84	29.2	139	30.9
60	40	58.1	53	58.9	84	62.0

8.2. Optimal hypothesis testing—an approximate solution.

8.2.1. Introduction. Let θ be an unknown parameter of a statistical population.

We wish to test the null hypothesis H_0: $\theta = \theta_0$, assumed to be true with probability p_0, against the one-sided simple alternative hypothesis H_1: $\theta = \theta_1 < \theta_0$, assumed to be true with probability $p_1 = 1 - p_0$. Let the decision criterion of the test be: If $\hat{\theta} < \theta_{\text{cr.}}$ – reject H_0, otherwise do not reject, where $\hat{\theta}$ is an unbiased most efficient estimate of θ, based on a random sample of n observations, and $\theta_{\text{cr.}}$ is the boundary (critical) value of the acceptance region of the test. $\theta_{\text{cr.}}$ is so chosen that

$$P_r(\hat{\theta} < \theta_{\text{cr.}}/H_0) = \alpha, \qquad P_r(\hat{\theta} \geqq \theta_{\text{cr.}}/H_1) = \beta,$$

and α and β are respectively the type 1 and type 2 error probabilities associated with the test.

It is common practice among applicants to choose the decision variables of the test (namely n, α and β) arbitrarily, with little or no regard to the sampling cost and to the costs (gains) incurred by wrong (correct) decisions.

Here we formulate, within the framework of traditional statistical hypothesis testing, a simple optimization model which takes account of the above costs. By

employing some of the approximations derived earlier we arrive at explicit expressions for the optimal values of the decision variables of the test and demonstrate its accuracy for the special case where θ is an unknown variance of a normal population.

8.2.2. The model. Let the costs (gains) of the test be:

G_0-the gain realized by correctly not rejecting H_0,
C_0-the cost of making a type 1 error,
G_1-the gain realized by correctly rejecting H_0,
C_1-the cost of making a type 2 error,
$C(n)$-the cost of sampling n observations.

Except for $C(n)$ all other costs and gains are assumed independent of α, β and n.

To test optimally the hypotheses under consideration, we seek an optimal solution (α^*, β^*, n^*) which maximizes the expected value of the net gain, that is

$$\max\{EV = p_0[(1-\alpha)G_0 - \alpha C_0] + p_1[(1-\beta)G_1 - \beta C_1] - C(n) = E_p - p_0[\alpha D_0 - \beta D_1] - \beta D_1 - C(n)\} \tag{33}$$

subject to

$$\theta_{\text{cr.}} = g(\alpha) = h(\beta), \tag{34}$$

where

$$D_0 = G_0 + C_0,$$
$$D_1 = G_1 + C_1,$$
$$E_p = p_0 G_0 + (1-p_0)G_1,$$

and $g(\alpha)$ and $h(\beta)$ are the boundary values of $\hat{\theta}$, expressed in terms of α and β, respectively.

8.2.3. Testing optimally for the standard deviation of a normal population. Let σ be an unknown standard deviation of a normal population.

To test the null hypothesis H_0: $\sigma = \sigma_0$ against H_1: $\sigma = \sigma_1 < \sigma_0$, a random sample of n observations is drawn and the sample standard deviation, S, is calculated.

The optimization problem is to find (α, β, n) that maximizes (33) subject to the feasibility condition

$$\sigma_0^2 \chi_\alpha^2(\nu) = \sigma_1^2 \chi_{1-\beta}^2(\nu) \tag{34a}$$

where $\chi_F^2(\nu)$ is the Fth fractile of a chi-square distribution with $\nu = n-1$ degrees of freedom.

Introducing for the unstandardized χ^2 from § 6.3 (using z_3) we obtain

$$\chi_F^2 = \begin{cases} [(2\nu)^{1/2} - 1.9107]z_F + \nu - 1.6238, & F < \frac{1}{2}, \\ [(2\nu)^{1/2} + 1.9107]z_F + \nu - 1.6238, & F > \frac{1}{2}, \end{cases} \tag{35}$$

where z_F is the Fth fractile of z_3. Presenting into (34a) we have

$$Q = (\sigma_0^2 - \sigma_1^2)\nu + (\sigma_0^2 z_\alpha - \sigma_1^2 z_{1-\beta})(2\nu)^{1/2} - 1.9107[\sigma_0^2 z_\alpha + \sigma_1^2 z_{1-\beta} + 0.85(\sigma_0^2 - \sigma_1^2)] = 0 \tag{34b}$$

from which

$$\nu^{1/2} = 2^{1/2}(\sigma_0^2 z_\alpha - \sigma_1^2 z_{1-\beta})/(\sigma_1^2 - \sigma_0^2) + O(\nu^{-1/2}). \tag{36}$$

Using a Lagrange multiplier, λ, the target function becomes

$$\max\{EV+\lambda Q\}. \tag{37}$$

Introducing from (34b) into (37) and differentiating with respect to $\nu^{1/2}$, α and β, where from z_3 we use

$$\partial z/\partial F = 0.4506/[F(1-F)],$$

we obtain

$$\partial(EV+\lambda Q)/\partial\nu^{1/2} = 0 = 2\lambda(\sigma_0^2-\sigma_1^2)\nu^{1/2} + 2^{1/2}\lambda(\sigma_0^2 z_\alpha - \sigma_1^2 z_{1-\beta}) - \partial C(n)/\partial\nu^{1/2},$$

which by (36) becomes approximately

$$\partial(EV+\lambda Q)/\partial\nu^{1/2} = 0 = \lambda(\sigma_0^2-\sigma_1^2)\nu^{1/2} - \partial C(n)/\partial\nu^{1/2}, \tag{38}$$

$$\partial(EV+\lambda Q)/\partial\alpha = 0 = 2.2193 p_0 D_0 \alpha(1-\alpha) - \lambda\sigma_0^2[(2\nu)^{1/2} - 1.9107], \tag{39}$$

$$\partial(EV+\lambda Q)/\partial\beta = 0 = 2.2193(1-p_0) D_1 \beta(1-\beta) + \lambda\sigma_1^2[(2\nu)^{1/2} + 1.9107]. \tag{40}$$

Isolating λ from (38), introducing it into (39) and (40) and then presenting for z_F from (14) into (34b) we finally obtain the optimality and feasibility conditions

$$\alpha^*(1-\alpha^*) = 0.6372[\sigma_0^2/(\sigma_0^2-\sigma_1^2)][1 - 1.3511/(\nu^*)^{1/2}][\partial C/\partial\nu^{1/2}]/(p_0 D_0), \tag{41}$$

$$\beta^*(1-\beta^*) = 0.6372[\sigma_1^2/(\sigma_0^2-\sigma_1^2)][1 + 1.3511/(\nu^*)^{1/2}][\partial C/\partial\nu^{1/2}]/[(1-p_0) D_1], \tag{42}$$

$$\{0.4506 \ln[\alpha^*/(1-\alpha^*)] - 0.2253\} - W\{-0.4506 \ln[\beta^*/(1-\beta^*)] + 0.2253\} + T = 0, \tag{43}$$

where

$$W = (\sigma_1^2/\sigma_0^2)[(2\nu^*)^{1/2} + 1.9107]/[(2\nu^*)^{1/2} - 1.9107],$$

$$T = (1-\sigma_1^2/\sigma_0^2)(\nu^* - 1.6238)/[(2\nu^*)^{1/2} - 1.9107].$$

Solving the quadratic (41) and (42) for α^* and β^* in terms of ν^*, and introducing into (43) we obtain an equation in ν^*, the roots of which may be easily identified.

Conditions for the existence of an optimal solution (besides the trivial nonnegativity ones related to the above quadratic equations) have yet to be established.

Example. Suppose a plant is suffering from large fluctuations in its daily output which management wishes to diminish by installing some electronic devices to control production processes.

Currently the standard deviation of daily production is σ_0 items and the expected standard deviation under improved control is σ_1 items.

The annual cost of achieving the desired improvement is estimated by C_0, and the expected net gain is G_1 per year.

A pretest of n days is designed and the annual cost of installing and running the control devices within the examination period is $a+bn$.

All costs are given in fixed prices for the expected depreciation period.

Management assigns a credibility of $100p_1\%$ to the supplier's claim concerning the expected STD.

Construct an optimal test.

Solution. We have $C(n) = a+bn$ so that $\partial C(n)/\partial\nu^{1/2} = 2b\nu^{1/2}$. Introducing into (41) and (42), and assuming that $\alpha^*, \beta^* < 1/2$ (α^*, β^* being the optimal values) we obtain thereof

$$\alpha^* = \tfrac{1}{2} - \{(1/4) - k_0[(\nu^*)^{1/2} - 1.3511]\}^{1/2}, \tag{41a}$$

$$\beta^* = \tfrac{1}{2} - \{(1/4) - k_1[(\nu^*)^{1/2} + 1.3511]\}^{1/2}, \tag{42a}$$

where

$$k_0 = 1.2744b[\sigma_0^2/(\sigma_0^2-\sigma_1^2)]/(p_0 D_0),$$

$$k_1 = 1.2744b[\sigma_1^2/(\sigma_0^2-\sigma_1^2)]/[(1-p_0)D_1].$$

Introducing the above α^* and β^* into (43) we obtain an equation in ν^*, the roots of which may be isolated by any standard routine installed in a hand calculator or a PC. Alternatively by trial and error we may put in (41a) and (42a) differing values of ν, calculate the respective α^* and β^*, and stop when these values result in a feasible solution (namely maintain relationship (34a) or its approximate equivalent (43)).

A numerical example. Suppose

$$\sigma_0 = 5000, \quad \sigma_1 = 2700, \quad p_0 = 0.3,$$

$$D_0 = 18{,}900, \quad D_1 = 485, \quad C(n) = 450 + 10n = 460 + 10\nu$$

(assume all costs are in 10^2\$). We obtain $k_0 = 0.00317$, $k_1 = 0.01545$.

Introducing into (41a) and (42a) and then into (43) we derive the root $\nu^* = 20.6$, thence $\alpha^* = 0.010$ and $\beta^* = 0.100$.

Table 8 gives values of (α, β, ν) in the vicinity of the above solution, and the associated $\widehat{EV}$ calculated from

$$\widehat{EV} = EV - E_p = -5670\alpha - 339.5\beta - (460 + 10\nu).$$

(Since E_p is a constant it need not be specified.) To make the transition from α to β and vice versa through the binding feasibility condition (34a), we used the highly accurate Wilson-Hilferty approximation

$$\chi_F^2 = \nu\{1 - 2/(9\nu) + z_F[2/(9\nu)]^{1/2}\}^3,$$

where z_F is the Fth fractile of a unit normal variate.

The results, as shown in Table 8, indicate that the approximate solution is practically the optimal one (the difference in the associated $\widehat{EV}$ between $\nu = 20$ and $\nu = 18$ is negligible and may partially be attributed to the use of the above W-H approximation).

TABLE 8

Values of (α, β, ν) in the vicinity of the approximate optimal solution, and the associated $\widehat{EV}$.

	ν						
	16	17	18	19	20	21	22
α (from (41a))	.0085	.0089	.0092	.0096	.0100	.0104	.0107
β (from (34a))	.2591	.2087	.1676	.1328	.1000	.0804	.0624
$\widehat{EV}$	−756.0	−751.1	−749.3	−749.5	−750.6	−756.3	−761.9
β (from (42a))	.0909	.0933	.0955	.0978	.1000	.1021	.1042
α (from (34a))	.0266	.0211	.0168	.0132	.0100	.0082	.0065
$\widehat{EV}$	−801.7	−781.3	−767.7	−758.2	−750.6	−751.4	−752.2

9. Conclusions. Simple general approximations for a random variable, based on linear transformations of standardized nonskewed random variables were derived. Whenever available the inverse distribution function of the latter were employed to approximate that of the former. Demonstration for five commonly used distributions shows that despite their simplicity the new approximations preserve an acceptable

degree of accuracy and may be employed to derive solutions for various statistical problems which are currently algebraically untractable. A full report on the accuracy and algebraic manipulability of the new approximations in comparison with existing ones for the above distributions and others is under preparation. The common four parameter density function which all of the above general approximations share requires some further characterization.

Appendix A: Derivation of the solution for $A_1 = A_2 = A$, $B_1 = -B_2 = B$. Assuming that the required approximation has the algebraic structure of (1), we look for solutions for A and B that yield approximate equality of the variance and the fourth cumulant, namely,

$$\text{(A.1)} \quad 2A^2M_2 - 4ABM_1 + B^2 = [(2M_2)^{1/2}A - M_1(2/M_2)^{1/2}B]^2 + [1 - 2M_1^2/M_2]B^2 = 1$$

and

$$\text{(A.2)} \qquad 2A^4M_4 - 8A^3BM_3 + 12A^2B^2M_2 - 8AB^3M_1 + B^4 - 3 = l_4.$$

Assume that

$$[1 - 2M_1^2/M_2]B^2 \ll 1.$$

Then this term in (A.1) may be neglected and we obtain therefrom:

$$\text{(A.3)} \qquad A = [1 + M_1(2/M_2)^{1/2}B]/(2M_2)^{1/2}.$$

Introducing it into (A.2) and neglecting terms which are $O(B^2)$ we have

$$\text{(A.4)} \qquad B = [l_4 + 3 - M_4/(2M_2^2)](M_2^3/8)^{1/2}/[M_1M_4/M_2 - M_3].$$

For the unit normal we have

$$M_1 = 1/(2\pi)^{1/2}, \quad M_2 = 0.5, \quad M_3 = (2/\pi)^{1/2}, \quad M_4 = 1.5$$

which yield

$$A = 1 + (2/\pi)^{1/2}B = 1 + (1/4)l_4,$$
$$B = (\pi/32)^{1/2}l_4.$$

The term in (A.1) was therefore properly neglected and we obtain equality of the variance and the fourth cumulant to $O(l_4^2)$. Similar arguements apply to z_2, the partial moments of which are developed in Appendix B.

Likewise introducing the partial moments of the logistic variate (these are given in Appendix B) we obtain A and B, given in § 5.

Appendix B: Derivation of the partial moments of approximations (13) and (14). Equation (13) is a linear transformation of the general term $G^{p-1}(1-G)^{q-1}$, the ith 'partial moment' of which is

$$\hat{M}_i = \int_{1/2}^{1} [G^{p-1}(1-G)^{q-1}]^i \, dG.$$

The complete beta function ratio, $B(p, q)$, is defined by

$$B(p, q) = \int_0^1 G^{p-1}(1-G)^{q-1} \, dG,$$

and the incomplete beta function ratio at x is defined by

$$I_x(p, q) = [1/B(p, q)] \int_0^x G^{p-1}(1-G)^{q-1}\, dG.$$

Since $\int_x^1 G^{p-1}(1-G)^{q-1}\, dG = [1 - I_x(p, q)]B(p, q)$, the above 'partial moment' may be calculated in terms of $B(p, q)$ and $I_x(p, q)$:

(B.1) $$\hat{M}_i = [1 - I_{1/2}(ip - i + 1, iq - i + 1)]B(ip - i + 1, iq - i + 1).$$

For (13) we have $p - 1 = -0.1193$, $q - 1 = 0.1193$, so that the 'partial moments' using (B.1) are

$$\hat{M}_1 = [1 - I_{1/2}(0.8807, 1.1193)]B(0.8807, 1.1193) = 0.42765,$$

$$\hat{M}_2 = [1 - I_{1/2}(0.7614, 1.2386)]B(0.7614, 1.2386) = 0.37166,$$

$$\hat{M}_3 = [1 - I_{1/2}(0.6421, 1.3579)]B(0.6421, 1.3579) = 0.32730,$$

$$\hat{M}_4 = [1 - I_{1/2}(0.5228, 1.4772)]B(0.5228, 1.4772) = 0.29146.$$

Introducing these into the expressions derived for the partial moments of (13), the values given in § 4.1 ensue.

To derive the partial moments of (14) we have the partial moments of the standard form logistic variate

$$y = \ln [G/(1-G)].$$

These are (see Johnson and Kotz (1970, Ch. 22))

$$M_1 = 0.6931, \quad M_2 = 1.6449, \quad M_3 = 5.4096, \quad M_4 = 22.7288.$$

Introducing these into the expressions derived for the partial moments of (14) the values given in § 4.1 result.

REFERENCES

B. J. R. BAILEY (1980), *Accurate normalizing transformations of a Student's t variate*, Appl. Statist., 29, pp. 304-306.

C. BEREDETTI (1956), *Sulla rappresentabilita di una distribuzione binomial mediante une distribuzione B e vice versa*, Metron, 18, pp. 121-131.

E. A. CORNISH AND R. A. FISHER (1937), *Moments and cumulants in the specification of distributions*, Rev. International Statistical Institute, 5, pp. 307-322.

N. L. JOHNSON AND S. KOTZ (1970), *Distributions in Statistics*, 3 Vol., Houghton-Mifflin, Boston.

H. SHORE (1982), *Simple approximations for the inverse cumulative function, the density function and the loss integral of the normal distribution.* Appl. Statist., 31, pp. 108-114.

J. WISHART (1947), *The cumulants of the z and of the logarithmic χ^2 and t distributions*, Biometrika, 34, pp. 170-178.

SIAM J. SCI. STAT. COMPUT.
Vol. 7, No. 1, January 1986

AN EXHAUSTIVE ANALYSIS OF MULTIPLICATIVE CONGRUENTIAL RANDOM NUMBER GENERATORS WITH MODULUS $2^{31}-1$*

GEORGE S. FISHMAN† AND LOUIS R. MOORE III‡

Abstract. This paper presents the results of an exhaustive search to find optimal full period multipliers for the multiplicative congruential random number generator with prime modulus $2^{31}-1$. Here a multiplier is said to be optimal if the distance between adjacent parallel hyperplanes on which k-tuples lie does not exceed the minimal achievable distance by more than 25 percent for $k=2,\cdots,6$. This criterion is considerably more stringent than prevailing standards of acceptability and leads to a total of only 414 multipliers among the more than 534 million candidate multipliers.

Section 1 reviews the basic properties of linear congruential generators and § 2 describes *worst case* performance measures. These include the maximal distance between adjacent parallel hyperplanes, the minimal number of parallel hyperplanes, the minimal distance between k-tuples, the lattice ratio and the discrepancy. Section 3 presents the five best multipliers and compares their performances with those of three commonly employed multipliers for all measures but the lattice test. Comparisons using packing measures in the space of k-tuples and in the dual space are also made. Section 4 presents the results of applying a battery of statistical tests to the best five to detect local departures from randomness. None were found. The Appendix contains a list of all optimal multipliers.

Key words. congruential generator, discrepancy, lattice test, random number generation, spectral test

Introduction. This paper presents the results of an exhaustive search to find *optimal* multipliers A for the multiplicative congruential random number generator $Z_i \equiv AZ_{i-1} (\text{mod } M)$ with prime modulus $M=2^{31}-1$. Since Marsaglia (1968) showed that k-tuples from this and the more general class of linear congruential generators lie on sets of parallel hyperplanes it has become common practice to evaluate multipliers in terms of their induced hyperplane structures. This study continues the practice and regards a multiplier as optimal if for $k=2,\cdots,6$ and each set of parallel hyperplanes the Euclidean distance between adjacent hyperplanes does not exceed the minimal achievable distance by more than 25 percent. The concept of using this distance measure to evaluate multipliers originated in the *spectral test* of Coveyou and MacPherson (1967) and has been used notably by Knuth (1981). However, the criterion of optimality defined here is considerably more stringent than the criteria that these writers proposed. In fact, among the more than 534 million full period multipliers A examined in this study, our research identified only 414 optimal multipliers.

First proposed by Lehmer (1951), the multiplicative congruential random number generator has come to be the most commonly employed mechanism for generating random numbers. Jannson (1966) collected the then known properties of these generators. Shortly thereafter Marsaglia (1968) showed that all such generators share a common theoretical flaw and Coveyou and MacPherson (1967), Beyer, Roof and Williamson (1971), Marsaglia (1972) and Smith (1971) proposed alternative procedures for rating the seriousness of this flaw for individual multipliers. Later Niederreiter (1976), (1977), (1978a, b) proposed a rating system based on the concept of *discrepancy*, a measure of error used in numerical integration. With regard to empirical evaluation, Fishman and Moore (1982) described a comprehensive battery of statistical tests and

* Received by the editors September 10, 1984, and in revised form December 5, 1984. This research was supported by the Office of Naval Research under contract N00014-26-C-0302.

† Curriculum in Operations Research and Systems Analysis, University of North Carolina, Chapel Hill, North Carolina 27514.

‡ Curriculum in Operations Research and Systems Analysis, and School of Business Administration, University of North Carolina, Chapel Hill, North Carolina 27514.

illustrated how they could be used to detect local departures from randomness in samples of moderate size taken from these generators.

Although the theoretical rating procedures have existed for some time, with the exception of Hoaglin (1976), Ahrens and Dieter (1977) and Knuth (1981), little use has been made of them. The present study, by its sheer exhaustiveness, removes this deficiency for generators with $M = 2^{31} - 1$. Section 1 reviews the basic properties of linear congruential generators. Then § 2 describes the *worst case performance measures* that have been proposed to rate generators in k dimensions. These include the maximal distance between adjacent parallel hyperplanes, the minimal number of parallel hyperplanes, the minimal distance between k-tuples, the lattice ratio and the discrepancy. These concepts are described in this study principally in terms of the space of k-tuples and, where appropriate, in terms of the dual lattice space. However, in order not to obfuscate central concepts the exposition relies on a minimal use of formal lattice theory.

Section 3 presents the five best multipliers and compares their performances with those of three commonly employed multipliers for all these measures but the lattice test. The Appendix contains a list of all optimal multipliers. Also, lattice packing measures are presented and again show the dominance of the five best over the three commonly used multipliers. Packing measures in the dual space are also computed. This last concept is identical with Knuth's figure of merit for evaluating generators. Our results indicate that with regard to this criterion the five best perform better than all 30 multipliers listed in Table 1 of Knuth (1981, pp. 102–103). Bounds on discrepancy are also computed and discussed.

Section 4 presents the results of a comprehensive empirical analysis of the local sampling properties of the best five, using the procedures in Fishman and Moore (1982). No evidence of departures from randomness was detected.

1. Linear congruential generators. A linear congruential generator produces a sequence of nonnegative integers

$$\{Z_0, Z_i \equiv AZ_{i-1} + C (\text{mod } M);\ i = 1, 2, \cdots\} \tag{1}$$

where the *modulus* M, and *multiplier* A are positive integers and the *seed* Z_0 and *constant* C are nonnegative integers. For purposes of conducting sampling experiments on a computer, the elements of the sequence Z are normalized to produce the sequence

$$U = \{U_i = Z_i / M;\ i = 1, 2, \cdots\}, \tag{2}$$

whose elements are treated as if they were sampled independently from the uniform distribution on the interval $[0, 1)$. The objective in assigning values to M, A, Z_0 and C is to make the errors incurred in this treatment of U tolerable ones. Here errors are principally of two types, one being the approximation of a continuous phenomenon on $(0, 1)$ by the discrete sequence U and the other being the distributional distortions in U induced by the use of the deterministic generator (1). In addition, computational considerations play a role in choosing M, A and C.

One property of the generator (1) is the period

$$T = \min \{k \geqq 1: Z_{n+k} = Z_n \text{ for all } n \geqq M\}. \tag{3}$$

The larger M is, the larger T can potentially be, and consequently the denser the points of U are in $[0, 1)$. The more dense these points are, the smaller the continuity error is.

Table 1 lists several types of linear congruential generators that are or have been in common use. Here A, C, Z_0 in the table guarantee maximal period for the corresponding modulus M. Note that types 1 and 2 give full periods whereas the remaining generators give only one fourth of the numbers between 1 and 2^β. Moreover, types 4a and 4b do not produce equidistributed sequences. Also, the use of $M=2^\beta$ enables one to replace division and multiplication by less time consuming shift and add operations. Although $M=2^\beta-1$ does not allow this substitution directly, a procedure due to Payne, Rabung and Bogyo (1969) enables one to retain part of this improved efficiency. Note that A is a primitive root of M if $A^{M-1}\equiv 1(\text{mod } M)$ and $A^Q \not\equiv 1(\text{mod } M)$ for $0<Q<M-1$.

TABLE 1
Linear congruential generators: $Z_i \equiv AZ_{i-1}+C(\text{mod } M)$.

Type	M	C	A	Z_0	Generated sequence is a permutation of	T
1	2^β	odd	1 mod 4	$\{0,1,\cdots,M-1\}$	$\{0,1,\cdots,M-1\}$	2^β
2	prime	0	primitive root of M	$\{1,\cdots,M-1\}$	$\{1,\cdots,M-1\}$	$M-1$
3a	2^β	0	5 (mod 8)	1 (mod 4)	$\{4j+1; j=0,1,\cdots,2^{\beta-2}-1\}$	$2^{\beta-2}$
3b	2^β	0	5 (mod 8)	3 (mod 4)	$\{4j+3; j=0,1,\cdots,2^{\beta-2}-1\}$	$2^{\beta-2}$
4a	2^β	0	3 (mod 8)	1 or 3 (mod 8)	$\{8j+1 \text{ and } 8j+3; j=0,1,\cdots,2^{\beta-3}-1\}$	$2^{\beta-2}$
4b	2^β	0	3 (mod 8)	5 or 7 (mod 8)	$\{8j+5 \text{ and } 8j+7; j=0,1,\cdots,2^{\beta-3}-1\}$	$2^{\beta-2}$

Source: Jannson (1966); A, C and Z_0 guarantee maximal period for the modulus $M=2^\beta$ with $\beta \geqq 3$.

Today only linear congruential generators of types 2 and 3 are commonly used. On IBM computers with a word size of 32 bits and $C=0$, the generator called SUPER-DUPER (Marsaglia (1972)) uses $M=2^{32}$, $A=69069$ and $Z_0=$ odd integer to give a period $T=2^{30}$. For generators of type 2 with prime number modulus $M=2^{31}-1$, APL (Katzan (1971)) uses $A=16807$, the SIMSCRIPT II programming language (Kiviat, Villanueva and Markowitz (1969)) uses $A=630360016$, SAS (1982) uses $A=397204094$ and the IMSL Library (1980) gives the user the choice of $A=16807$ or $A=397204094$. The resulting period is $T=2^{31}-2$.

2. Theoretical measures of performance. In practice, it is relatively common to use the *pseudorandom numbers* produced by (1) in groups or *k-tuples*. Consider the sequence of points

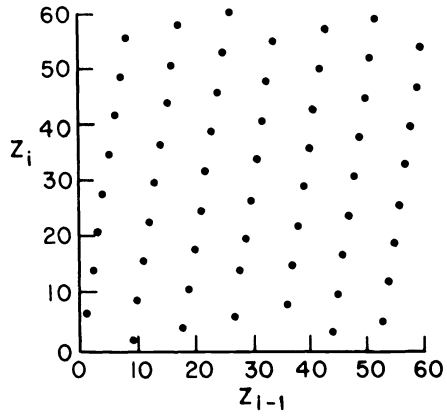
$$W_k=\{W_{i,k}=(Z_{i+1},\cdots,Z_{i+k}); i=1,2,\cdots\} \tag{4}$$

and the normalized sequence

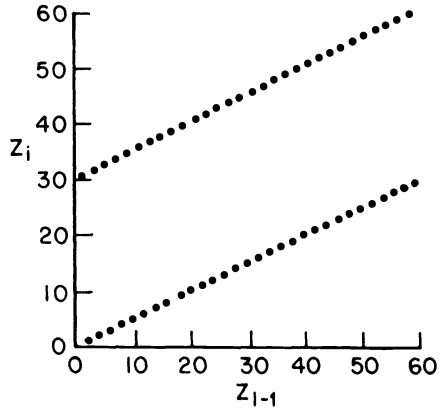
$$V_k=\{V_{i,k}=(Z_{i+1}/M,\cdots,Z_{i+k}/M); i=1,2,\cdots\}. \tag{5}$$

Ideally one wants the sequence of points V_k to be equidistributed in the k-dimensional unit hypercube for $k=2,3,\cdots$. However, the form of the generator (1) limits the extent to which one can achieve this ideal. For example, observe that an ideal generator of the integers $I=\{1,\cdots,M-1\}$ produces $(M-1)^k$ equidistributed points in k-dimensional space whereas a generator of type 2 produces only $M-1$ points in this space.

Although this constancy of the number of points is itself sobering, it is one of two importance issues. To illustrate the second issue, Fig. 1 shows a plot of 2-tuples



(a) $A=7$.



(b) $A=31$.

FIG. 1. $Z_i \equiv AZ_{i-1} (\text{mod } 61)$.

for the generators $Z_i \equiv 7Z_{i-1} (\text{mod } 61)$ and $Z_i \equiv 31Z_{i-1} (\text{mod } 61)$ where 61 is a prime number and 7 and 31 are primitive roots of 61. Although no one would seriously use either of these generators to produce random 2-tuples, a comparison of Figs. 1a and 1b arouses a concern that holds for more realistic generators as well. Notice that the distribution of points in Fig. 1b is considerably less uniform than the distribution in Fig. 1a. Since such differences in two and higher dimensions are attributable entirely to the choice of multiplier and since there are an enormous number of candidate multipliers, a deep analysis of k-tuples generated by (1) across all those multipliers is needed to assess the extent to which the resulting sequences V_k depart from the ideal of equidistributedness.

Several theoretical procedures have been proposed to make this assessment. They include:

(1) maximal distance between adjacent parallel hyperplanes (spectral test);
(2) minimal number of parallel hyperplanes;
(3) minimal distance between points;
(4) ratio of lengths of longest and shortest minimal basis vectors (lattice test);
(5) discrepancy.

Although diverse in what they measure, the procedures share a common unifying concept. All follow from recognizing that, with the exception of generators of types

4a and 4b, the k-tuples W_k can be regarded as points in a *regular lattice.* Moreover, generators of types 4a and 4b lead to k-tuples on two intermeshed regular lattices. Ahrens and Dieter (1977), Beyer, Roof and Williamson (1971) and Coveyou (1970) provide detailed descriptions of this relationship to lattice theory. To keep the focus of attention on the assessment of interest, the present paper presents only the features of lattice theory that are essential for describing these procedures. Also, unless otherwise noted our description applies for generators of type 2. Comparable analyses can be performed for each other type of generator.

2.1. Maximal distance between adjacent parallel hyperplanes. Observe that Z_i can be written in the form

$$Z_i \equiv Z_0 A^i (\text{mod } M) = Z_0 A^i - M \sum_{m=0}^{i-1} K_{i-m} A^m, \qquad i \geqq 1 \tag{6}$$

where $K_j = \lfloor AZ_{j-1}/M \rfloor$, $j = 1, 2, \cdots$. Now for $k \geqq 1$, $\mathbf{q} = (q_0, \cdots, q_{k-1})$ and y consider the k-dimensional hyperplane

$$H_k(\mathbf{q}, y) = \left\{ (x_0, \cdots, x_{k-1}) : \sum_{j=0}^{k-1} q_j x_j = y \right\}$$

and in particular the family of *parallel hyperplanes*

$$H_k(\mathbf{q}) = \{ H_k(\mathbf{q}, y) : y \equiv 0 \ (\text{mod } 1) \}. \tag{7}$$

Observe that the elements of V_k in (5) lie on hyperplanes in $H_k(\mathbf{q})$ in (7) if

(i) $q_0, \cdots, q_{k-1}$ integer

and

(ii) $q(A) = \sum_{j=0}^{k-1} q_j A^j \equiv 0 \pmod{M}$.

These restrictions are sufficient since for any $V_{i,k}$ in V_k the quantity

$$y_i = \frac{1}{M} \sum_{j=0}^{k-1} q_j Z_{i+j} = \frac{Z_0 A^i}{M} q(A) + k_i \tag{8}$$

where

$$k_i = -\sum_{j=1}^{k-1} q_j \sum_{m=0}^{i+j-1} K_{i+j-m} A^m. \tag{9}$$

Restriction (i) insures that k_i is an integer and restriction (ii) insures that $y_i - k_i$ is an integer. These restrictions hold throughout the remainder of this paper.

For the ensuing analysis it is convenient to extend V_k modulo one to the set

$$\begin{aligned} V_k^* = \{ & V^* = (v_0^*, \cdots, v_{k-1}^*) \text{ integer} \} \\ & \cup \{ V^* = (v_0^*, \cdots, v_{k-1}^*) \equiv V_{i,k} (\text{mod } 1); \ i = 1, \cdots, T \}. \end{aligned} \tag{10}$$

Since (i) and (ii) hold, the points in V_k^* also lie on hyperplanes in $H_k(\mathbf{q})$. Then the set of all hyperplanes containing at least one point of V_k^* is

$$H_k^*(\mathbf{q}) = \left\{ H_k(\mathbf{q}) : y = \sum_{j=0}^{k-1} q_j v_j^*, \ V^* \in V_k^* \right\}. \tag{11}$$

Moreover, one can index these hyperplanes by the set of integers

$$Y_k^*(\mathbf{q}) = \left\{ y^* = \sum_{j=0}^{k-1} q_j v_j^* \colon V^* \in V_k^* \right\}. \tag{12}$$

We now use these representations to show that for specified $k \geqq 1$ and $\mathbf{q}$ the k-tuples in V_k lie on a set of parallel hyperplanes for which the Euclidean distance between adjacent hyperplanes is fixed. The set of hyperplanes is $H_k^*(\mathbf{q})$ and for y and z in $Y_k^*(\mathbf{q})$ the Euclidean distance between $H_k(\mathbf{q}, y)$ and $H_k(\mathbf{q}, z)$ is $|y-z|/(\sum_{j=0}^{k-1} q_j^2)^{1/2}$. To prove the result, it suffices to show that the $Y_k^*(\mathbf{q})$ is composed of all integer multiples of some fixed constant $I_k(\mathbf{q})$, for then the Euclidean distance between adjacent hyperplanes in $H_k^*(\mathbf{q})$ is

$$d_k(\mathbf{q};\, A, M) = \frac{I_k(\mathbf{q})}{(\sum_{j=0}^{k-1} q_j^2)^{1/2}}. \tag{13}$$

By way of proof, note that if V and V' are two elements of V_k^* then $V'' = V' - V$ is also in V_k^* and therefore for y and z in $Y_k^*(\mathbf{q})$ one has $y - z$ in $Y_k^*(\mathbf{q})$. Also, for any integer j and point V^* in V_k^* the point $V' = jV^*$ is also in V_k^* so that $z = jy$, for $y \in Y_k^*(\mathbf{q})$, is also in $Y_k^*(\mathbf{q})$. Therefore, it follows that all elements of $Y_k^*(\mathbf{q})$ are multiples of

$$I_k(\mathbf{q}) = \min \{|y^*| > 0;\, y^* \in Y_k^*(\mathbf{q})\},$$

thus establishing (13). Without loss of generality we take

(iii) $I_k(\mathbf{q}) = 1$.

Since many different vectors $\mathbf{q}$ satisfy (i), (ii), and (iii) for a given multiplier A and induce families of parallel hyperplanes, additional criteria are needed to enable one to characterize the extent of equidistributedness of the k-tuples V_k in (5) in the k-dimensional unit hypercube for each possible multiplier. One such criterion is the *maximal distance* between adjacent parallel hyperplanes which is a *worst case measure* for a particular multiplier A. It is

$$d_k^*(A, M) = \max_{q_0, \cdots, q_{k-1}} \left[\left(\sum_{j=0}^{k-1} q_j^2 \right) \right]^{-1/2} \tag{14}$$

subject to restrictions (i), (ii) and (iii). In particular, note that the constraint (iii) eliminates the numerator of (13) from the maximization (14).

When using (14) to compare k-tuple performance for several multipliers for a type of generator, one prefers the multiplier that gives the minimal maximal distance since this implies smaller *empty regions* in the k-dimensional unit hypercube for this multiplier than for the other multipliers. However, there is a limit to how small this maximal distance can be; in particular, it is known that (Cassels (1959, p. 332))

$$M^{1/k} d_k^*(A, M) \geqq \gamma_k = \begin{cases} (3/4)^{1/4}, & k = 2, \\ 2^{-1/6}, & k = 3, \\ 2^{-1/4}, & k = 4, \\ 2^{-3/10}, & k = 5, \\ (3/64)^{1/12}, & k = 6. \end{cases} \tag{15}$$

To illustrate the significance of these bounds, note that with the modulus $M = 2^{31}-1$ one has

$$d_k^*(A, 2^{31}-1) \geqq \begin{cases} .2008\times 10^{-4}, & k=2, \\ .6905\times 10^{-3}, & k=3, \\ .3906\times 10^{-2}, & k=4, \\ .1105\times 10^{-1}, & k=5, \\ .2157\times 10^{-1}, & k=6, \end{cases}$$

indicating the relative coarseness of the grid of points in as few as four dimensions.

Using multivariable Fourier analysis, Coveyou and MacPherson (1967) advocated using the minimized "wave number"

$$\left(\sum_{j=0}^{k-1} q_j^2\right)^{1/2}$$

(s.t. $q_0, \cdots, q_{k-1}$ integer and $q(A) \equiv 0 (\text{mod } M)$) to determine the relative desirabilities of alternative multipliers; hence the name *spectral test.* Shortly thereafter, it became apparent (Coveyou (1970), Beyer, Roof and Williamson (1971)) that one could perform equivalent studies using (14) by viewing the k-tuples as being arranged on parallel hyperplanes and exploiting the mathematical properties of the so-induced *lattice structure.* In fact, it turns out that the physical interpretation of results can be more easily understood in the space of W_k whereas the computational procedures are more easily understood by working in the *dual space* of $\mathbf{q}$. We return to this issue in § 3.

2.2. Minimal number of parallel hyperplanes. A second measure of equidistributedness, suggested by Marsaglia (1968), is the *number of parallel hyperplanes* $N_k(q_0, \cdots, q_{k-1}; A, M)$ on which all the k-tuples lie. If this number is small for a particular multiplier A, then this is an indication that there exist large regions in the k-dimensional unit hypercube that contain no k-tuples.

Observe that with restriction (iii) gives the upper bound

$$N_k(q_0, \cdots, q_{k-1}, A, M) \leqq \sum_{j=0}^{k-1} |q_j|. \tag{16}$$

Using the development in Dieter (1975), one also observes that

$$-\sum_{j=0}^{k-1} (q_j)^- < y_i < \sum_{j=0}^{k-1} (q_j)^+, \qquad i=1, \cdots, T$$

where y_i is defined in (8) and

$$x^- = \begin{cases} 0 & \text{if } x \geqq 0, \\ -x & \text{if } x<0, \end{cases} \qquad x^+ = \begin{cases} x & \text{if } x \geqq 0, \\ 0 & \text{if } x<0. \end{cases}$$

Because of the restrictions (i) through (iii) the number of distinct y_i is precisely the maximal number of parallel hyperplanes that pass through the k-dimensional unit hypercube. This number is

$$N_k(q_0, \cdots, q_{k-1}; A, M) = \sum_{j=0}^{k-1} (q_j)^- + \sum_{j=0}^{k-1} (q_j)^+ = \sum_{j=0}^{k-1} |q_j| - 1. \tag{17}$$

Note that all these hyperplanes may not be occupied.

As before, there exist many vectors $\mathbf{q}$ that satisfy restrictions (i), (ii), and (iii). A worst case measure here is

$$N_k^*(A, M) = \min_{\mathbf{q}} N_k'(q_0, \cdots, q_{k-1}; A, M) \tag{18}$$

subject to the restrictions. When using this criterion to choose among several multipliers, one prefers the one that gives the maximal minimal $N_k(q_0, \cdots, q_{k-1}; A, M)$. As in the case of distance between hyperplanes, an upper bound exists on $N_k^*(A, M)$, namely (Marsaglia (1968))

$$N_k^*(A, M) \leqq (k!M)^{1/k}, \qquad k = 1, 2, \cdots.$$

In particular, for $M = 2^{31} - 1$ the bounds are

$$N_k^*(A, M) \leqq \begin{cases} 65536, & k = 2, \\ 2344, & k = 3, \\ 476, & k = 4, \\ 191, & k = 5, \\ 107, & k = 6. \end{cases}$$

Again, these bounds are limiting, and encourage one to search for multipliers that can come close to the bounds.

Knuth (1981, p. 92) points out that the ordering of several multipliers $A_1, \cdots, A_p$ according to the maximal distance measure $d_k^*(A, M)$ may differ from the ordering established by the minimal number of parallel hyperplanes measure $N_k^*(A, M)$. In particular, he notes that $N_k^*(A, M)$ "is biased by how nearly the slope of the lines or hyperplanes matches the coordinate axes of the cube." That is, $N_k^*(A, M)$ may be relatively large when $d_k^*(A, M)$ is also relatively large. Since in this case one inclines to discount the multiplier because of sparseness indicated by $d_k^*(A, M)$, there is some justification for valuing $d_k^*(A, M)$ more highly than $N_k^*(A, M)$ as a measure of performance. Section 3 takes this into consideration when searching for optimal multipliers.

Although $d_k^*(A, M)$ provides a more definite evaluation of a multiplier than $N_k^*(A, M)$ does, the latter quantity has at least one readily appealing attribute that justifies its consideration. We illustrate this feature for the type 3 generator with $A = 65539$ and $M = 2^{31}$. This generator is known as RANDU and was a standard feature of the IBM Scientific Subroutine Library on 360/370 series computers for many years. Observe that $65539 = 2^{16} + 3$ so that

$$\begin{aligned} Z_{i+1} &\equiv (2^{16} + 3)Z_i \pmod{2^{31}}, \\ Z_{i+2} &\equiv (6 \times 2^{16} + 9)Z_i \pmod{2^{31}}, \\ Z_{i+2} &\equiv 6Z_{i+1} - 9Z_i \pmod{2^{31}}, \\ Z_{i+2} - 6Z_{i+1} + 9Z_i &\equiv 0 \pmod{2^{31}}. \end{aligned}$$

Moreover,

$$U_{i+2} - 6U_{i+1} + 9U_i \equiv 0 \pmod{1}$$

indicating that $N_3^*(65539, 2^{31}) \leqq 16$, a devastating indictment of RANDU in three dimensions. Therefore, the valuable feature of $N_k^*(A, M)$ is that it can on occasion identify a poor multiplier with relatively little computational effort.

2.3. Distance between points. Smith (1971) has suggested an alternative measure of equidistributedness based on the minimal distance between k-tuples

$$c_k^*(A, M) = \min_{\substack{1 \leqq i,m \leqq T \\ i \neq m}} \frac{1}{M}\left[\sum_{j=0}^{k-1} (Z_{i+j} - Z_{m+j})^2\right]^{1/2}. \tag{19}$$

Since the total number of points is fixed at T, the smaller $c_k^*(A, M)$ is for a given A, the more clustered are points in the k-dimensional unit hypercube. Therefore, when comparing several multipliers in k dimensions one prefers the one that gives the maximal $c_k^*(A, M)$.

Whereas $d_k^*(A, M)$ measures distance between adjacent parallel hyperplanes, $c_k^*(A, M)$ measures distance between nearest points. An alternative, but equivalent interpretation is to view $1/c_k^*(A, M)$ as the maximal distance between adjacent parallel hyperplanes in the dual space of $\mathbf{q}$. The observation enables one to establish the upper bouds for $c_k^*(A, M)$ $k = 2, 3, \cdots$ (Cassels (1959, p. 332)):

$$c_k^*(A, M) \leqq 1/\gamma_k M^{1/k} \tag{20}$$

where γ_k is defined in (15). This duality relationship also facilitates the computation of $c_k^*(a, M)$ using the algorithm in Dieter [1975].

2.4. Discrepancy. The concept of discrepancy originated in the study of how well equidistributed sequences perform in sampling procedures designed to approximate the volumes of regions in the k-dimensional unit hypercube and in numerical integration. Having recognized the relationship between this problem and that of measuring the performance of a random number generator, Niederreiter (1977) adapted the discrepancy measure to this latter problem and gave bounds for it.

Consider the sequence of k-tuples $\{W_{i,k}; i = 1, \cdots, T\}$ defined in (4). For $N = 1, \cdots, T$ discrepancy in k dimensions for a multiplier A and a modulus M is defined as

$$D_N^{(k)}(A, M) = \max_R \left|\frac{\text{number of } W_{1,k}, \cdots, W_{N,k} \text{ in } R}{N} - \frac{\text{volume of } R}{M^k}\right| \tag{21}$$

where R ranges over all sets of points of the form $R = \{(w_1, \cdots, w_k) \mid \alpha_1 \leqq w_1 < \beta_1, \cdots, \alpha_k \leqq w_k < \beta_k\}$. Here α_j and β_j are integers in the range $0 \leqq \alpha_j < \beta_j < M$ for $1 \leqq j \leqq k$ so that the volume of R is

$$\prod_{j=1}^{k} (\beta_j - \alpha_j).$$

Niederreiter (1977), (1978a) gave upper and lower bounds for $D_N^{(k)}(A, M)$ for generators of types 1, 2 and 3 for arbitrary $N \leqq T$. In particular, the upper bound for generators of type 2 is

$$D_N^{(k)}(A, M) \leqq \frac{k}{M} + \frac{\min(N, (M-N)^{1/2})}{N} \sum_{\mathbf{q} (\text{mod } m)}^{*} \frac{1}{r(\mathbf{q}, M)}$$

$$+ \frac{\max(0, N-(M-N)^{1/2})}{N} \sum_{\substack{\mathbf{q} (\text{mod } M) \\ q(A) \equiv 0 (\text{mod } M)}}^{*} \frac{1}{r(\mathbf{q}, M)}$$

where the asterisk denotes exclusion of $q_0 = \cdots = q_{k-1} = 0$,

$$r(\mathbf{q}, M) = \sum_{j=0}^{k-1} r(q_j, M),$$

$$r(q, M) = \begin{cases} 1 & \text{if } q \equiv 0 \pmod{M}, \\ M \sin \pi \|q/M\| & \text{if } q \not\equiv 0 \pmod{M}, \end{cases}$$

and

$$\|t\| = \min(t, 1-t).$$

Note that this bound holds for any local sample of N successive k-tuples from the generator as well as for a global evaluation of performance when $N = T$.

At present there exists no algorithm, other than total enumeration, for computing this upper bound and this situation is likely to remain so. However, the form of the bound enables one to establish a valuable relationship between the spectral test and discrepancy. Note that $\sin \pi \|q/M\| > 0$ in the bound and that the number of such terms is a function of $q(A)$. Recall that the quantity

$$\left(\sum_{j=0}^{k-1} q_j^2 \right)^{1/2}$$

is minimized, subject to $q(A) \equiv 0 \pmod{M}$, to find the maximal distance between adjacent parallel hyperplanes. If this minimized quantity turns out to be small for a multiplier A, then the congruence occurs frequently for $0 \leqq |q_{j-1}| < M$ $j = 1, \cdots, k$. This clearly adds positive terms to the second summation and therefore the upper bound is large. If for an alternative multiplier A' the minimized quantity turns out large, the congruence holds less frequently and the upper bound is smaller than in the previous case. Thus the results for the spectral test convey useful information about the bounds on discrepancy.

For generators of types 1 and 2, Niederreiter (1976, 1978a) also gave the lower bound

$$D_T^{(k)}(A, M) \geqq \begin{cases} 1/k^k \rho^{(k)}(A, M) & \text{for } 2 \leqq k \leqq 6, \\ \pi/2(2\pi+1)^k \rho^{(k)}(A, M) & \text{for } k \geqq 7 \end{cases} \tag{22}$$

and the upper bound

$$\begin{aligned} D_T^{(k)}(A, M) < \frac{k}{M} + \min\left(1, \frac{\sqrt{M-T}}{T}\right)\left(\frac{2}{\pi}\log M + \frac{7}{5}\right)^k \\ + (\log 2)^{1-k}((2\log M)^k + 4(2\log M)^{k-1})/2\rho^{(k)}(A, M) \\ + 2^k(2^{k-2}-1)\binom{J+k-2}{k-1}\Big/\rho^{(k)}(A, M) \end{aligned} \tag{23}$$

where

$$\rho^{(k)}(A, M) = \min_{\substack{\mathbf{q} \pmod M \\ \mathbf{q} \not\equiv (0, \cdots, 0) \\ q(A) \equiv 0 \pmod M}} \left[\prod_{j=0}^{k-1} \max(1, |q_j|) \right] \tag{24}$$

and

$$J = (\log M)/\log 2.$$

Comparable results exist for generators of type 3.

With the exception of $k=2$ no algorithm exists for computing $\rho^{(k)}(A, M)$. Ahrens and Dieter (1977, Thm. 5.17) gave the stronger lower bound

$$D_T^{(k)}(A, M) \geqq 1 \Big/ \left[\min_{\substack{\mathbf{q} \neq (0,\cdots,0) \\ q(A) \equiv 0 (\mathrm{mod}\ M)}} \left(\lambda_m \prod_{i=0}^{k-1} |q_i| \right) \right] \tag{25}$$

where m denotes the number of nonzero q_i,

$$\lambda_m = \begin{cases} m^m & \text{if } m = 2 \text{ or } 3, \\ m^m/(m-1)^m H_m & \text{if } m \geqq 4 \end{cases} \tag{26}$$

and

$$H_m = \left[\sum_{j=0}^{\lfloor m/2 \rfloor + 1} (-1)^j \binom{m}{j} (\lfloor m/2 \rfloor + 1 - j)^{m-1}/(m-1)! \right]^m.$$

For $k=2$ Borosh and Niederreiter (1983) showed that

$$\rho^{(2)}(A, M) = \min_{0 \leqq |q_1| \leqq M/2} (|q_1| \cdot |tM - q_1 A|) \tag{27}$$

for some t satisfying $q_0 = tM - q_1 A$. This result makes the bounds in (22) and (23) operative for $k=2$.

Niederreiter (1977), (1978b) provided additional bounds for $k=2$. For type 2 generators

$$D_T^{(2)}(A, M) \leqq \left(2 + \sum_{i=1}^{p} a_i \right) \Big/ T \tag{28}$$

and

$$D_T^{(2)}(A, M) \leqq [2 + C(K) \log T]/T \tag{29}$$

where $a_1, \cdots, a_p$ are the partial quotients in the continued fraction expansion of A/M, $K = \max(a_1, \cdots, a_p)$ and $C(K) = 2/\log 2$ for $1 \leqq K \leqq 3$ and $C(K) = (K+1)/\log(K+1)$ for $K \geqq 4$. Expressions (28) and (29) also hold for type 3 generators with $2/T$ replaced by $1/T$ and with $a_1, \cdots, a_p$ being the partial quotients of $A/2^{\beta-2}$. Earlier, Dieter (1971) derived closely related results based on continued fractions to nearest integers rather than regular continued fractions.

Borosh and Niederreiter (1983, Table 2) have carried out a systematic search for multipliers of type 3 and type 4 for $k=2$. In particular, they gave maximal period multipliers with $K \leqq 3$ for $\beta = 6, 7, \cdots, 35$ for each type.

2.5. Lattice test. Beyer, Roof and Williamson (1971) and Marsaglia (1972) proposed an alternative figure of merit, for evaluating alternative multipliers, based on the concept of *squareness*. We use Fig. 1 to illustrate this concept. Clearly one can construct a vast number of parallelograms of varying areas that include no interior points. The presumption of the lattice test is that one prefers multipliers that produce parallelograms of minimal area whose sides are close, if not equal, in length; hence, the notion of squareness, where angles are neglected.

Now the minimal volume of a k-dimensional parallelepiped generated by k-tuples from (1) subject to (ii) is M^{k-1}. In evaluating a particular multiplier, the objective of the *lattice test* is to find the basis vectors $\boldsymbol{\alpha}_1, \cdots, \boldsymbol{\alpha}_k$ that come closest in k dimensions to achieving this squareness for parallelepipeds of volume M^{k-1}. To measure the extent of the departure from equidistributedness in k dimensions Beyer, et al. and Marsaglia

recommended the quantity

$$R_k(A) = \frac{\max_{1 \leqq i \leqq k} |\boldsymbol{\alpha}_i|}{\min_{1 \leqq i \leqq k} |\boldsymbol{\alpha}_i|}; \tag{30}$$

that is, the ratio of the lengths of the longest and the shortest basis vectors. Clearly $R_k(A) \geqq 1$ and presumably one prefers multipliers for which $R_k(A)$ is close to unity.

It is worthwhile noting that the basis vectors $\boldsymbol{\alpha}_1, \cdots, \boldsymbol{\alpha}_k$ play an implicit role in the previously mentioned tests as well. For example, one can show that for $k=2$ the maximal distance between parallel hyperplanes is

$$d_2^*(A, M)(|\boldsymbol{\alpha}_2|^2 - |\boldsymbol{\alpha}_1 \cdot \boldsymbol{\alpha}_2|^2/|\boldsymbol{\alpha}_1|^2)^{1/2}$$

where we take $\boldsymbol{\alpha}_2$ to be the longer vector.

Although the figure of merit in (30) has intuitive appeal, there is no universal agreement about its usefulness in identifying good multipliers. Marsaglia (1972, p. 275) suggested a generator of type 3 called SUPER-DUPER with $M = 2^{32}$ and $A = 69069$. It has $R_2(A) = 1.06$, $R_3(A) = 1.29$, $R_4(A) = 1.30$ and $R_5(A) = 1.25$; an appealing generator as evaluated by the lattice test. For this generator Niederreiter (1978, pp. 1027-1028) showed that $\rho^{(2)}(A, M) \leqq 69069$ so that (22) gives $D_T^{(2)}(A, M) \geqq 1/(4 \times 69069) = .3620 \times 10^{-5}$. But Borosh and Niederreiter gave a multiplier $A = 3039177861$ for $M = 2^{32}$ with $\rho^{(2)}(A, M) = .2517M$ and $\sum_{i=1}^{P} a_i = 51$ for which (28) based on $M = 2^{32}$ gives $D_T^{(2)}(A, M) \leqq (1+51)/2^{30} = .4843 \times 10^{-7}$. This result illustrates that although SUPER-DUPER has the appealing figure of merit $R_2(69069) = 1.06$, there exist multipliers with $A \equiv 5 (\text{mod } 2^{32})$ that dominate it by a substantial margin in $k=2$ dimensions with regard to discrepancy.

3. Analysis. This section presents results of an investigation based on the evaluation of $\{d_k^*(A, M); k = 2, \cdots, 6\}$ for all multipliers A that are primitive roots of $M = 2^{31} - 1$, using an algorithm of Dieter (1975), as described in Knuth (1981, algorithm S). Hardy and Wright (1960) show that the number of primitive roots for M prime is $\phi(M-1)$ where

$$\phi(M-1) = \text{number of integers not exceeding and relatively prime to } M-1.$$

This quantity is called the Euler totient function. Since $\phi(M-1)/(M-1) \doteq .249$ for $M = 2^{31}-1$ (Ahrens and Dieter (1977, p. 7, 6)) one has $\phi(2^{31}-2) \doteq 534723428$, a not inconsequential number.

To find the primitive roots, one notes that if B is the smallest primitive root of the prime modulus M, then every primitive root has the form

$$A \equiv B^I (\text{mod } M)$$

where I is an integer whose largest common factor with $M-1$ is unity. Since one also can show that for every such I there exists a pair of multipliers $B^I (\text{mod } M)$ and $B^{M-1-I} (\text{mod } M)$ with identical lattice structures, it suffices to investigate only half of all the primitive roots. In the present case 7 is the smallest primitive root of $2^{31}-1$ so that only 267361714 multipliers require examination. Note that the multiplier with exponent $M-1-I$ produces the same sequence as the multiplier with exponent I does, but in reverse order.

Clearly one needs to adopt a screening procedure to identify and collect those multipliers that "perform well". For present purposes, the multipliers of most interest are those that "perform well" in $k = 2, \cdots, 6$ dimensions relative to the constraints

imposed on all lattices in these dimensions. Consider the ratios

$$S_{1,k}(A, M) = \gamma_k / d_k^*(A, M) M^{1/k}, \qquad k = 2, \cdots, 6.$$

As seen from (15), $0 < S_{1,k}(A, M) \leqq 1$. Now the closer $S_{1,1}(A, M), \cdots, S_{1,6}(A, M)$ are to unity the better the performance is of this multiplier with regard to the achievable bounds in $2, \cdots, 6$ dimensions. Therefore, one way to perform the screening is to identify all multipliers for which

$$\min_{2 \leqq k \leqq 6} S_{1,k}(A, M) \geqq S, \qquad 0 < S < 1$$

for specified S.

Initially we chose $S = .75$. Since preliminary computations indicated that there were an unmanageable number of multipliers that satisfied this criterion, we changed S to .80. This resulted in a total of 207 optimal multipliers, as listed in the Appendix. Recall that there are actually twice this number of optimal multipliers. The abrupt reduction in the number of optimal multipliers when shifting from $S = .75$ to $S = .8$ is itself notable. Also note that any multiplier for which $S_{1,k}(A, M) > .8$ for $k = 2, \cdots, 6$ guarantees that for each k the distance between adjacent hyperplanes does not exceed the minimal achievable distance by more than 25 percent.

For each selected multiplier and $k = 2, \cdots, 6$ we also computed the ratios

$$S_{2,k}(A, M) = N_k^*(A, M)/(k! M)^k$$

and

$$S_{3,k}(A, M) = c_k^*(A, M) \gamma_k M^{1/k},$$

again using Dieter's algorithm.

Table 2 presents these ratios for the multipliers with the five largest min $S_{1,k}(A, M)$. It also presents results for $A = 16807$ which is in APL and IMSL, for $A = 397204094$ which is in IMSL and SAS, for $A = 630360016$ which is the SIMSCRIPT II multiplier, and for $A = 7$. This last multiplier illustrates the contrasts that are possible in performance.

Table 2 allows one to make several notable observations:

(a) The first five multipliers perform considerably better than the remaining multipliers in the table with regard to the screening measures $\{S_{1,k}(A, M)\}$ and with regard to $\{S_{2,k}(A, M)\}$ and $\{S_{3,k}(A, M)\}$.

(b) For each of these five multipliers $S_{1,2}(A, M), \cdots, S_{1,6}(A, M)$ are remarkably close.

(c) The measures $S_{3,2}(A, M), \cdots, S_{3,6}(A, M)$ are also remarkably close and behave essentially as $S_{1,2}(A, M), \cdots, S_{1,6}(A, M)$. As expected, $S_{1,2}(A, M) = S_{3,2}(A, M)$.

(d) $S_{2,2}(A, M), \cdots, S_{2,6}(A, M)$ show considerably more variation; no doubt a reflection of the suboptimality of these multipliers with regard to this criterion.

We now turn to another method of evaluating performance which derives from the concept of *packing* a lattice with spheres (see Cassels (1959)). Recall that $c_k^*(A, M)$ is the distance between nearest points in the unit hypercube of k-tuples. Then the volume of a sphere with this diameter is

$$L_k(A, M) = \frac{\pi^{k/2}(c_k^*(A, M)/2)^k}{\Gamma(k/2+1)}$$

where $\Gamma(\cdot)$ denotes the gamma function. Suppose one packs the lattice with such

TABLE 2

Performance measures for selected multipliers in $Z_i \equiv AZ_{i-1} (\text{mod } M)$[a].

$(M = 2^{31} - 1)$

Multiplier A		Dimension (k) 2	3	4	5	6
742938285	S_1	.8673	.8607	.8627	.8320	.8342
	S_2	.8362	.6613	.6618	.6021	.6075
	S_3	.8673	.8751	.8507	.7838	.7983
950706376	S_1	.8574	.8985	.8692	.8337	.8274
	S_2	.9211	.8183	.6555	.6806	.6822
	S_3	.8574	.9093	.8412	.7565	.7646
1226874159	S_1	.8411	.8787	.8255	.8378	.8441
	S_2	.8273	.7240	.7815	.6492	.6822
	S_3	.8411	.8877	.8468	.7107	.7743
62089911	S_1	.8930	.8903	.8575	.8630	.8249
	S_2	.7169	.7537	.7430	.7153	.6603
	S_3	.8930	.8286	.7712	.8150	.7385
1343714438	S_1	.8237	.8324	.8245	.8262	.8255
	S_2	.8676	.6404	.6492	.6702	.7103
	S_3	.8237	.7785	.7906	.7874	.7747
16807	S_1	.3375	.4412	.5752	.7361	.6454
	S_2	.2565	.3264	.5714	.6754	.5888
	S_3	.3375	.5404	.6162	.6187	.5889
397204094	S_1	.5564	.5748	.6674	.7678	.5947
	S_2	.5966	.5038	.6239	.6597	.4206
	S_3	.5564	.5543	.7302	.7849	.6417
630360016	S_1	.8212	.4317	.7832	.8021	.5700
	S_2	.8823	.4373	.6534	.7173	.5047
	S_3	.8212	.6354	.6441	.7983	.5510
7	1000 S_1	.1420	4.882	27.62	78.13	152.6
	1000 S_2	.1221	3.413	16.81	41.19	74.77
	1000 S_3	.1420	.02650	.02921	.06746	.2201

[a] $S_1 = \gamma_k / d_k^*(A, M) M^{1/k}$, $S_2 = N_k^*(A, M)/(k! M)^{1/k}$ and $S_3 = c_k^*(A, M) \gamma_k M^{1/k}$.

spheres centered on each of the $M-1$ points V_k in (5) and at the origin. Note that these spheres merely touch and that since there are only M k-tuples, the proportion of the unit hypercube packed with these spheres is $ML_k(A, M)$.

Let

$$\omega_k(A, M) = 2^k ML_k(A, M).$$

Using the lattice packing constants in (15) and (20) one has

$$\omega_k(A, M) \leqq \begin{cases} 3.63, & k = 2, \\ 5.92, & k = 3, \\ 9.87, & k = 4, \\ 14.89, & k = 5, \\ 23.87, & k = 6. \end{cases}$$

Table 3 lists $\omega_k(A, M)$ for the five best and the three other commonly employed multipliers. The benefits of the five multipliers is again apparent since their packings are considerably better across dimensions than those for the more commonly used multipliers.

TABLE 3
Packing measures in the sample space.

$$\omega_k(A, M) = \pi^{k/2} M[c_k^*(A, M)]^k / \Gamma(k/2+1)$$

$$(M = 2^{31} - 1)$$

Multiplier A	Dimension (k) 2	3	4	5	6
742938285	2.73	3.97	5.17	4.40	6.17
950706376	2.67	4.45	4.94	3.69	4.77
1226874159	2.57	4.14	5.07	2.70	5.14
62089911	2.89	3.37	5.17	5.36	3.87
1343714438	2.46	2.80	3.86	4.51	5.16
16807	.41	.93	.00	1.35	1.00
397204094	1.12	1.01	2.80	4.44	1.67
630360016	2.45	1.52	1.70	4.83	.67
Upper bound	3.63	5.92	9.87	14.89	23.87

Knuth (1981, p. 102) has also used this concept of packing to rate multipliers. However, his approach relates to packing spheres in the dual space of $q_0/M, \cdots, q_{k-1}/M$. This is done by noting that in addition to $d_k^*(A, M)$ being the maximal distance between neighboring parallel hyperplanes in the space of V_k, the quantity $1/Md_k^*(A, M)$ is the minimal distance between points in the dual space of $q_0/M, \cdots, q_{k-1}/M$. Therefore, the volume of a sphere with radius $1/2d_k^*(A, M)$ in the dual space is

$$W_k(A, M) = \frac{\pi^{k/2}}{\Gamma(k/2+1)[2Md_k^*(A, M)]^k}.$$

Now observe that restrictions (i) and (ii) determine that the hypercube $[-1, 1)^k$ contains exactly $2^k M^{k-1}$ k-dimensional points $\mathbf{q}/M$. In particular, the exponent $k-1$ instead of k on M is due to restriction (ii). Therefore, the volume of this hypercube packed

TABLE 4
Packing measures in the dual space.

$$\mu_k(A, M) = \frac{\pi^{k/2}}{\Gamma(k/2+1)M[d_k^*(A, M)]^k}$$

$$(M = 2^{31} - 1)$$

Multiplier A	Dimension (k) 2	3	4	5	6
742938285	2.73	3.78	5.47	5.94	8.04
950706376	2.67	4.30	5.63	6.00	7.66
1225874159	2.57	4.02	4.58	6.15	8.63
62089911	2.14	4.34	4.23	4.77	7.99
1343714438	2.46	3.42	4.56	5.73	7.55
16807	.41	.51	1.08	3.22	1.73
397204094	1.12	1.13	1.96	3.97	1.06
630360016	2.45	.48	3.71	4.94	.82
Upper bound	3.63	5.92	9.87	14.89	23.87

with spheres is

$$\mu_k(A, M) = 2^k M^{k-1} W_k(A, M) = \frac{\pi^{k/2}}{\Gamma(k/2+1) M[d_k^*(A, M)]^k},$$

which is the measure of packing in the dual space. This quantity is identical with the figure of merit suggested by Knuth (1981, p. 101). Note that because of the lattice structure in the dual space this result is invariant when the hypercube is translated by a vector of integers.

Table 4 lists $\mu_k(A, M)$ for the multipliers of interest. Again note the better performance of the top five. Knuth remarks that one might say that any multiplier for which $\mu_k(A, M) \geqq .1$, $k = 2, \cdots, 6$ passes the spectral test and any multiplier for which $\mu_k(A, M) \geqq 1$ $k = 2, \cdots, 6$ passes the test with flying colors. By this standard the top five multipliers are untouchable. In fact, since $S_{1,k}(A, M) \geqq .8$ $k = 2, \cdots, 6$ for all multipliers in the Appendix, those multipliers have

$$\mu_k(A, M) \geqq \begin{cases} 2.32, & k = 2, \\ 3.03, & k = 3, \\ 4.04, & k = 4, \\ 4.88, & k = 5, \\ 6.26, & k = 6, \end{cases}$$

indicating that all meet the Knuth criterion and dominate all multipliers listed in Knuth (1981, pp. 102-103).

Table 5 presents bounds on discrepancy computed from (25) and (28) and reveals several interesting results. First, note that the intervals for $k = 2$ can in no way be

TABLE 5
Bounds on discrepancy.

Multiplier A		Dimension (k) 2	3	4	5	6
742938285	Lower[a]	.1492	.5970	42.89	42.89	42.89
	Upper[b]	3.446				
950706376	Lower	.2680	1.072	9.607	10.08	10.08
	Upper	3.725				
1226874159	Lower	1.967	7.869	7.869	7.869	14.86
	Upper	10.52				
62089911	Lower	.4236	1.694	1.694	1.694	4.328
	Upper	6.333				
1343714438	Lower	.2541	1.016	1.016	1.016	7.045
	Upper	3.772				
16807	Lower	1488	5950	5950	5950	5950
	Upper	5952				
397204094	Lower	.4256	1.702	1.702	1.702	28.61
	Upper	4.517				
630360016	Lower	.1502	.6008	1.546	1.546	4.057
	Upper	2.980				
7	Lower	3571400	14286000	14286000	14286000	14286000
	Upper	14286000				

[a] Lower bound $= 10^8 \times 1/\min(\lambda_m \prod_{i=0}^{k-1} |q_i|)$. [b] Upper bound $= 10^8 \times (2 + \sum_{i=1}^{p} a_i)/T$.

regarded as narrow. Second, the top five multipliers do not dominate $A = 397204094$ and 630360016 unambiguously, as in the earlier tables. This lack of discrimination on the part of the lower bounds on discrepancy may be due to the fact that discrepancy is not a rotation invariant measure. That is, it is developed along the lines of the classical serial test in Statistics in which the sides of the cells are parallel to the coordinate axes and hence discrepancy detects the worst case with regard to this orientation only. By contrast, $d_k^*(A, M)$ measures the worst case with regard to all possible orientations. Although one can argue that many statistical testing procedures rely exclusively on this Cartesian product space specification, the fact that our study reveals so many multipliers that perform well on the more stringent measure $d_k^*(A, M)$ encourages us to recommend this criterion for general use.

As mentioned earlier the Appendix contains a list of all multipliers for which $\min_{2\leqq k\leqq 6} S_{1,k}(A, M) \geqq .80$. A perusal of this list reveals six multipliers for which $S_{3,k}(A, M) \geqq .80$. While these multipliers do not rank as high as the five best with regard to $\min_{2\leqq k\leqq 6} S_{1,k}(A, M)$, their relatively good bivariate behavior with regard to $S_{1,k}(A, M)$ and $S_{3,k}(A, M)$ encourages us to examine them more closely. Table 6 shows how these multipliers perform with regard to lattice packing in the sample space and in the dual space. A comparison of these results with those in Tables 2 and 3 makes clear that these multipliers are equally acceptable with regard to lattice packing considerations. Whether or not some other justifiable basis exists for choosing these multipliers over the best five is not apparent at present.

TABLE 6
Packing measures for multipliers with
$S_{1,k}(A, M) \geqq .8$ *and* $S_{3,k}(A, M) \geqq .8$
$k = 2, \cdots, 6$.

Multiplier A		Dimension (k) 2	3	4	5	6
809609776	$\omega_k(A, M)$	3.17	3.76	4.51	4.51	8.26
	$\mu_k(A, M)$	3.17	4.23	4.55	5.07	6.71
1567699476	$\omega_k(A, M)$	2.88	3.66	4.98	6.55	9.96
	$\mu_k(A, M)$	2.88	3.15	4.37	5.72	6.71
1294711786	$\omega_k(A, M)$	3.08	2.72	4.95	5.44	9.85
	$\mu_k(A, M)$	3.08	4.73	4.73	4.17	5.65
1554283637	$\omega_k(A, M)$	2.56	3.71	4.71	6.08	7.79
	$\mu_k(A, M)$	2.56	4.15	4.27	5.74	6.38
857010188	$\omega_k(A, M)$	2.39	4.16	5.97	5.21	7.74
	$\mu_k(A, M)$	2.39	4.20	6.39	5.95	5.08
1582405117	$\omega_k(A, M)$	3.09	3.13	4.02	4.85	8.02
	$\mu_k(A, M)$	3.09	4.25	5.24	4.88	5.78
Upper bound		3.63	5.92	9.87	14.89	23.87

4. Empirical evaluations. In addition to evaluating the global properties of a multiplier, one needs to consider the local randomness properties of subsequences of moderate length that a generator with this multiplier produces. This evaluation is usually performed by statistically testing these subsequences to detect departures from randomness. Fishman and Moore (1982) described a comprehensive battery of tests for this purpose, and we apply the same battery here to test the five best multipliers.

Recall from (2) that $U_1, U_2, \cdots$ are the random numbers normalized to $(0, 1)$. Hypotheses to be tested include:

H_0: $\{U_i; i = 1, \cdots, n\}$ is a sequence of i.i.d. random variables.
H_1: $\{U_i; i = 1, \cdots, n\}$ have a uniform distribution on $(0, 1)$.
H_2: (U_{2i-1}, U_{2i}) $i = 1, \cdots, n/2$ have a uniform distribution on the unit square.
H_3: $(U_{3i-2}, U_{3i-1}, U_{3i})$ $i = 1, \cdots, (n-2)/3$ have a uniform distribution on the unit cube.
H_4: H_0, H_1, H_2 and H_3 hold simultaneously.

For each multiplier we collected 100 consecutive subsequences of $n = 200{,}000$ numbers. For each subsequence i and each hypothesis j a test statistic T_{ij} was computed. Then for hypothesis j, $T_{ij}, \cdots, T_{100,j}$ were subjected to the battery of tests. Let T_{ij} have continuous cumulative distribution function (c.d.f.) G_j under hypothesis j. Then $G_j(T_{ij})$ and $P_{i,j} = 1 - G_j(T_{ij})$ are distributed uniformly on $(0, 1)$ and for $0 < t < 1$

$$F_{n,j}(t) = \frac{1}{n} \sum_{i=1}^{n} I_{(0,t]}(P_{ij}),$$

where I_B denotes the indicator function on the set B, is an empirical c.d.f. If H_j is true

$$D_{n,j} = \sup_t |F_{n,j}(t) - t|$$

has the Kolmogorov-Smirnov (K-S) distribution,

$$V_{n,j} = n \int_0^1 I_{[0,t]}(F_{n,j}(t))\, dt$$

has the uniform distribution on $(0, 1)$ (Dwass 1958) and for large n

$$A^2_{n,j} = n \int_0^1 \{[F_{n,j}(t) - t]^2 / t(1-t)\}\, dt$$

has a distribution given by Anderson and Darling (1952), (1954) and is denoted by A-D. The quantity $D_{n,j}$ measures the absolute deviation between the empirical and the hypothesized c.d.f.; $V_{n,j}$ measures the proportion of $F_{n,j}$ that lies below the hypothesized c.d.f.; and $A^2_{n,j}$ is a weighted measure of the extent of deviation, principally in the tails, of the empirical c.d.f.

Since Fishman and Moore (1982) provided complete descriptions of the testing of $H_0, \cdots, H_4$, here we merely review the most essential details. In particular each test statistic T_{ij} was chosen as follows. To test H_0 we relied on a comprehensive analysis of runs-up and runs-down statistics. For H_1 we chose a chi-squared goodness-of-fit statistic with $2^{12} = 4096$ cells. For H_2 the serial test statistic was used for nonoverlapping 2-tuples with a total of 4096 cells in the unit square. For H_3, a serial test statistic was used for nonoverlapping 3-tuples and 4096 cells in the unit cube.

The hypothesis H_4 is omnibus in character. Recall that $P_{ij} = 1 - G_j(T_{ij})$ $i = 1, \cdots, 100$ $j = 0, 1, \cdots, 3$ and set

$$X_{ij} = \Phi^{-1}(P_{ij})$$

where Φ^{-1} is the inverse of the unit normal distribution. Under H_j, X_{ij} has the unit normal distribution and $X_{i0}, X_{i1}, \cdots, X_{i3}$ have a multinormal distribution function ψ. Let $X_{i,\min} = \min(X_{i,0}, \cdots, X_{i,3})$ and $X_{i,\max}(X_{i,0}, \cdots, X_{i,3})$. Then under H_4

$$\bar{T}_{i,4} = 1 - \psi(-X_{i,\min}, -X_{i,\min}, -X_{i,\min}, -X_{i,\min})$$

and

$$T_{i,4} = 1 - \psi(X_{i,\max}, X_{i,\max}, X_{i,\max}, X_{i,\max}),$$

each have the unit normal distribution. Since $\bar{T}_{i,4}$ and $T_{i,4}$ measure how likely one is to encounter values as extreme as $X_{i,\min}$ and $X_{i,\max}$, they provide valuable information about the truth of $H_0, \cdots, H_3$. Accordingly we used $\{\bar{T}_{i,4}; i = 1, \cdots, 100\}$ and $\{T_{i,4}; i = 1, \cdots, 100\}$ to test H_4. As an interim result a test of the multinormality of $X_{i,0}, \cdots, X_{i,3}$ was also performed.

Table 7 presents the P values for $H_0, \cdots, H_4$ and the multinormality test for the five best multipliers. Although several multipliers show some small P values, no systematic rejection occurs across the K-S, V and A-D tests and across hypotheses. If one feels compelled to rank the multipliers, one might regard $A = 950706376$ as first and $A = 1343714438$ as last. However, we emphasize that in a table with so many entries some low values are to be expected when all hypotheses are true. In summary we conclude that, in addition to having optimal global properties, the five multipliers show no empirical aberrations.

TABLE 7
P values for testing hypotheses.

Multiplier A	Test (1)	H_0 (2)	H_1 (3)	H_2 (4)	H_3 (5)	Multi-normality (6)	H_4 min (7)	H_4 max (8)
742938285	K-S	.735	.499	.306	.633	.922	.776	.802
	V	.853	.012[b]	.971	.491	.463	.278	.353
	A-D	.408	.231	.406	.796	.990	.545	.870
950706376	K-S	.361	.304	.636	.766	.163	.244	.529
	V	.974	.827	.616	.493	.443	.401	.322
	A-D	.269	.254	.497	.629	.173	.279	.417
1226874159	K-S	.738	.115	.081[a]	.903	.151	.220	.532
	V	.378	.468	.646	.395	.183	.425	.749
	A-D	.442	.083[a]	.172	.914	.166	.420	.802
62089911	K-S	.232	.506	.493	.073[a]	.578	.121	.132
	V	.618	.923	.773	.193	.160	.305	.345
	A-D	.328	.457	.539	.139	.377	.151	.144
1343714438	K-S	.771	.068[a]	.024[b]	.845	.635	.904	.230
	V	.849	.440	.158	.781	.577	.365	.404
	A-D	.806	.099[a]	.041[b]	.863	.542	.903	.195

[a] $.05 < P$ Value $\leqq 0.1$.
[b] $.01 < P$ Value $\leqq .05$.

Appendix[1].

A	$\min_k S_{1,k}$	$\min_k S_{3,k}$	A	$\min_k S_{1,k}$	$\min_k S_{3,k}$
742938285	0.8319	0.7838	1760624889	0.8112	0.7943
950706376	0.8274	0.7565	1442273554	0.8111	0.7110
1226874159	0.8255	0.7107	959387418	0.8110	0.7790
62089911	0.8249	0.7385	1113127164	0.8108	0.7726
1343714438	0.8236	0.7747	1446285050	0.8107	0.7677
2049513912	0.8232	0.6545	231487336	0.8107	0.7820
781259587	0.8212	0.7699	231487336	0.8107	0.7820
482920380	0.8204	0.7489	403636263	0.8102	0.7946
1810831696	0.8198	0.7652	365870474	0.8098	0.7375
502005751	0.8196	0.6930	1683348964	0.8098	0.7113
464822633	0.8191	0.7368	56469953	0.8095	0.7021
1980989888	0.8186	0.7345	391539936	0.8095	0.7495
329440414	0.8184	0.7271	621389603	0.8093	0.7676
1930251322	0.8182	0.7199	1697836722	0.8092	0.7616
800218253	0.8182	0.7386	209720443	0.8092	0.7582
1575965843	0.8181	0.7242	1651132469	0.8090	0.7805
1100494401	0.8170	0.6828	1036489797	0.8090	0.7381
1647274979	0.8168	0.7124	1094002855	0.8088	0.7044
62292588	0.8166	0.7594	958373200	0.8088	0.7173
1904505529	0.8166	0.7577	1882462218	0.8087	0.7956
1032193948	0.8164	0.7470	1901918329	0.8087	0.7586
1754050460	0.8155	0.7455	1482800924	0.8084	0.7763
1580850638	0.8154	0.7723	1609286051	0.8078	0.7430
1622264322	0.8154	0.7076	1873448661	0.8075	0.6724
30010801	0.8152	0.7441	1394633840	0.8075	0.7039
1187848453	0.8150	0.7312	1691910501	0.8075	0.7119
531799225	0.8148	0.7179	155279822	0.8075	0.6776
1402531614	0.8147	0.7277	1499553667	0.8073	0.7992
988799757	0.8145	0.7567	2117906721	0.8073	0.7198
1067403910	0.8144	0.7545	1337239139	0.8072	0.7897
1434972591	0.8142	0.7517	1257701541	0.8072	0.7358
1542873971	0.8142	0.7938	1061023798	0.8072	0.7087
621506530	0.8141	0.7158	659947220	0.8071	0.6689
473911476	0.8139	0.7548	1472802766	0.8071	0.7432
2110382506	0.8139	0.7783	1709954462	0.8069	0.7457
150663646	0.8138	0.7012	1437555212	0.8069	0.7240
131698448	0.8136	0.7740	2112159807	0.8069	0.7122
1114950053	0.8133	0.7568	1610356818	0.8068	0.7029
1768050394	0.8130	0.7509	1362323644	0.8068	0.6809
513482567	0.8127	0.7803	1528100852	0.8068	0.7778
513482567	0.8127	0.7803	644912347	0.8067	0.7856
1626240045	0.8127	0.7308	1640011312	0.8063	0.7232
2099489754	0.8127	0.7468	1267201170	0.8062	0.7771
1262413818	0.8127	0.6294	809609776	0.8061	0.8222
334033198	0.8125	0.6849	292397876	0.8061	0.7322
404208769	0.8124	0.7266	1022131410	0.8061	0.7509
257260339	0.8124	0.7366	1636624282	0.8061	0.7595
1006097463	0.8121	0.7780	672536717	0.8060	0.7532
1393492757	0.8121	0.7484	1292868648	0.8059	0.6673

[1] The remaining 207 multipliers can be computed as follows: Set $B=7$; for each multiplier A find the smallest integer I such that $A \equiv B^I \pmod{M}$. Then the multiplier $A^* \equiv B^{M-1-I} \pmod{M}$ has the same properties as A.

A	$\min_k S_{1,k}$	$\min_k S_{3,k}$	A	$\min_k S_{1,k}$	$\min_k S_{3,k}$
964028288	0.8115	0.7029	965146404	0.8059	0.7546
1493834601	0.8059	0.6905	737154017	0.8023	0.7564
1037566960	0.8058	0.7469	764970606	0.8023	0.6581
743722486	0.8058	0.7659	1074109599	0.8023	0.7944
1509089937	0.8057	0.7264	1039219247	0.8023	0.6029
1567699476	0.8057	0.8428	428641844	0.8022	0.6706
1947306937	0.8053	0.7164	1522856686	0.8022	0.7639
1076532097	0.8052	0.7503	1019054714	0.8020	0.7589
1957811727	0.8052	0.6839	805874727	0.8019	0.7295
628467148	0.8051	0.7540	1165699491	0.8018	0.7391
1040895393	0.8049	0.7252	258880375	0.8017	0.7245
786824435	0.8049	0.7909	1554283637	0.8017	0.8094
556530824	0.8049	0.7320	1155862579	0.8017	0.7911
87921290	0.8047	0.7402	848396760	0.8016	0.5756
1457913431	0.8047	0.7980	915892507	0.8016	0.7204
385787459	0.8046	0.7590	614779685	0.8016	0.7329
1567316532	0.8046	0.7568	391842496	0.8015	0.7255
930959341	0.8044	0.7790	380006810	0.8015	0.7456
1588813465	0.8044	0.7850	2011769251	0.8014	0.6802
1035519219	0.8043	0.7590	1860139263	0.8014	0.7729
36944245	0.8043	0.6932	1920597088	0.8014	0.6861
1891356973	0.8043	0.7058	1993412958	0.8014	0.7026
1897412292	0.8043	0.7112	511806823	0.8014	0.6100
754680739	0.8043	0.7447	979167897	0.8014	0.7860
1971204812	0.8043	0.7753	1956806422	0.8012	0.7521
1888847798	0.8042	0.6658	1256909708	0.8011	0.6410
1571641634	0.8040	0.7445	581488682	0.8011	0.6965
1117435554	0.8040	0.7243	334258581	0.8011	0.7065
569170662	0.8040	0.7292	68580478	0.8011	0.7568
927407259	0.8040	0.7149	534897944	0.8011	0.7808
1490690267	0.8039	0.7250	251676340	0.8009	0.6418
235716977	0.8039	0.7313	1051072528	0.8009	0.7125
149289625	0.8038	0.7028	2101655234	0.8009	0.7710
1660576129	0.8038	0.7851	1413698051	0.8008	0.7819
1517266187	0.8038	0.6827	796322341	0.8008	0.7611
1229881012	0.8037	0.7146	698108846	0.8008	0.7543
707656279	0.8037	0.7617	1544249456	0.8008	0.7187
1869095734	0.8037	0.6714	857010188	0.8008	0.8001
995560464	0.8037	0.7182	1860488201	0.8008	0.7639
539146268	0.8037	0.7505	355389105	0.8008	0.6647
1604187179	0.8036	0.7013	1774722449	0.8007	0.7413
2082150220	0.8035	0.7624	1582405117	0.8007	0.8176
370594724	0.8035	0.7375	553469741	0.8007	0.7233
2044924591	0.8035	0.6988	1411007767	0.8006	0.6678
916100787	0.8035	0.6079	1230102545	0.8006	0.7507
1037414126	0.8035	0.7866	356267478	0.8005	0.7199
1838122410	0.8033	0.7246	778084663	0.8005	0.7903
1265438464	0.8031	0.6262	1905014417	0.8005	0.6782
1007804709	0.8029	0.6410	1109871330	0.8005	0.7312
1257431879	0.8029	0.7876	1704318220	0.8004	0.7326
2061749697	0.8029	0.6603	270593738	0.8004	0.6510
737009774	0.8026	0.7135	483389111	0.8003	0.7821
408432740	0.8024	0.7514	323128013	0.8003	0.7395
876389446	0.8024	0.7398	361076890	0.8000	0.7293
1294711786	0.8024	0.8040			

REFERENCES

J. H. AHRENS AND U. DIETER (1977), *Uniform Random Numbers*, Univ. Graz.

T. W. ANDERSON AND D. A. DARLING (1952), *Asymptotic theory of goodness of fit criteria based on stochastic processes*, Ann. Math. Statist., 23, pp. 193-212.

——— (1954), *A test of goodness of fit*, J. Amer. Statist. Assoc., 49, pp. 765-769.

W. A. BEYER, R. B. ROOF AND D. WILLIAMSON (1971), *The lattice structure of multiplicative congruential pseudo-random vectors*, Math. Comput., 25, pp. 345-363.

I. BOROSH AND H. NIEDERREITER (1983), *Optimal multipliers for pseudo-random number generation by the linear congruential method*, BIT, 23, pp. 65-74.

J. W. S. CASSELS (1959), *An Introduction to the Geometry of Numbers*, Springer-Verlag, New York.

R. R. COVEYOU (1970), *Random number generation is too important to be left to chance*, Stud. Appl. Math., 3, pp. 70-111.

R. R. COVEYOU AND R. D. MACPHERSON (1967), *Fourier analysis of uniform random number generators*, J. Assoc. Comput. Mach., 14, pp. 100-119.

M. DWASS (1958), *On several statistics related to empirical distribution functions*, Ann. Math. Statist., 29 pp. 188-191.

U. DIETER (1971), *Pseudo-random numbers: the exact distribution of pairs*, Math. Comp., 25, pp. 855-883.

——— (1975), *How to calculate shortest vectors in a lattice*, Math. Comp., 29, pp. 827-833.

G. S. FISHMAN AND L. R. MOORE (1982), *A statistical evaluation of multiplicative congruential random number generators with modulus* $2^{31}-1$, J. Amer. Statist. Assoc., 77, pp. 129-136.

G. H. HARDY AND E. M. WRIGHT (1960), *The Theory of Numbers*, 4th ed., Clarendon Press, Oxford.

D. HOAGLIN (1976), *Theoretical properties of congruential random-number generators: an empirical view*, Memorandum NS-340, Dept. Statistics, Harvard Univ., Cambridge, MA.

IMSL (1980), IMSL *Library Reference Manual*, 8th ed., IMSL INC., Houston, TX.

B. JANNSON (1966), *Random Number Generators*, Almqvist and Wiksell, Stockholm.

H. KATZAN JR. (1971), APL *User Guide*, Van Nostrand Reinhold, New York

P. KIVIAT, R. VILLANUEVA AND H. MARKOWITZ (1969), *The* SIMSCRIPT II *Programming Language*, Prentice-Hall, Englewood Cliffs, NJ.

D. E. KNUTH (1981), *The Art of Computer Programming Vol.* 2: *Semi-numerical Algorithms*, 2nd ed., Addison-Wesley, Reading, MA.

D. H. LEHMER, (1981), *Mathematical methods in large scale computing units*, Ann. Comp. Labs., 26, pp. 141-146, Harvard Univ., Cambridge, MA.

G. MARSAGLIA, (1968), *Random numbers fall mainly in the plane*, Proc. Nat. Acad. Sci., 61, pp. 25-28.

——— (1972), *The structure of linear congruential sequences*, in Applications of Number Theory to Numerical Analysis, S. K. Zaremba, ed., Academic Press, New York.

H. NEIDERREITER (1976), *Statistical independence of linear congruential pseudo-random numbers*, Bull. Amer. Math. Soc., 82, pp. 927-929.

——— (1977), *Pseudo-random numbers and optimal coefficients*, Adv. Math., 26, pp. 99-181.

——— (1978a), *The serial test for linear congruential pseudo-random numbers*, Bull. Amer. Math. Soc., 84, pp. 273-274.

——— (1978b), *Quasi-Monte Carlo methods and pseudo-random numbers*, Bull. Amer. Math. Soc., 84, pp. 957-1041.

W. H. PAYNE, J. R. RABUNG AND T. P. BOGYO (1969), *Coding the Lehmer pseudorandom number generator*, Comm. ACM, 12, pp. 85-86.

SAS Institute Inc. (1982), SAS *User's Guide*: *Basics*, Cary, NC.

C. S. SMITH (1971), *Multiplicative pseudo-random number generators with prime modulus*, J. Assoc. Comput. Mach., 18, pp. 586-593.

SIAM J. SCI. STAT. COMPUT.
Vol. 7, No. 1, January 1986

003

NUMERICAL SOLUTIONS FOR BAYES SEQUENTIAL DECISION PROBLEMS*

HERMAN CHERNOFF† AND A. JOHN PETKAU‡

Abstract. Certain sequential decision problems involving normal random variables reduce to optimal stopping problems which can be related to the solution of corresponding free boundary problems for the heat equation. The numerical solution of these free boundary problems can then be approximated by calculating the solution of simpler optimal stopping problems by backward induction. This approach is not well adapted for very precise results but is surprisingly effective for rough approximations. An estimate of the difference between the solutions of the related problems permits one to make continuity corrections which provide considerably improved accuracy. Further reductions in the necessary computational effort are possible by considering truncated procedures for one-sided boundaries and by exploiting monotone and symmetric boundaries.

Key words. backward induction, Bayes risk, decision theory, free boundary problem, optimal stopping, random walk, sequential analysis, Wiener process

AMS(MOS) subject classifications. Primary 62L10; secondary 65D99

1. Introduction. Certain Bayes sequential decision problems involving normal random variables reduce to discrete time optimal stopping problems of the following form. Let $Y(s)$ be a Wiener process in the $-s$ scale for $s_0 \geqq s \geqq s^*$ with $Y(s_0) = y_0$ and

$$E\{dY(s)\} = 0, \qquad \text{Var}\{dY(s)\} = -ds.$$

Select a *stopping time* $S \in \{s_0, s_1, s_2, \cdots, s_n, \cdots\}$ with $s_0 \geqq s_1 \geqq \cdots \geqq s_n \geqq s^*$ so as to minimize the *risk* or *expected cost* $E\{d(Y(S), S)\}$, where $d(y, s)$ is the cost associated with stopping at the point (y, s) and stopping is enforced at the end of the problem, namely, when $s = s^*$. A continuous time version of this problem is the same except that the stopping time S may take on any value in the interval $[s_0, s^*]$.

Two examples of importance in the statistical literature are the problems of deciding the sign of the mean of a normal distribution [1], [3]-[6], [8], [15], [16], [17] and the one-armed bandit problem [7], [8], [13]. Normalized forms of continuous time versions of these problems have stopping cost functions

$$d_1(y, s) = s^{-1} + s^{1/2}\psi(ys^{-1/2}) \quad \text{for } s \geqq 0 \tag{1.1}$$

and

$$d_2(y, s) = -\frac{y}{s} \quad \text{for } s \geqq 1, \tag{1.2}$$

respectively, where ϕ and Φ are the density and cumulative distribution functions for the standard normal distribution and

$$\psi(x) = \begin{cases} \phi(x) - x\{1 - \Phi(x)\} & \text{for } x \geqq 0, \\ \psi(-x) & \text{for } x < 0. \end{cases}$$

* Received by the editors February 27, 1984, and in final form August 27, 1984. This research was supported in part by the Office of Naval Research under contracts N00014-75-C-0555 (NR-042-331 and NR-609-001) and N00014-67-A-0112-0085 (NR-042-267) and in part by the Natural Sciences and Engineering Research Council of Canada under research grants A4586 and T1829.

† Statistics Center, Massachusetts Institute of Technology, Cambridge, Massachusetts 02139.

‡ Department of Statistics, University of British Columbia, Vancouver, British Columbia, Canada V6T 1W5.

The possible stopping times for the discrete time versions are of the form $s_n = a(\sigma_0^{-2} + n\sigma^{-2})^{-1}$ for $n = 0, 1, 2, \cdots$, where a is a constant arising from the normalization.

The solutions of these continuous time optimal stopping problems can be expressed in terms of a *stopping set* $\mathcal{S}$ and a *continuation set* $\mathcal{C} = \mathcal{S}^c$ in the (y, s) plane; that is, S consists of stopping when $(Y(s), s)$ reaches $\mathcal{S}$ as s decreases from s_0. These are related to those of corresponding *free boundary problems* involving the heat equation. More precisely, that free boundary problem is to find $(\mathcal{S}, b)$ so that

$$\begin{aligned} &\tfrac{1}{2}b_{yy}(y, s) = b_s(y, s) \quad \text{for } (y, s) \in \mathcal{C}, \\ &b(y, s) = d(y, s) \quad \text{for } (y, s) \in \mathcal{S}, \\ &b_y(y, s) = d_y(y, s) \quad \text{for } (y, s) \in \partial\mathcal{S}, \end{aligned} \tag{1.3}$$

where $\partial\mathcal{S}$ is the boundary of $\mathcal{S}$. The solution b of the free boundary problem corresponds to the optimal risk $\hat{d}$ of the stopping problem, that is,

$$b(y_0, s_0) = \hat{d}(y_0, s_0) = E\{d(Y(S), S)\}. \tag{1.4}$$

This relationship and several others stated in this paper are subject to regularity conditions on which we shall not here elaborate.

Discrete time versions of these problems can be regarded as special cases of the continuous time versions where stopping is restricted to a limited subset of the (y, s) space, and hence the optimal risks and related stopping sets are larger. Moreover, the discrete time solutions converge monotonically to the continuous time solutions if the set of possible stopping times $\{s_0, s_1, s_2, \cdots\}$ increases and $\sup_i |s_i - s_{i+1}| \to 0$.

The continuous time versions have statistical interpretations of their own, but they may be considered as approximations to the discrete time versions, approximations which are easier to analyze and for which asymptotic expansions and bounds can be derived. While such results provide valuable insight, in most problems they fall short of providing an adequate approximation to the solution. Consequently when closed form solutions are unavailable, as is usually the case, it is important to have techniques capable of providing numerical approximations. By what almost amounts to circular reasoning, we can find such approximations by solving discrete time versions by backward induction using a set of possible stopping times chosen for our convenience. The equation governing the backward induction solution is

$$\hat{d}(y, s_i) = \min\left[d(y, s_i), E\{\hat{d}(y + Z\sqrt{s_i - s_{i+1}}, s_{i+1})\}\right] \tag{1.5}$$

where Z is a standard normal random variable. This is not quite circular for three reasons. First, the s_i in the numerical approximation may be selected to be equally spaced although this was not the case for the possible stopping times in the original discrete time problem. Second, a process of normalization using the fact that aY is a Wiener process in the $-a^2 s$ scale, makes one normalized continuous time problem approximate a family of discrete time problems and hence makes one numerical calculation applicable to the entire family. Third, there is a *continuity correction* of the form

$$\tilde{y}_\delta(s) = \tilde{y}(s) \pm 0.5826\sqrt{\delta} \tag{1.6}$$

which applies when the spacing between many successive s_i is approximately δ, where $\tilde{y}_\delta$ and $\tilde{y}$ represent the optimal boundaries for the discrete and continuous time versions and the sign is determined so as to make the continuation region for the continuous time version larger. This correction may be used to go from the backward induction

to the continuous time version, and then again to go from the latter to the original discrete time problem. This continuity correction yields a considerably more accurate estimate of the optimal boundary.

Since the backward induction (1.5) involves a numerical integration to evaluate the risk at each point (y, s_i), it could be quite time consuming. Let us replace the Wiener process $Y(s)$ by a related process Y^* of independent Bernoulli increments $Z_i^*\sqrt{\delta}$, where $s_i - s_{i+1} = \delta$, and $Z_i^* = \pm 1$ each with probability $\frac{1}{2}$, that is,

$$Y^*(s_{i+1}) = Y^*(s_i) + Z_i^*\sqrt{\delta}.$$

Then, as for Y, the mean and variance of the independent increments are 0 and $s_i - s_{i+1}$, respectively. The backward induction solution of the optimal stopping problem of minimizing $E\{d(Y^*(S), S)\}$ for $S \in \{s_0, s_1, s_2, \cdots\}$ involves the simple calculation

$$\hat{d}^*(y, s_i) = \min\left[d(y, s_i), 0.5\{\hat{d}^*(y+\sqrt{\delta}, s_{i+1}) + \hat{d}^*(y-\sqrt{\delta}, s_{i+1})\}\right], \tag{1.7}$$

which can be confined to a grid of points

$$\{y = c \pm j\sqrt{\delta},\ s_i = s_0 - i\delta\} \tag{1.8}$$

for integer values of i and j. Each point on the grid can be determined to be a stopping or continuation point depending on whether $\hat{d}^*(y, s_i) = d(y, s_i)$ or $\hat{d}^*(y, s_i) < d(y, s_i)$. For most purposes, a grid with the s axis as one of its vertices ($c = 0$) would suffice.

In § 2 we describe the continuity correction for the optimal boundary using the above discrete time discrete process approach. In § 3 we see how the computing requirements of this method can be reduced by a truncation procedure in problems with one-sided optimal continuation regions. In § 4 we discuss the additional reduction allowed under symmetry and monotonicity. In § 5, the problem of extending the solution to large values of s is addressed by increasing, in stages, the distance between successive values of s. Finally, in § 6, results are presented for the two statistical problems mentioned above.

2. Continuity corrections. The continuity corrections are related to the difference between the discrete time versions and the continuous time version of the *canonical problem* defined by

$$d(y, s) = \begin{cases} -s & \text{for } s > 0 = s^*, \\ h(y) = 0 & \text{for } s = 0, \quad y > 0, \\ h(y) = -y^2 & \text{for } s = 0, \quad y \leqq 0. \end{cases} \tag{2.1}$$

The discrete time versions have the nonnegative integers as the possible stopping times. For the continuous time version, $b(y, s) = h(y) - s$, $\tilde{y}(s) = 0$ and the one-sided optimal continuation region is $\mathscr{C} = \{(y, s): y < 0, s > 0\}$. In [6], it was shown that for the discrete time version with normal increments, $\tilde{y}_1(n) \to -0.5826$ as $n \to \infty$. Note that the subscript 1 in $\tilde{y}_1(n)$ refers to the spacing $\delta = 1$, and $0.5826 \cdots = -\zeta(\frac{1}{2})/\sqrt{2\pi}$, where ζ is the Riemann zeta function. In [9], the solution of the discrete time version with Bernoulli increments is described by $\tilde{y}_1^*(n)$ and $\hat{d}^*(y, n)$ where, as $n \to \infty$,

$$\hat{d}^*(y, n) + n \to h_1(y), \tag{2.2}$$

$$\tilde{y}_1^*(n) \to -0.5, \tag{2.3}$$

and

$$h_1(y) = \begin{cases} 0 & \text{for } y \geqq -0.5, \\ -y^2 + \inf\limits_i (y+i)^2 & \text{for } y < -0.5. \end{cases} \tag{2.4}$$

Following the derivation in [6], it can be seen that for our general stopping problem when $\mathscr{C}$ is below $\mathscr{S}$ at the boundary point $(\tilde{y}(s), s)$,

$$\tilde{y}_\delta^*(s) \approx \tilde{y}(s) - 0.5\delta^{1/2} \tag{2.5}$$

where $\tilde{y}_\delta^*$ represents the optimal boundary for the discrete time version with Bernoulli increments. Further, for $y \in \mathscr{C}$ and close to $\tilde{y}(s)$,

$$D(y, s) \equiv \hat{d}^*(y, s) - d(y, s) \approx a\delta h_1\{(y - \tilde{y}(s))/\delta^{1/2}\}, \tag{2.6}$$

where

$$a = \tfrac{1}{2}d_{yy}(\tilde{y}(s), s) - d_s(\tilde{y}(s), s) \geqq 0 \tag{2.7}$$

may be interpreted as the local rate of loss associated with continuing from $(\tilde{y}(s), s)$ for a short fixed period of time and then stopping.

If we could calculate $\tilde{y}_\delta^*(s)$, we could use (2.5) for the continuity correction. Ordinarily $\tilde{y}_\delta^*$ will fall somewhere between the vertices of the grid on which our backward induction is computed and is not calculated directly. Although $\tilde{y}_\delta^*(s)$ could be approximated in various ways, the most expedient way to derive a correction is to use the values D_0 and D_1 of D at $y_0^*(s)$ and $y_1^*(s)$, the continuation points on the grid which are closest and second closest to the stopping region at the stopping time s, and to apply (2.6). Assuming that the boundary is above $y_0^*(s)$, if we represent

$$y_0^*(s) = \tilde{y}(s) + v\delta^{1/2}$$

and

$$y_1^*(s) = \tilde{y}(s) + (v-1)\delta^{1/2},$$

where $-1.5 < v \leqq -0.5$ because $y_0^* \leqq \tilde{y}_\delta^* \approx \tilde{y} - 0.5\delta^{1/2}$, we have

$$\begin{aligned} D_0 &\approx a\delta h_1(v) = a\delta[-v^2 + (v+1)^2] = a\delta(2v+1), \\ D_1 &\approx a\delta h_1(v-1) = a\delta[-(v-1)^2 + (v+1)^2] = a\delta(4v). \end{aligned} \tag{2.8}$$

Hence,

$$v = \frac{D_1}{4D_0 - 2D_1} \tag{2.9}$$

and the continuity correction becomes

$$\tilde{y}(s) = y_0^*(s) + \left\{\frac{D_1}{2(D_1 - 2D_0)}\right\}\delta^{1/2}. \tag{2.10}$$

Thus, by applying two corrections we can approximate the solution to the original discrete time normal version of our optimal stopping problem. That is, we calculate the backward induction solution with the discrete time Bernoulli process, use (2.10) to approximate the solution to the continuous time problem and end by applying (1.6) to estimate the solution to the original discrete time normal version.

Occasionally the statistical problem of interest involves Bernoulli rather than normal random variables. However, these Bernoulli variables may correspond to $p \neq \frac{1}{2}$; see, for example, the sequential decision models for clinical trials considered in [7], [10], [12] and [18]. In that case, the continuity correction for the return to this problem from the solution of the continuous time problem is different than either of the two corrections mentioned before, although the general approach is the same.

The canonical problem changes to one involving independent increments which take on the values a and b with probabilities p and $1-p$, respectively, where a and b are such that the increments have mean 0 and variance 1. The stopping cost function remains the same. Here, as $n \to \infty$,

$$\tilde{y}_1^*(n) \to g(p)$$

and

$$\hat{d}^*(y, n) + n \to h_1(y, p)$$

where g and h_1 are described in [9] in terms of the solution of a functional equation. These results provide a continuity correction similar to (2.10) for this case.

3. Truncation for one-sided continuation sets. The solution of the one-armed bandit problem is a one-sided continuation set $\mathscr{C} = \{(y, s): y > \tilde{y}(s), s > 1\}$ where $\tilde{y}(1) = 0$ and $\tilde{y}(s)$ decreases monotonically. The backward induction method we described is such that the calculation of $\hat{d}^*$ at a given point (y, s_i) requires that of $\hat{d}^*$ at a triangular array of points $(y \pm j\delta, s_k)$ with $|j| \leqq k - i$, $k = i+1, i+2, \cdots$. If we consider $\hat{d}^*$ as the expected optimal stopping cost, we may bypass some of these calculations for $y > c$ if $\tilde{y}(s) < c$ for $s_0 \geqq s > s^*$. Consider a path of the discrete time process with Bernoulli increments originating at the grid point $(y, s) = (c + \delta^{1/2}, s^* + n\delta)$. Then

$$\hat{d}^*(c+\delta^{1/2}, s^*+n\delta) = \sum_{m=1}^{n} p_m \hat{d}^*(c, s^*+(n-m)\delta) + \sum_{j=1}^{n+1} q_{n,j} d(c+j\delta^{1/2}, s^*) \tag{3.1}$$

where p_m is the probability that $\sum_1^r Z_i^*$ first hits -1 at $r = m$ and $q_{n,j}$ is the probability that $\sum_1^r Z_i^*$ never hits -1 for $r \leqq n$ and that $\sum_1^n Z_i^* = j - 1$. From [14, p. 89] it follows that

$$p_m = \frac{1}{m}\binom{m}{(m+1)/2} 2^{-m} \quad \text{for } m \text{ odd} \tag{3.2}$$

and 0 for m even, and from [14, p. 73] we have

$$q_{n,j} = \frac{j}{n+1}\binom{n+1}{(n+j+1)/2} 2^{-n} \tag{3.3}$$

except when $n+j$ is even in which case $q_{n,j} = 0$. The application of the backward induction for $(c+j\delta^{1/2}, s^*+n\delta)$ with $j \leqq 0$ no longer requires the computation of $\hat{d}^*$ at the grid points $(c+j\delta^{1/2}, s^*+m\delta)$ with $j > 1$, $1 \leqq m < n$.

The computational effort is reduced still further if $d(y, s^*) = 0$ for $y > c$ since the second sum in (3.1) then vanishes. If $d(y, s^*) \neq 0$ for $y > c$, it is possible to modify d without affecting the optimal boundary. In detail, we observe first that

$$u(y, s) = \frac{1}{\sqrt{s - s^*}} \int_{c^*}^{\infty} \phi\left(\frac{y - v}{\sqrt{s - s^*}}\right) d(v, s^*)\, dv \tag{3.4}$$

is a solution of the heat equation, corresponding to a source at $G = \{(y, s): y \geqq c^*, s = s^*\}$, which coincides with $d(y, s)$ on G. Second, if the stopping problem with stopping cost $d(y, s)$ is replaced by $d'(y, s) = d(y, s) - u(y, s)$ where u is a solution of the heat equation for $s > s^*$, then $b' = b - u$ and both problems have the same optimal stopping region.

The economic advantage of introducing d' depends on the effort involved in computing d'. In some cases this involves an integral which is easy to calculate. In other cases the effort required to evaluate the integral numerically throughout the course of the backward induction makes this "simplification" uneconomic.

For the one-armed bandit problem, $\tilde{y}(s)<0$ for $s>1=s^*$ and the truncation procedure can be implemented with $c=0$. The subtraction of $u=-y$, which corresponds to (3.4) with $c^*=-\infty$, leaves

$$d_2'(y, s)\equiv d_2(y, s)-u(y, s)=y(1-s^{-1}) \quad \text{for } s\geqq 1 \tag{3.5}$$

and accomplishes our goal of having the second sum in (3.1) vanish.

4. Monotonicity and symmetry. For the sequential analysis problem the boundary $\tilde{y}(s)$ of the optimal continuation set is monotonic and symmetric. For the one-armed bandit problem it is monotonic. Symmetry of $d(y, s)$ about $y=0$ implies that the computations involved in the backward induction can be confined to $y\geqq 0$. Monotonicity can also be exploited.

For the one-armed bandit problem $\tilde{y}(s)$ decreases in s and it is easy to show that $\tilde{y}_\delta^*$ inherits this property. Hence $\hat{d}^*(y, s_i)=0.5\{\hat{d}^*(y+\sqrt{\delta}, s_i-\delta)+\hat{d}^*(y-\sqrt{\delta}, s_i-\delta)\}$ for $y\geqq y_0^*(s_{i+1})$ and in computing the backward induction, there is no need to make the comparison between the above average and $d(y, s_i)$ until $y<y_0^*(s_{i+1})$. For small δ, y_0^* tends to remain constant and we ordinarily have to make this comparison only at one or two values of y before reaching the stopping set, after which $\hat{d}^*=d$. This property of monotonicity can also be exploited for two-sided continuation sets.

5. Large values of s. When δ is small the computational effort in reaching large values of s becomes prohibitive. For large values of s, we recommend changing the interval δ after a number of steps of the backward induction. By multiplying δ by a squared integer r^2 after reaching $s=s^{(1)}$, the appropriate increment in Y^* is changed by a factor r and the backward induction can be extended to this coarser grid for $s\geqq s^{(1)}$ without interpolating. The simple minded direct application of this method introduces some "ripples" in $\hat{d}^*$ and the approximation to $\tilde{y}(s)$ as one moves from one *stage* to the next. These ripples smooth out rapidly. Theory suggests that the ripple in the boundary cancels out quickly but that there may be a small average bias effect on $\hat{d}^*$. In our applications the risk ordinarily tends to increase with s and this bias remains relatively negligible.

To compensate for the ripple effect, we usually overlap successive stages. Thus for a problem with $s^*=1$, we may carry out the backward induction for $s=1(10^{-6})1.002$ and then use the results for $s=1.001$ to do $s=1.001(4\times 10^{-6})1.009$, and so on. It is relatively easy to move to very large values of s using a sequence of such overlapping stages.

When s is large, the round-off error depends on the magnitude of the risks involved. Greater numerical stability is anticipated if the risks are reduced by the addition of an appropriate solution of the heat equation. This has been exploited in several problems. For example, in [10] the continuous time version of Anscombe's formulation of the problem of comparing two treatments in a medical trial is reduced to a problem of the form under consideration with stopping cost function

$$d(y, s)=-(1-s^{-1})|y| \quad \text{for } s\geqq 1.$$

The optimal continuation region is $\mathscr{C}=\{(y, s): |y|<\tilde{y}(s), s>1\}$ where for large s, $\tilde{y}(s)\sim\{2s\log s\}^{1/2}$. For the computations, $d(y, s)$ was replaced by

$$\begin{aligned} d'(y, s) &= d(y, s)+\{|y|+2s^{1/2}\psi(ys^{-1/2})\} \\ &= |y|s^{-1}+2s^{1/2}\psi(ys^{-1/2}). \end{aligned}$$

For large v, $\psi(v)\sim v^{-2}\phi(v)$, and for large s, $d'(y, s)$ is small near the boundary $\tilde{y}(s)$. The cost involved is that of calculating ψ which is not trivial but not enormous.

6. Applications. The overall mechanics of the proposed techniques for reaching large values of s are specified by the grid spacing δ, the number of iterations in each stage of the backward induction and the extent of overlapping from each stage to the next. We have not systematically explored the possible versions of the technique, but rather have used the simple version in which the number of iterations is the same in all stages, the grid spacing is increased by the same multiple from each stage to the next, and the extent of overlapping is a fixed fraction of the interval of s values corresponding to the iterations of the previous stage.

The results presented were obtained using the technique with 2080 iterations in each stage, δ being increased by a factor of 4 from each stage to the next, and the extent of overlapping corresponding to one-half the interval of s values covered by the previous stage; this corresponds to 1040 iterations of the previous stage or 260 iterations of the current stage. Only grids centered on the s axis were employed (use of $c=0$ in (1.8)). For both examples, the grid spacing for the first stage was taken to be $\delta = 25\times 10^{-6}$ and estimates were obtained out to $s = 10^6$.

Example 1. *Deciding the sign of a normal mean.* Estimates of the stopping boundary for this problem are tabulated in various scales of interest in Table 1. The asymptotic expansions

$$\tilde{x}(s)=\frac{\tilde{y}(s)}{s}\sim\frac{1}{4}s\left[1-\frac{1}{12}s^3+\frac{7}{240}s^6-\cdots\right] \qquad \text{as } s\to 0,$$

$$\tilde{z}(s)=\frac{\tilde{y}(s)}{s^{1/2}}\sim\frac{1}{4}s^{3/2}\left[1-\frac{1}{12}s^3+\frac{7}{240}s^6-\cdots\right] \qquad \text{as } s\to 0,$$

$$\tilde{\beta}(s)=1-\Phi(\tilde{z}(s))\sim\frac{1}{2}-\frac{1}{\sqrt{2\pi}}\frac{1}{4}s^{3/2}\left[1-\frac{7}{96}s^3+\cdots\right] \quad \text{as } s\to 0$$

can be used to extend the table to even smaller values of s. On the other hand, the asymptotic expansions

$$\tilde{x}(s)=\frac{\tilde{y}(s)}{s}\sim s^{-1/2}\{\log s^3-\log(8\pi)-6(\log s^3)^{-1}+\cdots\}^{1/2} \quad \text{as } s\to\infty,$$

$$\tilde{z}(s)=\frac{\tilde{y}(s)}{s^{1/2}}\sim\{\log s^3-\log(8\pi)-6(\log s^3)^{-1}+\cdots\}^{1/2} \qquad \text{as } s\to\infty,$$

$$\tilde{\beta}(s)=1-\Phi(\tilde{z}(s))\sim 2(s^3\log s^3)^{-1/2}\{1+[2+\tfrac{1}{2}\log(8\pi)](\log s^3)^{-1}+\cdots\} \quad \text{as } s\to\infty$$

perform only moderately well at $s = 10^6$.

The Bayes risk corresponding to starting at the point (y_0, s_0) in the normalized form of the continuous time version is given by

$$\text{BR}\equiv E\{d_1(Y(S), S)\}-s_0^{-1},$$

and the contribution to the Bayes risk of the cost of sampling is given by

$$\text{ECS}\equiv E(S^{-1})-s_0^{-1},$$

where S is an optimal stopping rule and $d_1(y, s)$ is given in (1.1). BR and ECS depend only upon the initial values s_0 and y_0 or $t_0=1/s_0$ and $z_0=y_0/s_0^{1/2}$; some results are

TABLE 1

Estimates of stopping boundary for sequential analysis problem.

$t=1/s$	$\tilde{x}(s)=\tilde{y}(s)/s$	$\tilde{z}(s)=\tilde{y}(s)/s^{1/2}$	$\tilde{\beta}(s)=1-\Phi(\tilde{z}(s))$	$t=1/s$	$\tilde{x}(s)=\tilde{y}(s)/s$	$\tilde{z}(s)=\tilde{y}(s)/s^{1/2}$	$\tilde{\beta}(s)=1-\Phi(\tilde{z}(s))$
10.00	.02499	.0079	.4968	.16	.5144	1.2859	.09924
9.50	.02630	.0085	.4966	.15	.5159	1.3319	.09144
9.00	.02776	.0093	.4963	.14	.5170	1.3818	.08351
8.50	.02939	.0101	.4960	.13	.5175	1.4352	.07562
8.00	.03123	.0110	.4956	.12	.5170	1.4926	.06778
7.50	.03331	.0122	.4951	.11	.5158	1.5552	.05995
7.00	.03569	.0135	.4946	.10	.5134	1.6234	.05225
6.50	.03843	.0151	.4940	.09	.5096	1.6985	.04471
6.00	.04163	.0170	.4932	.08	.5039	1.7816	.03741
5.50	.04541	.0194	.4923	.07	.4961	1.8751	.03039
5.00	.04995	.0223	.4911	.06	.4855	1.9819	.02375
4.50	.05550	.0262	.4896	.05	.4707	2.1052	.01764
4.00	.06243	.0312	.4875	.04	.4507	2.2534	.01212
3.50	.07130	.0381	.4848	.03	.4222	2.4376	.007392
3.00	.08315	.0480	.4809	.02	.3797	2.6848	.003629
2.50	.09961	.0630	.4749	.01	.3074	3.0738	.001057
2.00	.1240	.0877	.4651	9 (−3)	.2969	3.1294	.8759 (−3)
1.50	.1636	.1336	.4469	8 (−3)	.2853	3.1903	.7107 (−3)
1.40	.1744	.1474	.4414	7 (−3)	.2726	3.2583	.5605 (−3)
1.30	.1865	.1636	.4350	6 (−3)	.2583	3.3345	.4273 (−3)
1.20	.2004	.1830	.4274	5 (−3)	.2420	3.4231	.3096 (−3)
1.15	.2080	.1940	.4231	4 (−3)	.2231	3.5276	.2097 (−3)
1.10	.2162	.2061	.4184	3 (−3)	.2004	3.6581	.1271 (−3)
1.05	.2250	.2196	.4131	2 (−3)	.1714	3.8329	.6334 (−4)
1.00	.2344	.2344	.4073	1 (−3)	.1300	4.1118	.1964 (−4)
.95	.2451	.2515	.4007	9 (−4)	.1246	4.1518	.1650 (−4)
.90	.2563	.2702	.3935	8 (−4)	.1187	4.1964	.1357 (−4)
.85	.2679	.2906	.3857	7 (−4)	.1123	4.2461	.1088 (−4)
.80	.2805	.3136	.3769	6 (−4)	.1054	4.3028	.8440 (−5)
.75	.2940	.3395	.3671	5 (−4)	.09767	4.3681	.6271 (−5)
.70	.3085	.3688	.3561	4 (−4)	.08895	4.4473	.4352 (−5)
.65	.3240	.4019	.3439	3 (−4)	.07874	4.5462	.2733 (−5)
.60	.3409	.4401	.3299	2 (−4)	.06620	4.6811	.1428 (−5)
.55	.3590	.4840	.3142	1 (−4)	.04902	4.9020	.4749 (−6)
.50	.3781	.5348	.2964	9 (−5)	.04681	4.9346	.4021 (−6)
.48	.3862	.5574	.2886	8 (−5)	.04446	4.9709	.3337 (−6)
.46	.3943	.5814	.2805	7 (−5)	.04193	5.0113	.2707 (−6)
.44	.4026	.6069	.2719	6 (−5)	.03918	5.0578	.2124 (−6)
.42	.4111	.6343	.2629	5 (−5)	.03614	5.1113	.1602 (−6)
.40	.4198	.6637	.2534	4 (−5)	.03274	5.1773	.1128 (−6)
.38	.4285	.6951	.2435	3 (−5)	.02881	5.2603	.7204 (−7)
.36	.4373	.7288	.2331	2 (−5)	.02403	5.3742	.3854 (−7)
.34	.4461	.7651	.2221	1 (−5)	.01759	5.5638	.1323 (−7)
.32	.4550	.8042	.2106	9 (−6)	.01678	5.5917	.1127 (−7)
.30	.4637	.8466	.1986	8 (−6)	.01590	5.6227	.9428 (−8)
.28	.4724	.8928	.1860	7 (−6)	.01497	5.6581	.7675 (−8)
.26	.4809	.9432	.1728	6 (−6)	.01396	5.6986	.6056 (−8)
.24	.4892	.9986	.1590	5 (−6)	.01285	5.7457	.4590 (−8)
.22	.4969	1.0595	.1447	4 (−6)	.01161	5.8031	.3266 (−8)
.20	.5038	1.1264	.1300	3 (−6)	.01018	5.8759	.2110 (−8)
.19	.5069	1.1629	.1224	2 (−6)	.008453	5.9770	.1140 (−8)
.18	.5097	1.2013	.1148	1 (−6)	.006146	6.1456	.3999 (−9)
.17	.5121	1.2421	.1071				

presented in Tables 2 and 3. These results reflect the form of the asymptotic expansions

$$\text{BR} \sim K s_0^{-1/2} \phi(z_0) \quad \text{as } s_0 \to \infty,$$

$$\text{ECS} \sim K' s_0^{-1/2} \phi(z_0) \quad \text{as } s_0 \to \infty,$$

provided in [5].

Example 2. *One-armed bandit problem.* Estimates of the stopping boundary are provided in Table 4. The asymptotic expansions

$$\tilde{x}(s) = \frac{\tilde{y}(s)}{s} \sim -(s-1)^{1/2}\{c_0 + (c_1 - c_0)(s-1) + \cdots\} \quad \text{as } s \to 1,$$

$$\tilde{z}(s) = \frac{\tilde{y}(s)}{s^{1/2}} \sim -(s-1)^{1/2}\left\{c_0 + \left(c_1 - \frac{1}{2}c_0\right)(s-1) + \cdots\right\} \quad \text{as } s \to 1,$$

$$\tilde{\beta}(s) = \Phi(\tilde{z}(s)) \sim \frac{1}{2} - \frac{1}{\sqrt{2\pi}}(s-1)^{1/2}\left\{c_0 + \left[c_1 - \frac{c_0(3+c_0^2)}{6}\right](s-1) + \cdots\right\} \quad \text{as } s \to 1,$$

where $c_0 \simeq 0.63883$ and $c_1 \simeq 0.23625$ are defined by

$$c_0\Phi(c_0) + \phi(c_0) = 0, \qquad c_1 = \frac{2c_0}{5 + c_0^2},$$

fit very well for values of s close to 1. Here, as in the previous example, the asymptotic expansions for large values of s

$$\tilde{x}(s) = \frac{\tilde{y}(s)}{s} \sim -s^{-1/2}\{\log s^2 - 2\log(\log s^2) - \log(8\pi) + \cdots\}^{1/2} \quad \text{as } s \to \infty,$$

TABLE 2

*Estimates of Bayes risk for sequential analysis problem.**

	$z_0 = y_0/s_0^{1/2}$							
$t_0 = 1/s_0$	0	0.5	1.0	1.5	2.0	3.0	4.0	5.0
5.00	.1759							
2.00	.2667							
1.00	.3414							
.50	.3876							
.20	.3765	.3183	.1828					
.10	.3322	.2910	.1934	.9099 (−1)				
.05	.2775	.2468	.1734	.9498 (−1)	.3737 (−1)			
.02	.2072	.1856	.1338	.7816 (−1)	.3705 (−1)			
.01	.1614	.1447	.1048	.6201 (−1)	.3052 (−1)	.3763 (−2)		
5 (−3)	.1234	.1106	.8001 (−1)	.4730 (−1)	.2343 (−1)	.3697 (−2)		
2 (−3)	.8468 (−1)	.7576 (−1)	.5447 (−1)	.3187 (−1)	.1559 (−1)	.2642 (−2)		
1 (−3)	.6284 (−1)	.5611 (−1)	.4011 (−1)	.2321 (−1)	.1117 (−1)	.1858 (−2)	.2094 (−3)	
5 (−4)	.4619 (−1)	.4118 (−1)	.2926 (−1)	.1674 (−1)	.7902 (−2)	.1253 (−2)	.1833 (−3)	
2 (−4)	.3040 (−1)	.2704 (−1)	.1907 (−1)	.1076 (−1)	.4954 (−2)	.7210 (−3)	.1167 (−3)	
1 (−4)	.2199 (−1)	.1953 (−1)	.1371 (−1)	.7660 (−2)	.3466 (−2)	.4698 (−3)	.7563 (−4)	
5 (−5)	.1583 (−1)	.1405 (−1)	.9819 (−2)	.5441 (−2)	.2425 (−2)	.3056 (−3)	.4697 (−4)	.6465 (−5)
2 (−5)	.1020 (−1)	.9032 (−2)	.6286 (−2)	.3451 (−2)	.1512 (−2)	.1743 (−3)	.2409 (−4)	.4874 (−5)
1 (−5)	.7283 (−2)	.6446 (−2)	.4475 (−2)	.2444 (−2)	.1060 (−2)	.1149 (−3)	.1429 (−4)	.3309 (−5)
5 (−6)	.5191 (−2)	.4591 (−2)	.3180 (−2)	.1729 (−2)	.7432 (−3)	.7639 (−4)	.8430 (−5)	.2092 (−5)
2 (−6)	.3308 (−2)	.2924 (−2)	.2021 (−2)	.1094 (−2)	.4660 (−3)	.4517 (−4)	.4185 (−5)	.1071 (−5)
1 (−6)	.2349 (−2)	.2075 (−2)	.1432 (−2)	.7733 (−3)	.3278 (−3)	.3066 (−4)	.2480 (−5)	.6247 (−6)

* The quantity tabulated is the normalized Bayes risk = BR = $E\{d_1(Y(S), S)\} - s_0^{-1}$.

TABLE 3
*Estimates of Bayes expected cost of sampling for sequential analysis problem.**

	$z_0 = y_0/s_0^{1/2}$							
$t_0 = 1/s_0$	0	0.5	1.0	1.5	2.0	3.0	4.0	5.0
5.00	.2501 (−2)							
2.00	.1457 (−1)							
1.00	.4931 (−1)							
.50	.1079							
.20	.1575	.1166	.2389 (−1)					
.10	.1606	.1345	.7380 (−1)	.1243 (−1)				
.05	.1459	.1275	.8386 (−1)	.3804 (−1)	.4990 (−2)			
.02	.1166	.1040	.7382 (−1)	.4141 (−1)	.1738 (−1)			
.01	.9410 (−1)	.8437 (−1)	.6101 (−1)	.3583 (−1)	.1710 (−1)	.5592 (−3)		
5 (−3)	.7391 (−1)	.6636 (−1)	.4824 (−1)	.2873 (−1)	.1429 (−1)	.1805 (−2)		
2 (−3)	.5209 (−1)	.4673 (−1)	.3388 (−1)	.2013 (−1)	.1007 (−1)	.1698 (−2)		
1 (−3)	.3927 (−1)	.3517 (−1)	.2537 (−1)	.1494 (−1)	.7384 (−2)	.1293 (−2)	.5924 (−4)	
5 (−4)	.2925 (−1)	.2614 (−1)	.1874 (−1)	.1090 (−1)	.5293 (−2)	.9086 (−3)	.1063 (−3)	
2 (−4)	.1951 (−1)	.1739 (−1)	.1236 (−1)	.7081 (−2)	.3347 (−2)	.5358 (−3)	.8427 (−4)	
1 (−4)	.1423 (−1)	.1266 (−1)	.8949 (−2)	.5066 (−2)	.2347 (−2)	.3509 (−3)	.5855 (−4)	
5 (−5)	.1031 (−1)	.9163 (−2)	.6442 (−2)	.3610 (−2)	.1642 (−2)	.2278 (−3)	.3787 (−4)	.2344 (−5)
2 (−5)	.6685 (−2)	.5929 (−2)	.4144 (−2)	.2295 (−2)	.1022 (−2)	.1286 (−3)	.2000 (−4)	.3204 (−5)
1 (−5)	.4793 (−2)	.4246 (−2)	.2958 (−2)	.1627 (−2)	.7151 (−3)	.8390 (−4)	.1200 (−4)	.2469 (−5)
5 (−6)	.3426 (−2)	.3033 (−2)	.2106 (−2)	.1152 (−2)	.5006 (−3)	.5512 (−4)	.7106 (−5)	.1669 (−5)
2 (−6)	.2190 (−2)	.1937 (−2)	.1341 (−2)	.7291 (−3)	.3132 (−3)	.3208 (−4)	.3518 (−5)	.8997 (−6)
1 (−6)	.1558 (−2)	.1377 (−2)	.9518 (−3)	.5155 (−3)	.2200 (−3)	.2153 (−4)	.2068 (−5)	.5382 (−6)

* The quantity tabulated is the normalized Bayes expected cost of sampling = ECS = $E(S^{-1}) - s_0^{-1}$.

$$\tilde{z}(s) = \frac{\tilde{y}(s)}{s^{1/2}} \sim -\{\log s^2 - 2\log(\log s^2) - \log(8\pi) + \cdots\}^{1/2} \quad \text{as } s \to \infty,$$

$$\tilde{\beta}(s) = \Phi(\tilde{z}(s)) \sim 2s^{-1}\left\{1 + \frac{2}{\log s^2} - \frac{1}{8}\left[\frac{\log(\log s^2)}{\log s^2}\right]^2 + \cdots\right\} \quad \text{as } s \to \infty$$

are only moderately accurate at $s = 10^6$.

The Bayes expected payoff corresponding to starting at the point (y_0, s_0) in the normalized form of the continuous time version is given by

$$\text{BEP} \equiv -s_0^{1/2}[E\{d_2(Y(S), S)\} - d_2(y_0, s_0)],$$

and the Bayes expected sample size is given by

$$\text{ESS} \equiv s_0[E(S^{-1}) - s_0^{-1}],$$

where S is an optimal stopping rule and $d_2(y, s)$ is given in (1.2). Since the use of $d_2'(y, s) = d_2(y, s) + y$ in place of $d_2(y, s)$ simplified implementation of the truncation modification, the computations for Bayes risk were implemented using

$$\text{BEP} = -s_0^{1/2}[E\{d_2'(Y(S), S)\} - d_2'(y_0, s_0)].$$

BEP and ESS depend only upon the initial values y_0 and s_0 or $z_0 = y_0/s_0^{1/2}$ and $t_0 = 1/s_0$ which has an interpretation as the fraction of the total potential information which is in the prior; some results are presented in Tables 5 and 6. These results reflect the

TABLE 4

Estimates of stopping boundary for one-armed bandit problem.

$t=1/s$	$\tilde{x}(s)=\tilde{y}(s)/s$	$\tilde{z}(s)=\tilde{y}(s)/s^{1/2}$	$\tilde{\beta}(s)=\Phi(\tilde{z}(s))$	$t=1/s$	$\tilde{x}(s)=\tilde{y}(s)/s$	$\tilde{z}(s)=\tilde{y}(s)/s^{1/2}$	$\tilde{\beta}(s)=\Phi(\tilde{z}(s))$
.9995	−.01430	−.0143	.4943	.16	−.4252	−1.0630	.1439
.999	−.02026	−.0203	.4919	.14	−.4176	−1.1161	.1322
.995	−.04524	−.0454	.4819	.12	−.4076	−1.1767	.1197
.99	−.06391	−.0642	.4744	.10	−.3946	−1.2477	.1061
.98	−.09022	−.0911	.4637	.09	−.3866	−1.2886	.09877
.97	−.1103	−.1120	.4554	.08	−.3773	−1.3340	.09110
.96	−.1272	−.1298	.4484	.07	−.3665	−1.3854	.08296
.95	−.1420	−.1456	.4421	.06	−.3540	−1.4450	.07423
.94	−.1554	−.1603	.4363	.05	−.3387	−1.5146	.06494
.93	−.1675	−.1737	.4310	.04	−.3198	−1.5988	.05493
.92	−.1789	−.1865	.4260	.03	−.2956	−1.7069	.04392
.91	−.1894	−.1986	.4213	.02	−.2626	−1.8569	.03166
.90	−.1993	−.2101	.4168	.01	−.2108	−2.1081	.01751
.88	−.2177	−.2321	.4082	9 (−3)	−.2035	−2.1454	.01596
.86	−.2344	−.2528	.4002	8 (−3)	−.1956	−2.1868	.01438
.84	−.2498	−.2725	.3926	7 (−3)	−.1869	−2.2336	.01275
.82	−.2640	−.2915	.3853	6 (−3)	−.1772	−2.2871	.01109
.80	−.2775	−.3103	.3782	5 (−3)	−.1662	−2.3500	.009388
.78	−.2900	−.3284	.3713	4 (−3)	−.1534	−2.4258	.007638
.76	−.3017	−.3461	.3646	3 (−3)	−.1381	−2.5222	.005832
.74	−.3129	−.3637	.3580	2 (−3)	−.1188	−2.6554	.003961
.72	−.3234	−.3811	.3516	1 (−3)	−.09090	−2.8744	.002024
.70	−.3333	−.3984	.3452	9 (−4)	−.08720	−2.9067	.001826
.68	−.3428	−.4157	.3388	8 (−4)	−.08321	−2.9420	.001630
.66	−.3517	−.4329	.3325	7 (−4)	−.07892	−2.9829	.001428
.64	−.3602	−.4503	.3262	6 (−4)	−.07420	−3.0293	.001226
.62	−.3683	−.4678	.3200	5 (−4)	−.06895	−3.0836	.001023
.60	−.3760	−.4854	.3137	4 (−4)	−.06297	−3.1487	.8201 (−3)
.58	−.3832	−.5032	.3074	3 (−4)	−.05597	−3.2316	.6155 (−3)
.56	−.3901	−.5212	.3011	2 (−4)	−.04730	−3.3445	.4122 (−3)
.54	−.3965	−.5396	.2947	1 (−4)	−.03533	−3.5327	.2057 (−3)
.52	−.4026	−.5583	.2883	9 (−5)	−.03378	−3.5605	.1851 (−3)
.50	−.4086	−.5778	.2817	8 (−5)	−.03213	−3.5917	.1643 (−3)
.48	−.4138	−.5973	.2752	7 (−5)	−.03034	−3.6263	.1438 (−3)
.46	−.4187	−.6173	.2685	6 (−5)	−.02838	−3.6640	.1242 (−3)
.44	−.4231	−.6379	.2618	5 (−5)	−.02625	−3.7116	.1030 (−3)
.42	−.4272	−.6592	.2549	4 (−5)	−.02383	−3.7680	.8230 (−4)
.40	−.4309	−.6813	.2479	3 (−5)	−.02103	−3.8397	.6162 (−4)
.38	−.4340	−.7041	.2407	2 (−5)	−.01762	−3.9388	.4095 (−4)
.36	−.4367	−.7278	.2334	1 (−5)	−.01297	−4.1013	.2055 (−4)
.34	−.4388	−.7525	.2259	9 (−6)	−.01238	−4.1257	.1849 (−4)
.32	−.4405	−.7787	.2181	8 (−6)	−.01175	−4.1532	.1640 (−4)
.30	−.4415	−.8061	.2101	7 (−6)	−.01107	−4.1839	.1434 (−4)
.28	−.4419	−.8351	.2018	6 (−6)	−.01033	−4.2190	.1228 (−4)
.26	−.4416	−.8660	.1932	5 (−6)	−.009526	−4.2602	.1022 (−4)
.24	−.4404	−.8990	.1843	4 (−6)	−.008620	−4.3101	.8163 (−5)
.22	−.4383	−.9345	.1750	3 (−6)	−.007570	−4.3706	.6200 (−5)
.20	−.4354	−.9736	.1651	2 (−6)	−.006307	−4.4597	.4107 (−5)
.18	−.4311	−1.0160	.1548	1 (−6)	−.004608	−4.6077	.2038 (−5)

TABLE 5

*Estimates of Bayes expected payoff for one-armed bandit problem.**

	$z_0 = y_0/s_0^{1/2}$						
$t_0 = 1/s_0$	0	−0.5	−1.0	−1.5	−2.0	−3.0	−4.0
.50	.1774	.0035					
.20	.1059 (1)	.2390					
.10	.2769 (1)	.8971	.0839				
.05	.6420 (1)	.2479 (1)	.5175				
.02	.1784 (2)	.7773 (1)	.2375 (1)	.3003			
.01	.3728 (2)	.1708 (2)	.5961 (1)	.1249	.0268		
5 (−3)	.7658 (2)	.3618 (2)	.1362 (2)	.3592 (1)	.3915		
2 (−3)	.1953 (3)	.9447 (2)	.3758 (2)	.1150 (2)	.2219 (1)		
1 (−3)	.3940 (3)	.1925 (3)	.7838 (2)	.2538 (2)	.5867 (1)		
5 (−4)	.7920 (3)	.3892 (3)	.1607 (3)	.5387 (2)	.1370 (2)	.0233	
2 (−4)	.1987 (4)	.9812 (3)	.4093 (3)	.1406 (3)	.3824 (2)	.6137	
1 (−4)	.3981 (4)	.1969 (4)	.8245 (3)	.2861 (3)	.7992 (2)	.2089 (1)	
5 (−5)	.7969 (4)	.3946 (4)	.1657 (4)	.5783 (3)	.1640 (3)	.5440 (1)	
2 (−5)	.1994 (5)	.9878 (4)	.4154 (4)	.1456 (4)	.4176 (3)	.1628 (2)	
1 (−5)	.3988 (5)	.1977 (5)	.8319 (4)	.2920 (4)	.8409 (3)	.3482 (2)	.0493
5 (−6)	.7977 (5)	.3955 (5)	.1665 (5)	.5850 (4)	.1689 (4)	.7247 (2)	.4370
2 (−6)	.1995 (6)	.9890 (5)	.4165 (5)	.1464 (5)	.4232 (4)	.1860 (3)	.2117 (1)
1 (−6)	.3989 (6)	.1978 (6)	.8330 (5)	.2929 (5)	.8477 (4)	.3765 (3)	.5354 (1)

* The quantity tabulated is the normalized Bayes expected payoff = BEP = $-s_0^{1/2}[E\{d_2'(Y(S), S)\} - d_2'(y_0, s_0)]$.

TABLE 6

*Estimates of Bayes expected sample size for one-armed bandit problem.**

	$z_0 = y_0/s_0^{1/2}$						
$t_0 = 1/s_0$	0	−0.5	−1.0	−1.5	−2.0	−3.0	−4.0
.50	.5786	.0835					
.20	.2217 (1)	.1047 (1)					
.10	.4851 (1)	.2637 (1)	.7097				
.05	.1001 (2)	.5782 (1)	.2256 (1)				
.02	.2528 (2)	.1513 (2)	.6942 (1)	.1884 (1)			
.01	.5052 (2)	.3064 (2)	.1480 (2)	.5065 (1)	.5210		
5 (−3)	.1008 (3)	.6157 (2)	.3055 (2)	.1154 (2)	.2588 (1)		
2 (−3)	.2512 (3)	.1542 (3)	.7795 (2)	.3125 (2)	.9084 (1)		
1 (−3)	.5016 (3)	.3086 (3)	.1571 (3)	.6434 (2)	.2018 (2)		
5 (−4)	.1002 (4)	.6172 (3)	.3155 (3)	.1308 (3)	.4257 (2)	.5929	
2 (−4)	.2502 (4)	.1543 (4)	.7913 (3)	.3308 (3)	.1103 (3)	.4387 (1)	
1 (−4)	.5003 (4)	.3086 (4)	.1584 (4)	.6643 (3)	.2236 (3)	.1094 (2)	
5 (−5)	.1000 (5)	.6172 (4)	.3171 (4)	.1332 (4)	.4507 (3)	.2417 (2)	
2 (−5)	.2501 (5)	.1543 (5)	.7931 (4)	.3336 (4)	.1132 (4)	.6433 (2)	
1 (−5)	.5001 (5)	.3086 (5)	.1586 (5)	.6677 (4)	.2270 (4)	.1314 (3)	.1032 (1)
5 (−6)	.1000 (6)	.6171 (5)	.3173 (5)	.1336 (5)	.4544 (4)	.2659 (3)	.4101 (1)
2 (−6)	.2500 (6)	.1543 (6)	.7935 (5)	.3340 (5)	.1137 (5)	.6696 (3)	.1334 (2)
1 (−6)	.5000 (6)	.3086 (6)	.1587 (6)	.6681 (5)	.2274 (5)	.1344 (4)	.2907 (2)

* The quantity tabulated is the normalized Bayes expected sample size = ESS = $s_0[E(S^{-1}) - s_0^{-1}]$.

form of the asymptotic expansions

$$\text{BEP} \sim s_0\{\phi(z_0) + z_0\Phi(z_0)\} \quad \text{as } s_0 \to \infty,$$

$$\text{ESS} \sim s_0\Phi(z_0) \quad \text{as } s_0 \to \infty,$$

provided in [13].

The computation for these two examples required 31.4 and 184.5 seconds of CPU time, respectively, on the 12-megabyte Amdahl 470 $V/8$ at the University of British Columbia. The large difference is due to the one-sided nature of the continuation set for the one-armed bandit problem; such problems typically require substantially more computation than symmetric problems even if the truncation procedure is implemented.

7. Conclusions. The main approach of this paper is to use the very simple backward induction involving the discrete time process with Bernoulli increments. The accuracy of this is enhanced by the continuity correction. Numerical experiments with a number of examples indicate that coarse intervals δ often yield surprisingly good results, and dividing δ by a factor of 4, which increases the calculation effort by a factor of 8, tends to increase the accuracy of the resulting estimates of $\tilde{y}(s)$ by a factor closer to 4 than to 3. This implies that doubling the accuracy requires approximately three times as much computation when the continuity correction is used.

Other enhancements can sometimes be obtained by making use of truncation when one-sided boundaries are involved or monotonicity and symmetry when appropriate. Changing the interval between successive values of s_i can be used to calculate results for large s.

A general purpose program designed to perform these computations for a wide variety of such stopping problems should be capable of taking advantage of the special features of particular problems which might allow the computational effort to be substantially reduced. One should anticipate that special versions may occasionally require intelligent intervention to avoid numerical difficulties such as underflows, overflows and round-off errors. For example, rather careful programming was required to obtain a good approximation to the optimal stopping boundary for the problem with stopping cost function

$$d(y, s) = \min(y, 0) \exp\left(-\frac{1}{s}\right) \quad \text{for } s \geqq 0,$$

discussed in [2].

These techniques are investigated in greater detail in [11] where five additional applications are considered. Two of these are problems with known solutions which allow a careful evaluation of the accuracy of the computed approximations. There, as in this paper, the focus is on obtaining numerical solutions for continuous time versions of these optimal stopping problems. The question of whether these numerical solutions, when adjusted by the appropriate continuity correction, yield accurate approximations to the solutions of the original, discrete time version must be considered on a problem-by-problem basis. Details are presented for a problem involving Bernoulli data in [18] and for a problem involving normal data in [10]; in both cases, accurate approximations to the solutions of the discrete time versions resulted.

Acknowledgment. The second author gratefully acknowledges the first author as a constant source of stimulation and inspiration throughout the lengthy period of time over which the techniques reported here have evolved, and would like to dedicate his work on this project to Herman Chernoff on the recent occasion of his sixtieth birthday.

REFERENCES

[1] J. BATHER, *Bayes procedures for deciding the sign of a normal mean*, Proc. Cambridge Philos. Soc., 58 (1962), pp. 599-620.

[2] ———, *Optimal stopping of Brownian motion: a comparison technique*, Recent Advances in Statistics, Papers in Honor of Herman Chernoff on his Sixtieth Birthday, M. H. Rizvi, J. Rustagi and D. Siegmund, eds., Academic Press, New York, 1983, pp. 19-50.

[3] J. BREAKWELL AND H. CHERNOFF, *Sequential tests for the mean of a normal distribution* II, Ann. Math. Statist., 35 (1964), pp. 162-173.

[4] H. CHERNOFF, *Sequential tests for the mean of a normal distribution*, Proc. 4th Berkeley Symp., 1 (1961), pp. 79-91.

[5] ———, *Sequential tests for the mean of a normal distribution* III, Ann. Math. Statist., 36 (1965), pp. 28-54.

[6] ———, *Sequential tests for the mean of a normal distribution* IV, Ann. Math. Statist., 36 (1965), pp. 55-68.

[7] ———, *Sequential models for clinical trials*, Proc. 5th Berkeley Symp., 4 (1967), pp. 805-812.

[8] ———, *Sequential Analysis and Optimal Design*, CBMS Regional Conference Series in Applied Mathematics 8, Society for Industrial and Applied Mathematics, Philadelphia, 1972.

[9] H. CHERNOFF AND A. J. PETKAU, *An optimal stopping problem for sums of dichotomous random variables*, Ann. Prob., 4 (1976), pp. 875-889.

[10] ———, *Sequential medical trials involving paired data*, Biometrika, 68 (1981), pp. 119-132.

[11] ———, *Numerical methods for Bayes sequential decision problems*, Technical Report 83-26, Institute of Applied Mathematics and Statistics, University of British Columbia, Vancouver, B.C., 1983 and Technical Report 34, Statistics Center, Massachusetts Institute of Technology, Cambridge, MA, 1984.

[12] ———, *Sequential medical trials with ethical cost*, 1984, to appear.

[13] H. CHERNOFF AND S. N. RAY, *A Bayes sequential sampling inspection plan*, Ann. Math. Statist., 36 (1965), pp. 1387-1407.

[14] W. FELLER, *An Introduction to Probability Theory and Its Applications*, Vol. 1, 3rd edition, John Wiley, New York, 1968.

[15] D. V. LINDLEY, *Dynamic programming and decision theory*, Applied Statistics, 10 (1961), pp. 39-51.

[16] D. V. LINDLEY AND B. N. BARNETT, *Sequential sampling: two decision problems with linear losses for binomial and normal random variables*, Biometrika, 52 (1965), pp. 507-532.

[17] S. MORIGUTI AND H. ROBBINS, *A Bayes test of* $p \leqq \frac{1}{2}$ *versus* $p > \frac{1}{2}$, Rep. Statist. Appl. Res., Un. Japan Sci. Engrs., 9 (1962), pp. 39-60.

[18] A. J. PETKAU, *Sequential medical trials for comparing an experimental with a standard treatment*, J. Amer. Statist. Assoc., 73 (1978), pp. 328-338.

SIAM J. SCI. STAT. COMPUT.
Vol. 7, No. 1, January 1986

004

COMPUTATIONS OF MIXTURES OF DIRICHLET PROCESSES*

LYNN KUO†

Abstract. The computation of Bayes estimators based on mixtures of Dirichlet processes is treated in this article. These estimators may be written as ratios of two multidimensional integrals, each of which may be decomposed into a weighted average of products of one-dimensional integrals. An importance sampling Monte Carlo method is proposed to approximate each of the weighted averages. A prior error bound for each of the Monte Carlo estimators and a posterior error bound for the ratio are developed to measure the efficiency of the Monte Carlo method. Jackknife and random group error estimates are also considered. Two examples are given which illustrate the computation of the Bayes estimators.

Key words. Monte Carlo method, multivariable functions, mixtures of Dirichlet processes, Dirichlet process, empirical Bayes estimation

1. Introduction. In this article, we are concerned with the computational aspects of a Bayes estimator which arises in many nonparametric Bayesian studies. Let us consider the following two kinds of problems: empirical Bayes problems and density estimation. In empirical Bayes problems, the unobservable parameters $\{\theta_i\}$, $i = 1, \cdots, n$, are assumed to be independent from an unknown distribution G. Associated with each θ_i, a random variable X_i is observed with known density $f(x_i|\theta_i)$. The unknown parameters $\{\theta_i\}$ and the mixing distribution G can be estimated from the observed data. In density estimation, the density functions are assumed expressible as a convolution of a known kernel and an unknown distribution function G. Various approaches to estimating this density function have been developed. See, for example, Tapia and Thompson (1978) and Prakasa Rao (1983).

In a Bayesian approach to each of these problems, a Ferguson's Dirichlet prior (1973) can be put on G. It is shown in Antoniak (1974) that the posterior distribution is a mixture of Dirichlet processes. The Bayes estimators of $\theta_1, \cdots, \theta_n$ in the first problem are derived by Kuo (1980). The Bayes estimators of the mixing distribution G in the first problem and the density function in the second problem are derived by Lo (1978), (1984). All of the estimators may be written as ratios of two n-dimensional integrals. The biggest difficulty in applications so far is to evaluate these multidimensional integrals. The difficulty arises from the high dimensionality and complexity of the integrals, and the fact that the integrand is peaked in a small region of the parameter space. The usual Monte Carlo method does not work well. The objective of this article is to give a satisfactory method for evaluating the Bayes estimators.

In § 2, we study the numerator and denominator of the Bayes rule separately. As shown in Kuo (1980), each of them can be expressed as a weighted average of products of one-dimensional integrals. Both weighted averages can be represented by a multivariable function Φ given below in (1). This Φ, which is more general than the multivariable function considered by Hammersley and Handscomb (1960) or (1964, p. 143), is the main object studied in § 2. These results may have applications to other multivariable problems. We generate a mean of N independent unbiased estimates of Φ, where each of the unbiased estimates is produced by importance sampling (each summand in the weighted average is sampled according to its weight). As a result of the law of large numbers, the mean converges to the value of the integral as $N \to \infty$. To measure the

* Received by the editors February 3, 1982, and in revised form October 15, 1984.
† Department of Applied Mathematics and Statistics, State University of New York, Stony Brook, New York 11794.

efficiency of this Monte Carlo method, a prior error bound and a posterior error bound, which evaluate the standard deviation of the mean, are developed.

In § 3, a modification of the importance sampling method is applied to approximate each of the multidimensional integrals. The ratio of these two approximations is used as an estimate of the Bayes estimator. The modification is tailored to account for the "doubling up" of the θ's in the mixing distribution. A posterior error bound, which measures the standard deviation of the estimated Bayes rule, is developed from large sample theory. The posterior error bound depends on the integrals (the values we propose to evaluate). These integrals can be estimated by the Monte Carlo estimators. It is difficult to assess the accuracy of this estimated error bound. Therefore, jackknife and random group methods are also used to estimate the standard deviation of the estimated Bayes rules. Finally, two numerical illustrations are given based on the baseball data of Efron and Morris (1975) and the Portsmouth Naval Shipyard data of Martz and Lian (1974).

2. Multivariable problems. Let us consider the following multivariable function Φ for given probability vectors $\mathbf{p}_1, \mathbf{p}_2, \cdots, \mathbf{p}_n$ with $\mathbf{p}_i = (p_{i1}, \cdots, p_{ii})$ and $\sum_{j=1}^{i} p_{ij} = 1$.

$$\Phi(\mathbf{p}_1, \cdots, \mathbf{p}_n) = \sum_{\delta \in Z^*} p_{1\delta_1} \cdot p_{2\delta_2} \cdot \cdots \cdot p_{n\delta_n} \phi(\delta_1, \delta_2, \cdots, \delta_n) \tag{1}$$

where $\phi(\delta_1, \cdots, \delta_n)$ is a real-valued function defined on $Z_1 \times Z_2 \times \cdots \times Z_n \equiv Z^*$ with $Z_k = \{1, 2, \cdots, k\}$. Automatically, we have $\delta_1 = 1$ and $p_{1\delta_1} = 1$. It is more and more difficult to evaluate Φ as n increases, since the number of summands is $n!$. For example when $n = 20$, there are $20! \approx 2.43 \times 10^{18}$ terms to be added. Therefore, it is desirable to devise an algorithm which generates a tiny fraction of these summands and enables us to approximate Φ satisfactorily. An importance sampling Monte Carlo method is proposed here to approximate Φ. It can be seen in Theorem 1 and Corollary 1 that this method is appropriate for evaluating Φ. The Monte Carlo method and its statistical analysis are given in the following.

(i) *Monte Carlo method.* (a) We first generate independent integer random variables $\zeta_1, \zeta_2, \cdots, \zeta_n$ taking values in $Z_1, Z_2, \cdots, Z_n$ respectively with probabilities $P(\zeta_i = j) = p_{ij}$, and $\sum_{j=1}^{i} p_{ij} = 1$ for $i = 1, \cdots, n$. Therefore

$$P(\zeta_1 = \delta_1, \zeta_2 = \delta_2, \cdots, \zeta_n = \delta_n) = p_{1\delta_1} \cdot p_{2\delta_2} \cdot p_{3\delta_3} \cdot \cdots \cdot p_{n\delta_n}$$

and

$$E\phi(\zeta_1, \cdots, \zeta_n) = \Phi(\mathbf{p}_1, \mathbf{p}_2, \cdots, \mathbf{p}_n). \tag{2}$$

(b) We repeat step (a) independently N times. Let $\boldsymbol{\zeta} = (\zeta_1, \cdots, \zeta_n)$. We generate N mutually independent copies of $\boldsymbol{\zeta}$ to obtain $\boldsymbol{\zeta}(1), \boldsymbol{\zeta}(2), \cdots, \boldsymbol{\zeta}(N)$. We evaluate $\phi(\boldsymbol{\zeta}(1)), \phi(\boldsymbol{\zeta}(2)), \cdots, \phi(\boldsymbol{\zeta}(N))$, and compute the mean of the resulting function values

$$\sum_{l=1}^{N} \phi(\boldsymbol{\zeta}(l)) \Big/ N \tag{3}$$

and use it to approximate (1).

(ii) *Statistical analysis.* Since $\phi(\boldsymbol{\zeta}(l))$ are independent for $l = 1, \cdots, N$, it follows from (2) and independence that $\sum_{l=1}^{N} \phi(\boldsymbol{\zeta}(l))/N$ is an unbiased estimator of $\Phi(\mathbf{p}_1, \cdots, \mathbf{p}_n)$ with variance

$$\sigma^2 = \frac{1}{N} \operatorname{Var} \phi(\boldsymbol{\zeta}). \tag{4}$$

Therefore, this procedure is appropriate for approximating $\Phi(\mathbf{p}_1, \cdots, \mathbf{p}_n)$ provided σ

is small enough. Moreover, by the strong law of large numbers, (3) converges almost surely to $\Phi(\mathbf{p}_1, \cdots, \mathbf{p}_n)$ as $N \to \infty$. Finding a prior standard error bound is equivalent to finding an upper bound on σ in (4) which definitely depends on how well-behaved the function ϕ is.

For each i, $1 \leqq i \leqq n$, let us define the regularity quantity Δ_i of ϕ as follows:

$$\Delta_i = \max_{\substack{\text{all } \delta_j \\ (j \neq i)}} \max_{\substack{1 \leqq i_1 \leqq i \\ 1 \leqq i_2 \leqq i}} [\phi(\delta_1, \cdots, \delta_{i-1}, i_1, \delta_{i+1}, \cdots, \delta_n) - \phi(\delta_1, \cdots, \delta_{i-1}, i_2, \delta_{i+1}, \cdots, \delta_n)]^2. \tag{5}$$

Then, based on Hammersley's result (1960), we have the following theorem which gives an upper bound for the variance of $\phi(\zeta)$. It is proved in Appendix I.

THEOREM 1. *Let $\phi(\delta_1, \cdots, \delta_n)$ be a real-valued function defined on Z^* and let $\{\zeta_1, \zeta_2, \cdots, \zeta_n\}$ be chosen independently with probabilities $P(\zeta_i = j) = p_{ij}$, $\sum_{j=1}^{i} p_{ij} = 1$ for all $i = 1, \cdots, n$. Then the variance of $\phi(\zeta_1, \cdots, \zeta_n) \leqq \frac{1}{4}[\sum_{i=2}^{n} \Delta_i]$.*

Using the result of Theorem 1 and the fact that the variance of $\phi(\zeta)$ is bounded above by one quarter of the square of its range (this follows from Lemma 1 in Appendix I), we can obtain an upper bound on the variance of the Monte Carlo estimator even before doing the calculation. The following corollary follows straightforwardly; therefore it is only stated.

COROLLARY 1. *Let σ denote the prior standard error of the Monte Carlo estimator (3). Then*

$$\sigma \leqq \min\left(\left(\sum_{i=2}^{n} \Delta_i\right)^{1/2} \Big/ (4N)^{1/2}, \left(\max_{\delta \in Z^*} (\delta_1, \cdots, \delta_n) - \min_{\delta \in Z^*} (\delta_1, \cdots, \delta_n)\right) \Big/ (4N)^{1/2}\right). \tag{6}$$

Remark. Easy examples can be constructed to show neither of the two upper bounds in square brackets is uniformly better than the other one.

As a consequence of Corollary 1 and the central limit theorem, we can approximate the number of iterations N needed in this Monte Carlo method to achieve a certain accuracy. For example, we could achieve an accuracy within ε error with probability at least .68 by setting N equal to $\min[\sum_{i=1}^{n} \Delta_i, (\text{range } \phi(\zeta))^2]/4\varepsilon^2$.

When the function $\phi(\delta_1, \cdots, \delta_n)$ is complicated, it might be difficult to obtain the prior error bound. Even when it is available, the bound may be too conservative. Instead, one may use the sample (posterior) standard error $\hat{\sigma} = [\sum_{l=1}^{N} (\phi(\zeta(l) - \bar{\phi})^2)/N]^{1/2}$, where $\bar{\phi} = \sum_{l=1}^{N} (\phi(\zeta(l)))/N$.

3. Bayes estimators. It is shown in Kuo (1980) that the proper Bayes estimators with Dirichlet prior for the empirical Bayes problems are given as follows:

$$\hat{\theta}_k = \frac{\displaystyle\int_{R_n} \cdots \int \left[\theta_k \prod_{i=1}^{n} f_i(x_i|\theta_i)\right] \prod_{i=1}^{n} \frac{(\alpha + \sum_{j=1}^{i-1} \delta_{\theta_j})}{\alpha(\mathbb{R}) + i - 1}(d\theta_i)}{\displaystyle\int_{R_n} \cdots \int \left[\prod_{i=1}^{n} f_i(x_i|\theta_i)\right] \prod_{i=1}^{n} \frac{(\alpha + \sum_{j=1}^{i-1} \delta_{\theta_j})}{\alpha(\mathbb{R}) + i - 1}(d\theta_i)}, \tag{7}$$

where f_i is a density, α is any finite measure and δ_{θ_j} is the measure concentrated at θ_j with mass 1.

The formula (7) looks deceptively simple. Since each of the integrals is n-dimensional, a Monte Carlo method of approximation seems appropriate. However, the complexity of the product measure

$$\prod_{i=1}^{n} \frac{(\alpha + \sum_{j=1}^{i-1} \delta_{\theta_j})(d\theta_i)}{\alpha(\mathbb{R}) + i - 1}$$

creates great problems. A straightforward Monte Carlo method suggests that we take

$$\theta_1 \text{ from } \frac{\alpha(\cdot)}{\alpha(\mathbb{R})},$$

$$\theta_2 \text{ from } \frac{(\alpha+\delta_{\theta_1})(\cdot)}{\alpha(\mathbb{R})+1},$$

$$\theta_3 \text{ from } \frac{(\alpha+\sum_{j=1}^{2}\delta_{\theta_j})(\cdot)}{\alpha(\mathbb{R})+2},$$

etc., and evaluate the integrand at these θ's, then repeat this process independently N times (N large), and use the mean of these N resulting values to approximate the n-dimensional integral. This method does not work well, because the integrand, being a product of n densities (or a function of a product of n densities) is peaked at a very small region of the $\boldsymbol{\theta}$ depending on $\mathbf{x}$. Since $\boldsymbol{\theta}$ is chosen independently of the $\mathbf{x}$, only rarely does $\boldsymbol{\theta}$ fall in this region. The resulting mean value has undesirably big variance, since it is dominated by a few relatively large values. We propose another Monte Carlo method which avoids these difficulties.

If we replace $\alpha(\theta)$ by $MF_0(\theta)$ where F_0 is a probability measure, and $M=\alpha(\mathbb{R})$, then it can be shown as in Kuo (1980) that the numerator and denominator of (7) are each of the form (1) with

$$(p_{i1},\cdots,p_{ii-1},p_{ii})=\left(\frac{1}{M+i-1},\cdots,\frac{1}{M+i-1},\frac{M}{M+i-1}\right)$$

and with many of the summands of (1) equal. For example,

$$\phi(1,1,1,4,5,\cdots,n)=\phi(1,1,2,4,5,\cdots,n), \quad \text{and}$$

$$\phi(1,1,1,\cdots,1)=\phi(1,1,\delta_3,\delta_4,\cdots,\delta_n) \quad \text{for all } \delta_3\in Z_2,\ \delta_4\in Z_3,\cdots,\delta_n\in Z_{n-1},$$

where $\phi(\delta_1,\cdots,\delta_n)$ is a product of single integrals. To account for this "doubling up," we apply a modification of the Monte Carlo method developed in § 2 with δ_i generated sequentially as a function of $\delta_1,\delta_2,\cdots,\delta_{i-1}$ and i.

(i) *Monte Carlo method.* (A) For the lth iteration, $l=1,\cdots,N$, we do the following steps (a)–(c):

(a) We generate $\delta_1,\delta_2,\cdots,\delta_n$ sequentially by the following steps 1 through n, where the probabilities are independently generated for steps, 1 through n, and the step i, for all $1\leqq i\leqq n$ may be written as follows:

$$\text{with probability } \frac{M}{M+i-1}, \text{ choose } \delta_i=i;$$

$$\text{with probability } \frac{1}{M+i-1}, \text{ choose } \delta_i=\delta_{i-1};$$

$$\vdots$$

$$\text{with probability } \frac{1}{M+i-1}, \text{ choose } \delta_i=\delta_1.$$

(b) After the above vector $\boldsymbol{\delta}=(\delta_1,\delta_2,\delta_3,\cdots,\delta_n)$ is selected, the index set $\{1,\cdots,n\}$ is partitioned into a disjoint union of subsets. Each subset consists of indices i such that δ_i equals a particular integer value. Let π denotes this partition and K its

nonvoid subsets. Then evaluate

$$\prod_{K\subset\pi}\int\prod_{i\in K}f_i(x_i|\theta_i)F_0(d\theta). \tag{8}$$

Let (8) be denoted by V_l. (Note that these are one-dimensional integrals which can be evaluated by standard numerical integration methods. It is often desirable to have F_0 be the conjugate prior of $\prod_{i\in K}f_i(x_i|\theta)$, since then the integral can be evaluated immediately.)

(c) With the same partition π as in (b), pick the subset of π containing the index k. Let it be denoted by K_k. Then we evaluate the product of two terms, where the first term is $\int\theta\prod_{i\in K_k}f_i(x_i|\theta)F_0(d\theta)$, and the second term is the product over the rest of the subsets of π as in (b). Therefore, we have:

$$\int\theta\prod_{i\in K_k}f_i(x_i|\theta)F_0(d\theta)\cdot\prod_{\substack{K\subset\pi\\K\neq K_k}}\int\prod_{i\in K}f_i(x_i|\theta)F_0(d\theta). \tag{9}$$

Let (9) be denoted by U_l.

(B) Repeat (A) independently N times. Then compute:

$$\bar{U}/\bar{V},\quad\text{where }\bar{U}=\sum_{l=1}^{N}U_l/N\text{ and }\bar{V}=\sum_{l=1}^{N}V_l/N, \tag{10}$$

and use $\bar{U}/\bar{V}$ to approximate (7).

(ii) *Statistical analysis.* The prior error bound for each of the Monte Carlo estimators $\bar{U}$ and $\bar{V}$ can be obtained as in § 2(ii). The prior error bound of the ratio depends on the magnitude of the denominator. It can be seen from the following theorem that a sharp prior error bound (an upper bound on (11)) is hard to obtain. Nevertheless, a posterior error bound can be computed along with the Monte Carlo simulation. We need the following theorem which gives the asymptotic distribution of the estimator $\bar{U}/\bar{V}$. The proof is omitted, since the result follows from a straightforward application of large sample theory (see, for example, § 6a. 2 in C. R. Rao (1973)).

THEOREM 2. *Let* $(U_1, V_1), \cdots, (U_N, V_N)$ *be independent, identically distributed 2-dimensional vectors with mean vector* (u_U, u_V), $u_V\neq 0$, *and covariance matrix*

$$\begin{pmatrix}\sigma_U^2 & \sigma_{UV}\\ \sigma_{UV} & \sigma_V^2\end{pmatrix},\qquad \sigma_U^2<\infty,\ \sigma_V^2<\infty.$$

Let $\bar{U}=\sum_{l=1}^{N}U_l/N$ *and* $\bar{V}=\sum_{l=1}^{N}V_l/N$. *Then* $\sqrt{N}(\bar{U}/\bar{V}-u_U/u_V)$ *converges in distribution to a normal random variable with zero mean and variance*

$$\sigma_U^2/u_V^2-2u_U\sigma_{UV}/u_V^3+u_V^2\sigma_V^2/u_V^4.$$

Note that u_U/u_V is the Bayes estimator (7) which we seek to evaluate. As an application of Theorem 2, we can derive the standard error of the Monte Carlo estimator proposed in (10). It is

$$(\sigma_U^2/u_V^2-2u_U\sigma_{UV}/u_V^3+u_U^2\sigma_V^2/u_V^4)^{1/2}/\sqrt{N}, \tag{11}$$

which is a function of unknown quantities. Nevertheless (11) can be approximated by the empirical values generated by the Monte Carlo method. Therefore, the estimated posterior standard error, denoted by $\hat{e}_{\bar{U}/\bar{V}}$, is given by

$$(S_U^2/\bar{V}^2-2\bar{U}S_{UV}/\bar{V}^3+\bar{U}^2S_V^2/\bar{V}^4)^{1/2}/\sqrt{N} \tag{12}$$

where $S_U^2 = \sum_{l=1}^N (U_l - \bar{U})^2/N$, $S_{UV} = \sum_{l=1}^N (U_l - \bar{U})(V_l - \bar{V})/N$, and $S_V^2 = \sum_{l=1}^N (V_l - \bar{V})^2/N$.

If $u_U \sigma_{UV}/u_V^3 < 0$, where $u_U \sigma_{UV}/u_V^3$ can be approximated by $\bar{U}S_{UV}/\bar{V}^3$, then (11) suggests that we can improve the standard error by taking $\sigma_{UV} = 0$. Therefore, in this case, it is worthwhile modifying the Monte Carlo method in § 3(i) as follows. Replace the first phrase in step A(c) by (c'):

(c') Repeat step (a) independently to obtain another $\boldsymbol{\delta}$ and π. Then evaluate U_l according to this $\boldsymbol{\delta}$ and π.

In short, we can generate U_l and V_l independently at each iteration. The reduction of variance is done at the expense of requiring more computing time to generate a new set of $\boldsymbol{\delta}$ and π.

If $u_U \sigma_{UV}/u_V^3 > 0$, then we can see from (11) that the Monte Carlo method proposed in § 3 is superior to the above method, since the variance and computing time are reduced simultaneously.

(iii) *Numerical examples.* The data for the first example given by Efron and Morris (1975) consists of the batting averages of the first 45 at bats in the 1970 baseball season for each of 18 major league players. These batting averages are denoted by $\{Y_k\}$, $k = 1, \cdots, 18$, $0 < Y_k < 1$. It is assumed that $\{45 Y_k\}$ are distributed independently according to a binomial distribution with sample sizes 45 and parameters P_k, where P_k, the true season batting average for player k, is to be predicted. Since the variance of Y_k depends on the mean, an arc-sine transformation for stabilizing the variance of a binomial random variable is used:

$$X_k = T_{45}(Y_k), \quad \text{where } T_{45}(Y) = \sqrt{45} \arcsin(2Y - 1). \tag{13}$$

Then we have that the data $\{X_k\}$ are independently distributed, approximately normal with mean θ_k and variance 1, where $\theta_k = T_{45}(P_k)$. We intend to predict θ_k.

If we assume θ_k are taken independently from an unknown distribution G which is chosen by the Dirichlet process, then the Bayes estimator of θ_k under squared error loss is shown to be (7) in Kuo (1980) with

$$f_i(x|\theta) = \frac{1}{\sqrt{2\pi}} e^{-(x-\theta)^2/2}$$

for all $i = 1, \cdots, 18$, and $\alpha(\theta) = MF_0(\theta)$. M and $F_0(\theta)$ are selected by the statistician to represent strength of prior belief and the prior guess of G respectively. To simplify the evaluation of (8) and (9), we are choosing $F_0(\theta)$ to be the normal distribution $\mathcal{N}(a, b^2)$ which is a conjugate prior for the normal densities. We follow the Monte Carlo procedure proposed in § 3(i) to approximate $\hat{\theta}_k$. Let $\tilde{\theta}_k$ denote the Monte Carlo estimator (10). An inverse transformation of (13) is needed to predict P_k:

$$\tilde{P}_k = T_{45}^{-1}(\tilde{\theta}_k) = [\sin(\tilde{\theta}_k/\sqrt{45}) + 1]/2.$$

The posterior standard error for $\tilde{P}_k$, denoted by $\tilde{e}(P_k)$, should be adjusted as follows:

$$\tilde{e}(P_k) = \tilde{e}_k \cdot (T^{-1})'(\tilde{\theta}_k) = \tilde{e}_k \cdot \cos(\tilde{\theta}_k/\sqrt{45})/(2\sqrt{45}),$$

where $\tilde{e}_k$ is the posterior standard error for $\tilde{\theta}_k$ computed from (12).

There are also sample reuse methods to estimate variances as pointed out by a referee. The jackknife (see Miller 1974) and random group method (see Raj (1968, pp. 194–195) are also used to estimate the standard deviation of the Monte Carlo estimator (10). Let the standard deviations be denoted by SDJACK and SDRG respectively, and the corresponding ones for $\tilde{P}$ be denoted by SDJACKP and SDRGP.

In Table 1, M, a, b^2 are arbitrarily chosen to be 1, $\bar{x}$, and $\sum (x_i - \bar{x})^2/18$. The number of iterations in the Monte Carlo method N is 2000. The jackknife variance estimator for $\bar{U}/\bar{V}$ is computed by deleting a group of 50 iterations each time. The random group variance estimator is computed from 40 groups of 50 iterations in each group. See Appendix II for a more detailed description of these variance estimators. Observe that $\tilde{e}$, SDJACK, and SDRG are quite comparable in all the cases.

In Table 2, M, a, b^2 are chosen to be 500, -3.317 and $(0.514)^2$ to simulate the estimates that Efron and Morris obtained. The number of iterations is 200. SDJACK

TABLE 1
Monte Carlo estimators for batting averages.

($a = -3.317$, $b = 1$, $M = 1$, $N = 2000$, $g = 40$)

k	x_k	$\tilde{\theta}_k$	$\tilde{e}_k$	SDJACK$_k$	SDRG$_k$	$\tilde{P}_k$	$\tilde{e}(P_k)$	SDJACKP_k	SDRGP$_k$
1	−1.35	−2.936	.0200	.0202	.0186	.288	.00135	.00136	.00125
2	−1.66	−3.024	.0211	.0169	.0149	.282	.00142	.00114	.00100
3	−1.97	−3.093	.0166	.0163	.0152	.276	.00111	.00109	.00102
4	−2.28	−3.179	.0186	.0184	.0150	.272	.00123	.00122	.00099
5	−2.60	−3.174	.0231	.0235	.0194	.272	.00153	.00156	.00129
6	−2.60	−3.185	.0193	.0200	.0166	.271	.00128	.00133	.00111
7	−2.92	−3.300	.0143	.0097	.0090	.264	.00094	.00064	.00059
8	−3.26	−3.315	.0145	.0131	.0113	.263	.00095	.00086	.00074
9	−3.60	−3.396	.0168	.0199	.0164	.258	.00110	.00130	.00107
10	−3.60	−3.345	.0140	.0145	.0127	.261	.00092	.00095	.00083
11	−3.95	−3.446	.0151	.0159	.0129	.254	.00098	.00103	.00084
12	−3.95	−3.412	.0156	.0143	.0123	.256	.00101	.00093	.00080
13	−3.95	−3.428	.0119	.0100	.0102	.255	.00078	.00065	.00067
14	−3.95	−3.429	.0124	.0097	.0083	.255	.00081	.00063	.00054
15	−3.95	−3.433	.0143	.0119	.0106	.255	.00093	.00077	.00069
16	−4.32	−3.491	.0141	.0166	.0139	.251	.00091	.00107	.00089
17	−4.70	−3.564	.0152	.0160	.0142	.247	.00098	.00103	.00091
18	−5.10	−3.639	.0145	.0136	.0136	.242	.00092	.00087	.00087

TABLE 2
Monte Carlo estimators for batting averages.

($a = -3.317$, $b = 0.514$, $M = 500$, $N = 200$, $g = 40$)

k	x_k	$\tilde{\theta}_k$	$\tilde{e}_k$	SDJACK$_k$	SDRG$_k$	$\tilde{P}_k$	$\tilde{e}(P_k)$	SDJACKP$_k$	SDRGP$_k$	$\hat{\hat{P}}_k$ (Efron and Morris)
1	−1.35	−2.908	.0009	.0009	.0010	.290	.00006	.00006	.00007	.290
2	−1.66	−2.969	.0015	.0015	.0015	.286	.00010	.00010	.00010	.286
3	−1.97	−3.037	.0022	.0022	.0022	.281	.00015	.00015	.00015	.281
4	−2.28	−3.104	.0020	.0019	.0020	.277	.00013	.00013	.00013	.277
5	−2.60	−3.169	.0022	.0025	.0026	.273	.00015	.00017	.00017	.273
6	−2.60	−3.166	.0024	.0022	.0023	.273	.00016	.00014	.00015	.273
7	−2.92	−3.235	.0024	.0025	.0026	.268	.00016	.00017	.00017	.268
8	−3.26	−3.305	.0019	.0019	.0019	.264	.00012	.00012	.00013	.264
9	−3.60	−3.378	.0020	.0020	.0019	.259	.00013	.00013	.00013	.259
10	−3.60	−3.380	.0023	.0024	.0024	.259	.00015	.00016	.00016	.259
11	−3.95	−3.448	.0021	.0021	.0022	.254	.00014	.00014	.00014	.254
12	−3.95	−3.488	.0030	.0030	.0030	.254	.00019	.00019	.00019	.254
13	−3.95	−3.449	.0020	.0017	.0017	.254	.00013	.00011	.00011	.254
14	−3.95	−3.447	.0026	.0027	.0028	.254	.00017	.00017	.00018	.254
15	−3.95	−3.454	.0025	.0024	.0024	.254	.00016	.00016	.00015	.254
16	−4.32	−3.527	.0006	.0004	.0005	.249	.00004	.00002	.00002	.249
17	−4.70	−3.607	.0010	.0010	.0010	.244	.00006	.00006	.00006	.244
18	−5.10	−3.686	.0017	.0016	.0017	.239	.00011	.00010	.00011	.239

is computed by deleting a group of 5 iterations each time and SDRG is computed by dividing the sample into 40 groups of 5 iterations each. The Monte Carlo estimators agree with the Efron and Morris estimators.

It is revealed from simulations that the number of iterations N must decrease as M increases to achieve the same accuracy. This is expected. As M increases, it can be observed from Antoniak (1974) or Kuo (1980) that, there is less "doubling up" of the θ's and less variation in the partition π described in § 3. Therefore, the variance given by Theorem 1 decreases. This explains why fewer iterations are needed.

The second example uses the Portsmouth Naval Shipyard data of Martz and Lian (1974) (see also Berry and Christensen (1979)). The Portsmouth Naval Shipyard routinely must assess the quality of submitted lots of vendor-produced material. The number of defects is observed in a sample of size 5 from each of the past lots of welding material; they are denoted by x_k for the kth lot, where k goes from 1 to $n-1$, the number of past lots. In the current lot, the number of defects of a sample of size 5 is also observed, denoted by x_n. Assume x_k, $k=1, \cdots, n$ are binomially distributed with parameters θ_k (sample size is 5), where θ_k denotes the true fraction of the defective material in the kth lot. The objective is to estimate θ_n, the lot fraction defective in the current lot.

We can estimate θ_n as well as all other θ's by (7) with $f_i(x|\theta)=\binom{5}{x}\theta^x(1-\theta)^{5-x}$ for all $i=1, \cdots, n$ and $MF_0(\theta)$ chosen by the statistician.

Based on 5, 1, 0, 0, 0, 0, for the observed values of x_k, $k=1, \cdots, 6$, and the beta distribution $\beta e(a, b)$ chosen to be $F_0(\theta)$, we have computed the Monte Carlo estimator $\tilde{\theta}_k$ from (10). Its posterior standard error has been computed by (12), the jackknife, and the random group methods for two sets of iterations. It takes more iterations for the random group method to obtain stabilized variances.

These estimates are listed in Table 3 and Table 4 for M, a, b, chosen to be 1, 1, 1 and 1, 0.5, 0.5 respectively. We have guessed that the prior is more concentrated at

TABLE 3
Monte Carlo estimators for naval shipyard data.

($M=1$, $a=1$, $b=1$)

(I) $N=2000$, $g=40$					
k	x_k	$\tilde{\theta}_k$	$\tilde{e}_k$	SDJACK$_k$	SDRG$_k$
1	5	0.851	0.0008	0.0009	0.0013
2	1	0.138	0.0064	0.0073	0.0064
3	0	0.079	0.0015	0.0016	0.0017
4	0	0.079	0.0015	0.0021	0.0026
5	0	0.079	0.0015	0.0020	0.0018
6	0	0.081	0.0017	0.0018	0.0020
(II) $N=500$, $g=25$					
k	x_k	$\tilde{\theta}_k$	$\tilde{e}_k$	SDJACK$_k$	SDRG$_k$
1	5	0.853	0.0012	0.0012	0.0135
2	1	0.154	0.0134	0.0152	0.0163
3	0	0.077	0.0032	0.0037	0.0067
4	0	0.074	0.0029	0.0037	0.0055
5	0	0.075	0.0031	0.0039	0.0046
6	0	0.078	0.0035	0.0026	0.0036

TABLE 4
Monte Carlo estimators for naval shipyard data.

($M=1$, $a=0.5$, $b=0.5$)

(I) $N=2000$, $g=40$

k	x_k	$\tilde{\theta}_k$	$\tilde{e}_k$	SDJACK$_k$	SDRG$_k$
1	5	0.912	0.0006	0.0007	0.0009
2	1	0.126	0.0060	0.0070	0.0059
3	0	0.056	0.0016	0.0017	0.0016
4	0	0.056	0.0016	0.0021	0.0022
5	0	0.056	0.0016	0.0020	0.0017
6	0	0.058	0.0017	0.0019	0.0019

(II) $N=500$, $g=25$

k	x_k	$\tilde{\theta}_k$	$\tilde{e}_k$	SDJACK$_k$	SDRG$_k$
1	5	0.914	0.0009	0.0010	0.0147
2	1	0.140	0.0120	0.0137	0.0182
3	0	0.053	0.0034	0.0040	0.0051
4	0	0.050	0.0030	0.0037	0.0042
5	0	0.052	0.0034	0.0041	0.0037
6	0	0.054	0.0035	0.0026	0.0029

the end points of the interval [0, 1] in the latter case. Therefore, we expect that the Bayes estimates are spread out more toward 0 or 1 in that case. This is confirmed by the Monte Carlo estimates given in Tables 3 and 4.

4. Conclusion. Although only empirical Bayes problems are illustrated here, the proposed Monte Carlo method could be applied to compute any estimators based on a mixture of Dirichlet processes, such as densities, response curves in bio-assay, the biometric functions in life tables, or regression functions which are treated in Lo (1978). This method has the advantages that it can achieve satisfactory accuracy with a moderate number of iterations and it takes only a few programming instructions to implement. The number of iterations required to achieve a certain accuracy can be approximated by the prior error bound. The accuracy can also be improved later by increasing the number of iterations, using the posterior error bound.

Appendix I. In this appendix, the proof of Theorem 1 is given.

Proof of Theorem 1. We have

$$\text{Variance } \phi(\zeta_1, \cdots, \zeta_n) = \sum_{\delta \in Z^*} p_{1\delta_1} \cdots p_{n\delta_n} \phi(\delta_1, \cdots, \delta_n)^2 - \left(\sum_{\delta \in Z^*} p_{1\delta_1} \cdots p_{n\delta_n} \phi(\delta_1, \cdots, \delta_n) \right)^2, \tag{A1}$$

which we may write, for brevity as

$$V_n = \Sigma w_n \phi^2 - (\Sigma w_n \phi)^2. \tag{A2}$$

Fix $\delta_1, \cdots, \delta_{n-1}$. Let us define $b_j(\delta_1, \cdots, \delta_{n-1})$, for $j=1, \cdots, n$, by

$$b_j(\delta_1, \cdots, \delta_{n-1}) = \phi(\delta_1, \cdots, \delta_{n-1}, j) \tag{A3}$$

and define ψ to be the $(n-1)$-variable function

$$\psi(\delta_1, \cdots, \delta_{n-1}) = \sum_{j=1}^{n} p_{nj} b_j(\delta_1, \cdots, \delta_{n-1}). \tag{A4}$$

Let us write $\psi = \sum_{j=1}^{n} p_{nj} b_j$ for brevity.

It will be proved in Lemma 1 and 2 that

$$\sum_{j=1}^{n} p_{nj} b_j^2 - \psi^2 \leqq \tfrac{1}{4}\Delta_n.$$

Using this result, we have from (A1) and (A2) that

$$\begin{aligned} V_n &= \Sigma' w' \left(\sum_{j=1}^{n} p_{nj} b_j^2 \right) - \left(\Sigma' w' \left(\sum_{j=1}^{n} p_{nj} b_j \right) \right)^2 \\ &= \Sigma' w' \left(\sum_{j=1}^{n} p_{nj} b_j^2 - \psi^2 + \psi^2 \right) - (\Sigma' w' \psi)^2 \\ &\leqq \Sigma' w' \cdot \tfrac{1}{4}\Delta_n + \Sigma' w' \psi^2 - (\Sigma' w' \psi)^2 \\ &= \tfrac{1}{4}\Delta_n + \Sigma' w' \psi^2 - (\Sigma' w' \psi)^2 \end{aligned}$$

where Σ' denotes the summation over all $\delta_1, \cdots, \delta_{n-1}$, and $w' = p_{1\delta_1} \cdots p_{(n-1)\delta_{n-1}}$.

Let $V_{n-1} = \Sigma' w' \psi^2 - (\Sigma' w' \psi)^2$. Then,

$$V_n \leqq V_{n-1} + \tfrac{1}{4}\Delta_n. \tag{A5}$$

Moreover, for $k_1, k_2 \in Z_{n-1}$, we have:

$$\begin{aligned} &[\psi(\delta_1, \cdots, \delta_{n-2}, k_1) - \psi(\delta_1, \cdots, \delta_{n-2}, k_2)]^2 \\ &\quad = \left(\sum_{j=1}^{n} p_{nj} \phi(\delta_1, \cdots, \delta_{n-2}, k_1, j) - \sum_{j=1}^{n} p_{nj} \phi(\delta_1, \cdots, \delta_{n-2}, k_2, j) \right)^2 \\ &\quad \leqq \sum_{j=1}^{n} p_{nj} [\phi(\delta_1, \cdots, \delta_{n-2}, k_1, j) - \phi(\delta_1, \cdots, \delta_{n-2}, k_2, j)]^2 \end{aligned}$$

by the Cauchy-Schwarz inequality

$$\begin{aligned} &\leqq \sum_{j=1}^{n} p_{nj} \Delta_{n-1} \quad \text{by (5)} \\ &= \Delta_{n-1}. \end{aligned}$$

Therefore, we have inductively:

$$\begin{aligned} V_{n-1} &\leqq V_{n-2} + \tfrac{1}{4}\Delta_{n-1}, \\ &\vdots \\ V_2 &\leqq V_1 + \tfrac{1}{4}\Delta_2, \\ V_1 &= 0. \end{aligned}$$

Therefore, together with (A5), we have:

$$V_n \leqq \tfrac{1}{4} \sum_{i=2}^{n} \Delta_i.$$

LEMMA 1. *If X is a random variable with values in* $[0, 1]$, *then* $\operatorname{Var} X \leqq \frac{1}{4}$.
Proof.

$$\operatorname{Var} X = EX^2 - (EX)^2 = EX^2 - EX + EX - (EX)^2$$
$$= -EX(1-X) + EX(1-EX) \leqq EX(1-EX) \leqq \tfrac{1}{4}.$$

LEMMA 2. *For any fixed* $\delta_1, \cdots, \delta_{n-1}$, *let* b_j, $j = 1, \cdots, n$, *and* ψ *be defined as in* (A3), (A4). *Then* $\sum_{j=1}^{n} p_{nj} b_j^2 - \psi^2 \leqq \frac{1}{4}\Delta_n$.

Proof. Let Y be a discrete random variable which takes on the values $b_1, \cdots, b_n$ with probabilities $p_{n1}, \cdots, p_{nn}$ and X be the random variable such that:

$$X = \frac{Y - \min_{1 \leqq j \leqq n} b_j}{\max_{1 \leqq j \leqq n} b_j - \min_{1 \leqq j \leqq n} b_j}.$$

Then X is a random variable with values in $[0, 1]$. Therefore, $\sum_{j=1}^{n} p_{nj} b_j^2 - \psi^2 = \operatorname{Var} Y \leqq \frac{1}{4}\Delta_n$ by straightforward computation and Lemma 1.

Appendix II. Jackknife and random group methods to estimate variances are discussed in this appendix.

Let the sample $(U_1, V_1), \cdots, (U_N, V_N)$ be independent identically distributed two-dimensional vectors from a distribution F_θ. Estimator $\hat{\theta} = \bar{U}/\bar{V}$ is computed from the full sample. We partition the complete sample into g groups of m observations each ($N = mg$). Let $\hat{\theta}_{(j)}$ be the estimator of the same functional form as $\hat{\theta}$ but computed from the sample after omitting the jth group. Define "pseudo values"

$$\theta_j^* = g\hat{\theta} - (g-1)\hat{\theta}_{(j)},$$

and

$$\theta_Q^* = \sum_{j=1}^{g} \theta_j^*/g.$$

Then the jackknife estimator of variance is

$$V_{\text{JACK}} = \frac{1}{g(g-1)} \sum_{j=1}^{g} (\theta_j^* - \theta_Q^*)^2.$$

Let θ_j^{**} be the estimator of the same functional form as $\hat{\theta}$ but computed from the subsample from the jth group. Let

$$\bar{\theta} = \sum_{j=1}^{g} \theta_j^{**}/g.$$

Then the random group estimator of $\operatorname{Var} \hat{\theta}$ is

$$V_{RG} = \frac{1}{g(g-1)} \sum_{j=1}^{g} (\theta_j^{**} - \bar{\theta})^2.$$

Acknowledgments. This article is revised from Chapter 1 of the author's dissertation written at UCLA under the direction of Professor Thomas Ferguson. I am extremely grateful to him for his initial suggestion of the topic and patient guidance throughout the project. I would also like to thank the referee and Michael P. Cohen for many constructive comments which helped to improve the presentation.

REFERENCES

C. E. ANTONIAK (1974), *Mixtures of Dirichlet processes with applications to Bayesian nonparametric problems*, Ann. Statist., 2, pp. 1152–1174.

D. BERRY AND R. CHRISTENSEN (1979), *Empirical Bayes estimation of a binomial parameter via mixtures of Dirichlet processes*, Ann. Statist., 7, pp. 558–568.

B. EFRON AND C. MORRIS (1975), *Data analysis using Stein's estimator and its generalizations*, J. Amer. Statist. Assoc., 70, pp. 311–319.

T. S. FERGUSON (1973), *A Bayesian analysis of some nonparametric problems*, Ann. Statist., 1, pp. 209–230.

J. M. HAMMERSLEY AND D. C. HANDSCOMB (1964), *Monte Carlo Methods*, John Wiley, New York.

J. M. HAMMERSLEY (1960), *Monte Carlo methods for solving multivariable problems*, Ann. New York Acad. Sci. 86, Art. 3, pp. 844–874.

L. KUO (1980). *Computations of mixtures of Dirichlet processes*, Technical Report No. 96, Dept Statistics, Univ. Michigan, or Dissertation, Mathematics Dept, Univ. California, Los Angeles.

A. Y. LO (1978), *Bayesian nonparametric density methods*, Technical Report, Univ. California, Berkeley.

——— (1984), *On a class of Bayesian nonparametric estimates*: 1. *Density estimates*, Ann. Statist., 12, pp. 351–357.

H. F. MARTZ AND M. G. LIAN (1974), *Empirical Bayes estimation of the binomial parameter*, Biometrika, 61, pp. 517–523.

R. G. MILLER (1974), *The jackknife—a review*, Biometrika, 61, pp. 1–15.

B. L. S. PRAKASA RAO (1983), *Nonparametric Functional Estimation*, Academic Press, Orlando, FL.

Des RAJ (1968), *Sampling Theory*, McGraw-Hill, New York.

C. R. RAO (1973), *Linear Statistical Inference and Its Applications*, second ed., John Wiley, New York.

R. A. TAPIA AND J. R. THOMPSON (1978), *Nonparametric Probability Density Estimation*, Johns Hopkins Univ. Press, Baltimore.

SIAM J. SCI. STAT. COMPUT.
Vol. 7, No. 1,January 1986

005

FINITE ALGORITHMS FOR HUBER'S *M*-ESTIMATOR*

D. I. CLARK† AND M. R. OSBORNE‡

Abstract. Two finite algorithms for Huber's *M*-estimator are presented. Both proceed in a constructive manner by moving from one partition to an adjacent one. One of the algorithms, which uses the tuning constant as a continuation parameter, also has the facility to simultaneously estimate the tuning constant and scaling factor. Stable and efficient implementation of the algorithms is presented, together with numerical results.

Key words. Huber's *M*-estimator, algorithms, finiteness

1. Introduction. Along with the rapid development of and great interest in the theory of robust estimators over the last decade, there has naturally arisen a corresponding interest in algorithms to calculate their estimates. In particular, Huber's *M*-estimator [14] has been the subject of a variety of approaches, some of which have been extensively tested, particularly in the location (single dimension) case. Of these, approaches which attempt to find a specific number of outliers by considering in some manner all possible subsets [1], [10], have not proved successful, both because of the amount of work required and because of possible ambiguities in the result.

The most popular approaches, which are summarized below, are based on iterative schemes. In some of these, a scaling factor is estimated by the algorithm at each iteration [15], [17], whilst in others it is estimated once, before computation begins [3], [13].

We are concerned with the problem

$$\min f(\mathbf{x}) = \sum_{i=1}^{n} \rho\left(\frac{r_i}{\tau}\right), \tag{1.1}$$

where ρ is the function

$$\rho(t) = \begin{cases} \frac{1}{2}t^2 & \text{for } |t| \leqq \gamma, \\ \gamma|t| - \frac{1}{2}\gamma^2 & \text{for } |t| > \gamma, \end{cases} \tag{1.2}$$

r_i is the ith residual or error,

$$\mathbf{r} = A\mathbf{x} - \mathbf{b}, \tag{1.3}$$

τ is a scaling parameter and γ a tuning constant. The vectors $\mathbf{r}$, $\mathbf{b}$ and $\mathbf{x}$ are of dimension n, n and p respectively, whilst A is an $n \times p$ matrix ($n \geqq p$), which we assume is full rank.

Letting $\psi = \rho'$, (1.1) may be expressed as solving

$$\sum_{i=1}^{n} \psi(r_i) = 0. \tag{1.4}$$

Three iterative schemes have been suggested for solving (1.4),

$$\mathbf{x}^{i+1} = \mathbf{x}^i + [A^T\langle\psi'(\mathbf{r}^i)\rangle A]^{-1}A^T\psi(\mathbf{r}^i), \tag{1.5}$$

$$\mathbf{x}^{i+1} = \mathbf{x}^i + [A^TA]^{-1}A^T\psi(\mathbf{r}^i), \tag{1.6}$$

$$\mathbf{x}^{i+1} = \mathbf{x}^i + [A^T\langle w(\mathbf{r}^i)\rangle A]^{-1}A^T\langle w(\mathbf{r}^i)\rangle\mathbf{r}^i, \tag{1.7}$$

* Received by the editors June 28, 1983, and in final revised form October 25, 1984.

† School of Information Sciences, Canberra College of Advanced Education, A.C.T., Australia 2616.

‡ Department of Statistics, Institute of Advanced Studies, Australian National University, A.C.T., Australia 2600.

where $\langle a \rangle$ denotes the diagonal matrix with $\langle\ \rangle_{ii} = a_i$, and w is a weight function such as $w(t) = \psi(t)/t$.

Equation (1.5) is Newton's method and has been applied by Huber and Dutter [17]. According to Holland and Welsch [13], it is the fastest of the three methods, but requiring ψ makes it difficult to implement and also $A^T\langle\psi'\rangle A$ need not be positive definite. Equation (1.6) is Huber's method [15]. As opposed to the other two methods where the updating matrix must be inverted at each iteration, it has the advantage that the generalized inverse need only be calculated once, but Holland and Welsch report it as being the slowest method of the three. Equation (1.7) is the iteratively reweighted least squares attributed to Beaton and Tukey [3], and widely discussed [4], [5], [6], [13].

The above methods are iterative and the stopping criteria most favoured appear to be to take five iterations (in the location case) [4], [13], and to specify a tolerance parameter and compare it to the relative change in the estimator at successive iterations [13], [17]. Huber [15] has also noted that if the correct final partitioning into $\{i \,|\, |r_i| \leqq \gamma\}$, $\{i \,|\, r_i > \gamma\}$, $\{i \,|\, r_i < -\gamma\}$ were known, the M-estimate could be calculated in a single step, and he and Dutter refine their algorithm by keeping track of partition changes between iterations.

The first algorithm presented below is based on the result that $\mathbf{z} = \mathbf{z}(\gamma)$, the solution of (1.1), is a continuous piecewise linear of γ. When γ is arbitrarily large, the M-estimate is simply the least squares (LS) estimate, and the algorithm proceeds from this point by reducing the value of γ in ranges until either a desired value of γ is reached or a specified number of outliers is identified. (An alternative approach is to start with the l_1 estimate, $\gamma = 0$, and increase γ).

A second algorithm, which was developed to solve a problem akin to degeneracy in linear programming in the first algorithm is also presented. This algorithm finds the correct partitioning of the residuals in a constructive manner. Both algorithms have been tested, and results are given.

This paper relies heavily on results from a previous paper [8], which investigates the mathematical structure of Huber's M-estimator. Results from it will be quoted as appropriate.

2. The continuation algorithm.

2.1. General description. As mentioned above, the first algorithm depends on the behaviour of the M-estimate as γ is varied. We show that it is a continuous piecewise linear function of γ, and hence so is each residual. As for large enough γ, the M-estimator is the LS estimator; this is taken as the starting point and the value of γ is reduced until the size of some residual is equal to γ. At this point the partitioning of residuals changes, and the rate of change of the M-estimate and hence each residual with respect to γ has to be recalculated. Piecewise linearity now gives the estimate on the new partition until a new tie occurs.

This process is repeated until either a predetermined number of outliers is identified, or a desired value of γ is reached. The algorithm can thus be thought of as a continuation algorithm with parameter γ. The choice of the "stopping" value of γ is discussed in § 2.5.

As γ is reduced, the normal pattern is that a residual changes status from being $\leqq \gamma$ to becoming $> \gamma$ in size. Occasionally, however, the opposite may occur, so that a residual which was an outlier at one range of values of γ is not one for smaller γ. Further, it is theoretically possible that more than one residual could be involved in changing status at the same value of γ. The resolution of this problem, analogous to degeneracy in linear programming, has led to the algorithm described in § 3.

2.2. Piecewise linearity. We will consider the class of procedures for estimating $\mathbf{x}$ defined by

$$\min F(\mathbf{x}, y) = \tfrac{1}{2} \sum_{i \in \sigma} r_i^2 + \sum_{i \in \bar{\sigma}} (\gamma |r_i| - \tfrac{1}{2}\gamma^2) \tag{2.1}$$

where σ and $\bar{\sigma}$ are disjoint complementary subsets of $N = \{i | i = 1, \cdots, n\}$. The subsets $\sigma, \bar{\sigma}$ define a partition P.

When we wish to refer to the optimal value of (2.1) for a specific partition, say P_a, and tuning constant γ, we will use the notation $\mathbf{z}(\gamma, P_a)$ or, more simply, $\mathbf{z}_a$ with corresponding $r_i(\gamma, P_a)$ or $r_i(\mathbf{z}_a)$.

On occasion it may be necessary to distinguish between $\mathbf{z}(\gamma, P)$ and the solution to (1.1) which we shall denote by $\mathbf{z}(\gamma)$ or, where there is no ambiguity, $\mathbf{z}$, with corresponding r_i.

A partition P will be called feasible if $i \in \sigma \Leftrightarrow |r_i(\gamma, P)| \leqq \gamma$ and the $r_i(\gamma, P)$, $i \in \bar{\sigma}$ are of the assumed sign.

In Clark [8, Thm. 2, Cor.] it was shown that function values at the minima of all feasible partitions at the current value of γ are equal, so that if P is feasible (2.1) is equivalent to (1.1) and $\mathbf{z}(\gamma, P)$ is an M-estimate. (We will assume for the time being that τ, the scaling factor of (1.1) is unity).

In showing that $\mathbf{z}$ is a piecewise linear function of γ we will make two assumptions: uniqueness of the M-estimate, and nondegeneracy (i.e. the set $\{k | |r_k| = \gamma\}$ has at most one element $\forall \gamma$). Neither assumption is necessary, but the alternatives require careful analysis and the added complexity would detract from the simplicity of the basic procedure. The uniqueness assumption is commented upon below, and the question of degeneracy is addressed in § 3.4.

The proof of the piecewise linearity of $\mathbf{z}$ is a constructive one. We first show that within a feasible partition $\mathbf{z}$ is linear in γ, is feasible for a range of values of γ, and at the end of this range also minimizes a new partition which in turn is feasible for a range of values of γ, $\mathbf{z}$ again being linear in γ.

We will need the following results from Clark [8] where they occur as Lemma 3, Lemma 6 and Theorem 4.

LEMMA 2.1. *If P_a and P_b are adjacent partitions with $\sigma_a = \sigma_b \cup \{k\}$, then $|r_k(\mathbf{z}_a)| = \gamma \Rightarrow \mathbf{z}_a$ minimizes P_b, and $|r_k(\mathbf{z}_b)| = \gamma \Rightarrow \mathbf{z}_b$ minimizes P_a.*

LEMMA 2.2. *A partition has a unique minimizer iff the vectors $\mathbf{a}_i$, $i \in \sigma$, span R^p.*

LEMMA 2.3. *Let P_a and P_b be adjacent partitions with unique minimizers $\mathbf{z}_a$ and $\mathbf{z}_b$ respectively, and $\sigma_a = \sigma_b \cup \{k\}$. Then*

$$\text{(i)} \qquad |r_k(\mathbf{z}_a)| > \gamma \Leftrightarrow |r_k(\mathbf{z}_b)| > \gamma,$$

$$\text{(ii)} \qquad |r_k(\mathbf{z}_a)| \leqq \gamma \Leftrightarrow |r_k(\mathbf{z}_b)| \leqq \gamma.$$

THEOREM 2.4. *If a partition has a unique minimizer, then the minimizer is a linear function of γ.*

Proof. Differentiating (2.1) with respect to $\mathbf{x}$ gives

$$\begin{aligned} \mathbf{0} &= \sum_{\sigma} \mathbf{a}_i r_i + \gamma \sum_{\bar{\sigma}} \theta_i \mathbf{a}_i \\ &= \sum_{\sigma} \mathbf{a}_i (\mathbf{a}_i^T \mathbf{z} - b_i) + \gamma \sum_{\bar{\sigma}} \theta_i \mathbf{a}_i \\ &= A_\sigma^T A_\sigma \mathbf{z} - \sum_{\sigma} \mathbf{a}_i b_i + \gamma \sum_{\bar{\sigma}} \theta_i \mathbf{a}_i, \end{aligned} \tag{2.2}$$

where A_σ is the submatrix of A pointed to by the members of σ, $\mathbf{a}_i^T$ is the ith row of A and $\theta_i = \operatorname{sgn}(r_i)$.

The result now follows from the nonsingularity of $A_\sigma^T A_\sigma$, a consequence of Lemma 2.2. □

THEOREM 2.5. *Let P_a be a feasible partition at $\gamma > \bar{\gamma}$, but infeasible at $\gamma < \bar{\gamma}$, the infeasibility being caused by the size of a single residual $r_k(\mathbf{z}_a)$. Then the partition $\sigma_b = \sigma_a \backslash \{k\} (k \in \sigma_a)$ or $\sigma_b = \sigma_a \cup \{k\}$ $(k \in \bar{\sigma}_a)$ is feasible for some $\gamma < \bar{\gamma}$.*

Proof. For $\gamma \leqq \bar{\gamma}$,

$$|r_k(\mathbf{z}_a)| > \gamma (k \in \sigma_a) \quad \text{or} \quad |r_k(\mathbf{z}_a)| \leqq \gamma (k \in \bar{\sigma}_a),$$

so by Lemma 2.3 and observing that $k \in \sigma_a$ $(\bar{\sigma}_a) \Rightarrow k \in \bar{\sigma}_n$ (σ_b),

$$|r_k(\mathbf{z}_b)| > \gamma \ (k \in \bar{\sigma}_b) \quad \text{or} \quad |r_k(\mathbf{z}_b)| \leqq \gamma \ (k \in \sigma_b).$$

Thus provided the reduction in γ is slight enough so that $|r_i(\mathbf{z}_b)| \neq \gamma$, $i \neq k$, P_b is a feasible partition. □

Remark 2.1. The point $\gamma = \bar{\gamma}$, where $|r_k| = \gamma$, will belong to the lower limit of the upper range if $k \in \sigma$, and the upper limit of the lower range if $k \in \bar{\sigma}$.

THEOREM 2.6. *In the absence of degeneracy, if the M-estimate is unique, then it is a continuous piecewise linear function of γ.*

Proof. For any feasible partition, as $\mathbf{z}$ is a linear function of γ, $r_i(\mathbf{z}) = \mathbf{a}_i^T \mathbf{z} - b_i$ is also a linear function of γ for all i, so that P will remain feasible for a range of values of γ. Moreover, at the end of the feasible range of γ, $|r_k| = \gamma$. The result now follows from Lemma 2.1 and Theorems 2.4 and 2.5. □

Remark 2.2. In the presence of degeneracy, it is possible for the feasible range of γ for a partition to be a single point. An example of this is given in [8, Remark 5].

Remark 2.3. If the M-estimate is nonunique, there is still an M-estimate which is a continuous piecewise linear function of γ with rank $A_\sigma = p$. The proof depends on the set of M-estimates being bounded and convex with rank $A = p$ on the boundaries. It is rather intricate to prove in the presence of degeneracy, and in the light of the comments below on nonuniqueness it has not been included in this paper.

Remark 2.4. We have assumed that the M-estimate is unique. In Clark [8] it is shown that if P_a and P_b are distinct feasible partitions then $|r_i(\mathbf{z}_a)| < \gamma$ or $|r_i(\mathbf{z}_b)| < \gamma \Rightarrow$ $i \in \sigma_a \cap \sigma_b$ (Theorem 1b), and that the $\mathbf{a}_i$, $i \in \sigma_a \cap \sigma_b$ do not span R^p. It is also shown that if a partition has a nonunique minimizer, the $\mathbf{a}_i$, $i \in \sigma$, do not span R^p. Consequently if, as is normal, $n \gg p$ and $|\sigma| \approx \text{rank } A_\sigma$ (or $\sigma_a \cap \sigma_b$ if there are two partitions), then nearly all the residuals are outliers $(> \gamma)$ or near-outliers $(= \gamma)$. Recalling that the terminal value of γ is only an estimate, all this would seriously question the validity of the model in the presence of nonuniqueness. (Recall also that by the time γ has been reduced sufficiently for this to happen, $\mathbf{z}(\gamma)$ and $r_i(\gamma)$ are known for all greater values of γ.)

2.3. Updating at change of partition. We have shown that $\mathbf{z}$ is piecewise linear in γ, and that when a new range of γ is entered there will be a change of status of one or more residuals. Here we show how to update the $dr_i/d\gamma$ and $d\mathbf{z}/d\gamma$ in the usual case where only one residual, say r_k, is involved in changing status.

The method has several features of interest. It can make use of the work done in obtaining the LS estimate making it unnecessary to restore the data matrix A (the appropriate method for obtaining the LS estimate is the modified Gram-Schmidt algorithm which gives directly the initial $\mathbf{w}_k$ used in the discussion below). It uses orthogonal transformations, well-known for their numerically stable properties. The updating is extremely efficient, for after an operation requiring the $2(n+p)p$ multiplications and additions characteristic of a rank one modification scheme using orthogonal

transformations, each $dr_i/d\gamma$ requires only a single multiplication and addition to update. (It is for this reason that we have preferred our approach to the more standard methods of updating triangular factors after rank one updating of a positive definite matrix—see, for example, [11].) And finally, a slight modification allows the l_1 estimate to be used as the starting point with γ increasing, rather than starting from the LS estimate and reducing γ.

Differentiating (2.2) with respect to γ gives

$$A_\sigma^T A_\sigma \frac{d\mathbf{z}}{d\gamma} = -\sum_{\bar{\sigma}} \theta_i \mathbf{a}_i. \tag{2.3}$$

If we can find a nonsingular matrix M such that $A_\sigma^T A_\sigma = MM^T$, then (2.3) becomes

$$M^T \frac{d\mathbf{z}}{d\gamma} = -\sum_{i\in\bar{\sigma}} \theta_i M^{-1}\mathbf{a}_i = -\sum_{i\in\bar{\sigma}} \theta_i \mathbf{w}_i, \tag{2.4}$$

where $\mathbf{w}_i = M^{-1}\mathbf{a}_i$. Then

$$\frac{dr_j}{d\gamma} = \mathbf{a}_j^T \frac{d\mathbf{z}}{d\gamma} = -\mathbf{w}_j^T \sum_{i\in\bar{\sigma}} \theta_i \mathbf{w}_i. \tag{2.5}$$

At a change of partition, $\sigma \to \sigma \pm \{k\}$, so that

$$MM^T \to MM^T \pm \mathbf{a}_k\mathbf{a}_k^T = M\{I \pm \mathbf{w}_k\mathbf{w}_k^T\}M^T = M'M'^T. \tag{2.6}$$

Now for any orthogonal transformation $QQ^T = I$,

$$I \pm \mathbf{w}_k\mathbf{w}_k^T = QQ^T \pm QQ^T\mathbf{w}_k\mathbf{w}_k^T QQ^T = Q\{I \pm Q^T\mathbf{w}_k\mathbf{w}_k^T Q\}Q^T. \tag{2.7}$$

If we select Q such that

$$Q^T\mathbf{w}_k = \|\mathbf{w}_k\|\mathbf{e}_\alpha, \qquad 1 \leqq \alpha \leqq p, \tag{2.8}$$

(2.7) becomes

$$I \pm \mathbf{w}_k\mathbf{w}_k^T = Q\{I \pm \|\mathbf{w}_k\|^2\mathbf{e}_\alpha\mathbf{e}_\alpha^T\}Q^T = QD^{1/2}(D^{1/2})^T Q^T, \tag{2.9}$$

as $M'M'^T$ is positive definite and so $1 \pm \mathbf{w}_k^T\mathbf{w}_k > 0$. Then from (2.6)

$$\mathbf{w}_j' = (M')^{-1}\mathbf{a}_j = D^{-1/2}Q^T\mathbf{w}_j. \tag{2.10}$$

The matrix Q can be calculated from the Householder transformation [17]

$$\{I - 2\hat{\mathbf{q}}\hat{\mathbf{q}}^T\}\mathbf{w}_k = \theta_k\|\mathbf{w}_k\|\mathbf{e}_\alpha, \tag{2.11}$$

$$(2\hat{\mathbf{q}}^T\mathbf{w}_k)\hat{\mathbf{q}} = \mathbf{w}_k - \theta_k\|\mathbf{w}_k\|\mathbf{e}_\alpha, \tag{2.12}$$

$$2\hat{\mathbf{q}}^T\mathbf{w}_k = \{2(\mathbf{w}_k^T\mathbf{w}_k - \theta_k\|\mathbf{w}_k\|(\mathbf{w}_k)_\alpha)\}^{1/2}. \tag{2.13}$$

After the $\mathbf{w}_j$ have been updated in this manner, the $dr_j/d\gamma$ can be updated rather efficiently in the following way.

From (2.3) and (2.6),

$$M\{I \pm \mathbf{w}_k\mathbf{w}_k^T\}M^T\frac{d\mathbf{z}'}{d\gamma} = -\sum_{i\in\bar{\sigma}'} \theta_i\mathbf{a}_i = -\sum_{i\in\bar{\sigma}} \theta_i\mathbf{a}_i \pm \theta_k\mathbf{a}_k. \tag{2.14}$$

Now multiplying (2.14) by M^{-1} and using $\mathbf{w}_i = M^{-1}\mathbf{a}_i$ and $\mathbf{a}_i^T d\mathbf{z}/d\gamma = dr_i/d\gamma$, we have

$$
\begin{aligned}
M^T\frac{d\mathbf{z}'}{d\gamma} &= M^T\frac{d\mathbf{z}}{d\gamma} \mp \mathbf{w}_k\mathbf{w}_k^T M^T\frac{d\mathbf{z}'}{d\gamma} \pm \theta_k M^{-1}\mathbf{a}_k \\
&= M^T\frac{\mathrm{d}\mathbf{z}}{d\gamma} \mp \mathbf{w}_k\mathbf{a}_k^T\frac{d\mathbf{z}'}{d\gamma} \pm \theta_k\mathbf{w}_k \\
&= M^T\frac{d\mathbf{z}}{d\gamma} \mp \mathbf{w}_k\frac{dr_k'}{d\gamma} \pm \theta_k\mathbf{w}_k.
\end{aligned} \tag{2.15}
$$

Multiplying (2.15) by $\mathbf{w}_j^T$ yields

$$
\frac{dr_j'}{d\gamma} = \frac{dr_j}{d\gamma} \mp \mathbf{w}_j^T\mathbf{w}_k\frac{dr_k'}{d\gamma} \mp \theta_k\mathbf{w}_j^T\mathbf{w}_k, \tag{2.16}
$$

and setting j to k in (2.16) gives

$$
\frac{dr_k'}{d\gamma} = \left(\frac{dr_k}{d\gamma} \mp \theta_k\|\mathbf{w}_k\|^2\right)\Big/(1 \pm \|w_k\|^2). \tag{2.17}
$$

Now from (2.10),

$$
\mathbf{w}_j^T\mathbf{w}_k = \mathbf{w}_j'^T D^{1/2}Q^TQD^{1/2}\mathbf{w}_k' = (D^{1/2}\mathbf{w}_j')^T(D^{1/2}\mathbf{w}_k') = (D^{1/2}\mathbf{w}_j')_\alpha(D^{1/2}\mathbf{w}_k')_\alpha,
$$

as $\mathbf{w}_k'$ is parallel to $\mathbf{e}_\alpha$. Hence from (2.9)

$$
\mathbf{w}_j^T\mathbf{w}_k = (\mathbf{w}_j')_\alpha(\mathbf{w}_k')_\alpha(1 \pm \|\mathbf{w}_k\|^2). \tag{2.18}
$$

Applying (2.17) and (2.18) to (2.16), we have

$$
\begin{aligned}
\frac{dr_j'}{d\gamma} &= \frac{dr_j}{d\gamma} \mp (\mathbf{w}_j')_\alpha(\mathbf{w}_k')_\alpha\left(\frac{dr_k}{d\gamma} \mp \theta_k\|\mathbf{w}_k\|^2\right) \pm \theta_k(\mathbf{w}_j')_\alpha(\mathbf{w}_k')_\alpha(1 \pm \|\mathbf{w}_k\|^2) \\
&= \frac{dr_j}{d\gamma} \mp (\mathbf{w}_j')(\mathbf{w}_k')\theta_k\left(\left|\frac{dr_k}{d\gamma}\right| - 1\right).
\end{aligned} \tag{2.19}
$$

If we append an initial identity matrix onto A as columns $N+j$, $j = 1, \cdots, p$, there is a corresponding formula for $d\mathbf{z}/d\gamma$,

$$
\left(\frac{d\mathbf{z}'}{d\gamma}\right)_j = \left(\frac{d\mathbf{z}}{d\gamma}\right)_j \mp (\mathbf{w}_{n+j}')_\alpha(\mathbf{w}_k')_\alpha\theta_k\left(\left|\frac{dr_k}{d\gamma}\right| - 1\right). \tag{2.20}
$$

Equations (2.10) to (2.13), (2.19) and (2.20) describe an updating formula at change of partition. In practice, the initial step is to find the LS solution ($\sigma = N$). This can be done using a Choleski factorisation $A^TA = LL^T$, or a modified Gram-Schmidt $A = Q_1U$ so that $A^TA = U^TU$ and $\mathbf{w}_i = \boldsymbol{\kappa}_i(Q_1^T)$. Either way the initial M is triangular, but this does not persist beyond the first step.

Examination of the updating equations shows that updating the $\mathbf{w}_j$ requires $2(n+p)p$ multiplications and additions and then updating the $d\mathbf{r}_j/d\gamma$ and $d\mathbf{z}/d\gamma$ requires $n+p$ of each. The operation count per iteration is thus $2np(1+O(P/n))$ for fixed p as $n \to \infty$. Our experience indicates that it is very rare for a residual to change status more than once, so that if there are l outliers there are likely to be l iterations when starting from LS (γ large) and $n-p-l$ when starting from l_1 ($\gamma = 0$). The major storage requirement is the $p(n+p)$ array required to store the $\mathbf{w}_j$ which can over-write the initial pn array A. Single arrays are needed for $\mathbf{r}$, $\mathbf{z}$, $d\mathbf{r}/d\gamma$, $d\mathbf{z}/d\gamma$ and for pointers to the index set σ.

If the algorithm is started with the LS estimate, so that γ is reduced then at almost all changes of partition $\sigma' = \sigma \backslash \{k\}$, corresponding to a small residual being reclassified as large, so that MM^T must be "downdated" (i.e. $M'M'^T = MM^T - \mathbf{a}_k \mathbf{a}_k^T$). This is unlikely to cause numerical difficulties in the usual situation where a relatively small number of residuals are outliers. Indeed, in the numerical experiments reported in § 4, where for testing purposes γ was reduced to zero, instability was never encountered. However, as downdating is potentially a problem numerically, it may be advisable to monitor the determinant of MM^T. If consistent estimates are to be possible, the determinant would be expected to decrease linearly in $|\sigma|$ (for large enough $|\sigma|$), so that monitoring $\det MM^T$ (from the initial decomposition and the formula $\det(M'M'^T) = \det(MM^T)(1 - \|\mathbf{w}_k\|^2)$) and reporting any sudden drop is recommended.

If the l_1 estimate is used as a starting point, the typical iteration corresponds to a large residual being reclassified as small and would involve the more stable updating $M'M'^T = MM^T + \mathbf{a}_k \mathbf{a}_k^T$. However, in the usual situation where $n \gg l+p$, this would require many more iterations than starting from LS.

Remark 2.5. In (2.8), Q was selected so that $Q^T \mathbf{w}_k$ was parallel to $\mathbf{e}_\alpha$, without specifying α. It can be shown [7] that as far as the size of the $\mathbf{w}_j$ is concerned, it does not matter which α is chosen. However, choosing the same α for each iteration could cause a build-up in the αth element of some of the $\mathbf{w}_j$ resulting in an accelerating process of the $\mathbf{w}_j$ becoming parallel to $\mathbf{e}_\alpha$ and hence to each other. It may thus be safer to cycle α. Both ways ($\alpha = 1$ always, and cycling α) were tested in the implementation of the algorithm without any apparent difference being observed.

2.4. Choice of scaling factor and tuning constant. Thus far, we have sidestepped the question of the scaling factor by assuming that τ in (1.1) is unity, but this choice suffers from the disadvantage that it does not give an unbiased estimator. Huber [15] suggests that in the location case, the best estimate of scale is given by

$$\tau = \underset{i}{\text{med}} \{|a_i - \underset{j}{\text{med}} \{a_j\}|\}.$$

However, in the regression case the analogue of the median, the l_1 estimate, requires a calculation of perhaps similar complexity to that of the M-estimate itself. Holland and Welsch [13] suggest

$$\tau = 1.48(\underset{i}{\text{med}} \{|r_i - \underset{j}{\text{med}} \{r_j\}|\}) = 1.48\mu, \tag{2.21}$$

introducing the factor 1.48 to give an approximately unbiased estimate of scale when the error model is Gaussian. This factor is also cited by Birch [5] as being a popular choice.

The choice of γ is taken as 1.345 by Holland and Welsch [13] as giving 95% asymptotic efficiency at the Gaussian distribution.

Most algorithms estimate τ only once, using (2.21) at the starting point which is usually the LS estimator. However, as we shall see, estimating τ iteratively can easily be accommodated by the continuation algorithm with minimal extra effort.

If τ is not assumed to be unity in (1.1), the definition of σ becomes $i \in \sigma \Leftrightarrow |r_i/\tau| \leqq \gamma$, or $i \in \sigma \Leftrightarrow |r_i| \leqq \gamma\tau$. This suggests using $\gamma\tau = \delta$ as our continuation parameter. Moreover, the inclusion of τ does not affect equations (2.2) f.f. and thus $d\mathbf{z}/d\delta = 1/\tau \, d\mathbf{z}/d\gamma$. The rates of change of $\mathbf{z}$ and r_i are thus calculated as before, and then divided by τ.

The stopping rule now becomes: If $\delta' > 1.48 \times 1.345\mu' = 1.99\mu'$ at the beginning of a range and $\delta'' \leqq 1.99\mu''$ at the end of the range, choose as the final value of δ, $\delta' + (\delta' - \delta'')(1.99\mu'' - \delta'')/(\delta' - 1.99\mu')$.

This approach has two important advantages. First, although the scale factor is estimated at each iteration, its use in the algorithm only affects the stopping rule. This means that questions of convergence of τ are sidestepped. (This contrasts with the difficulty of simultaneously estimating $\mathbf{z}$ and τ for fixed γ in the iterative schemes—see for example the method of Huber and Dutter [15] when τ is defined by Huber's Proposal 2 [14]. Indeed, when τ is defined as in (2.21), a scheme such as Huber and Dutter's need not converge).

Second, although we have chosen (2.21) as defining τ, other choices of scaling factor can easily be incorporated. In particular, methods of determining scale by satisfying an auxiliary equation of the form $f(x, \tau, \gamma) = 0$, such as Huber's Proposal 2, may be used.

Remark 2.6. There is an assumption in the above stopping rule that μ varies linearly over a range of δ. However, although the r_i do change linearly, it may be that $\arg\operatorname{med}_i \{r_i\}$ changes, in which case μ would only be piecewise linear. This has not given rise to any difficulties in practice, and it does make available an estimate of scale which uses (2.21) at a point close to the final value of the M-estimator. It seems likely that this would suffice for practical purposes.

2.5. Finiteness. In order to demonstrate that the algorithm is finite, we need to show that

(i) cycling does not occur at change of partition and

(ii) the number of ranges of γ is finite.

That cycling does not occur when there is only one residual involved in changing status is a consequence of Theorem 2.5. When more than one is involved, we use the partitioning algorithm described in § 3. The specific use of the algorithm to avoid cycling is discussed in § 3.3.

We now show that the number of ranges of γ is finite.

THEOREM 2.7. *The number of ranges of γ is finite.*

Proof. For any partition P for which A_σ is of full rank, $\mathbf{z}(\gamma, P)$ and hence the $r_i(\gamma, P)$ are linear functions of γ. Hence, as γ decreases, once a given residual has changed status it cannot return to its original status. The result now follows from the finiteness of the number of partitions. □

2.6. The algorithm summarised.

Step 1. Set $j \leftarrow 1$ (counter).
Find the LS estimate, $\mathbf{z}_1$.
Set $d\mathbf{z}/d\gamma$, $dr_i/d\gamma \leftarrow 0$.
Determine $k = \arg\max_i \{|r_i|\}$.
Let $P = \{N\backslash\{k\}, \{k\}\}$ and $\gamma_1 = |r_k|$.
Calculate $\mu_1 = \operatorname{med}_i \{|r_i - \operatorname{med}_j \{r_j\}|\}$
If $\gamma_1 \leqq 1.99\mu_1$, stop.

Step 2. Update $d\mathbf{z}/d\gamma$, $dr_i/d\gamma$ using equations (2.10) to (2.13), (2.20) and (2.21).
Let

$$\eta_1 = \min\left\{\left.\frac{r_i - \gamma_j}{\frac{dr_i}{d\gamma} - 1}\right| i \in \bar{\sigma},\ r_i > \gamma,\ \frac{dr_i}{d\gamma} > 1\right\},$$

$$\eta_2 = \min\left\{\left.\frac{-r_i - \gamma_j}{\frac{-dr_i}{d\gamma} - 1}\right| i \in \bar{\sigma},\ r_i < -\gamma,\ \frac{dr_i}{d\gamma} < -1\right\},$$

$$\eta_3 = \min\left\{\left.\frac{\gamma_j + r_i}{1 + \frac{dr_i}{d\gamma}}\right| i \in \sigma, \frac{dr_i}{d\gamma} > \frac{r_i}{\gamma}\right\},$$

$$\eta_4 = \min\left\{\left.\frac{\gamma_j - r_i}{1 - \frac{dr_i}{d\gamma}}\right| i \in \sigma, \frac{dr_i}{d\gamma} \leqq \frac{r_i}{\gamma}\right\}.$$

Set $\gamma_{j+1} = \gamma_j - \min\{\eta_1, \eta_2, \eta_3, \eta_4\}$.

Calculate $\mathbf{z}(\gamma_{j+1}) = \mathbf{z}(\gamma_j) - \frac{d\mathbf{z}}{d\gamma}(\gamma_{j+1} - \gamma_j)$

$$r_i(\gamma_{j+1}) = r_i(\gamma_j) - \frac{dr_i}{d\gamma}(\gamma_{j+1} - \gamma_j)$$

$$\mu_{j+1} = \text{med}_i\{|r_i(\gamma_{j+1}) - \text{med}_k\{r_k(\gamma_{j+1})\}|\}.$$

Step 3. If $\gamma_{j+1} \leqq 1.99\mu_{j+1}$, stop, with

$$\mathbf{z} = \alpha\mathbf{z}(\gamma_j) + (1-\alpha)\mathbf{z}(\gamma_{j+1})$$

$$\sigma = 1.48[\alpha\mu_j + (1-\alpha)\mu_{j+1}]$$

$$\gamma = [\alpha\gamma_j + (1-\alpha)\gamma_{j+1}]/\sigma$$

where $\alpha = \dfrac{1.99\mu_{j+1} - \gamma_{j+1}}{\gamma_j - 1.99\mu_j}$.

Step 4. If only r_k is involved in changing status, define the new partition by

$$\sigma_{j+1} = \sigma_j \cup \{k\} (k \in \bar{\sigma})$$

$$\sigma_{j+1} = \sigma_j \backslash \{k\} (k \in \sigma).$$

If more than one residual is involved in changing status, define the new partition as in § 3.3.

$$j \leftarrow j+1.$$

Go to Step 2.

3. The partitioning algorithm.

3.1. General description. If the value of γ is known a priori, the algorithm presented below finds a feasible partition by a sequence of partition changes to an adjacent partition $\sigma \to \sigma \pm \{k\}$. Although it was developed in order to solve the problem of degeneracy (two or more residuals being involved in changing status simultaneously) in the continuation algorithm, it is an algorithm in its own right. It does have the ability to take advantage of a good initial estimate to find a feasible partition quickly, although if given a poor one it can perform badly. An interesting feature of the algorithm is that it is finite, despite not being monotonic in either $|\sigma|$ or $F(\mathbf{x}, \sigma)$.

Remark 3.1. The degeneracy problem here seems much rarer than in linear programming, where it can arise naturally if the data matrix has special structure, as in network problems. No naturally occurring examples have been found, although several constructed ones are given in Clark [8].

THE PARTITIONING ALGORITHM. To find a feasible partition starting from an arbitrary partition and proceeding only by adjacent partition changes.

DEFINITION. A partition P_a is $\bar{\sigma}$-feasible if $i \in \bar{\sigma} \Rightarrow |r_i(\mathbf{z}_a)| > \gamma$ and $r_i(\mathbf{z}_a)$ is of the assumed sign.

Step 1. Starting from an initial partition do until P_j is $\bar{\sigma}$-feasible $\sigma' = \sigma \cup \{k\}$, where $k \in \bar{\sigma}$ and $|r_k(z_j)| \leqq \gamma$.
Set $j = 1$, $\sigma_j = \sigma'$.

Step 2. While P_j is $\bar{\sigma}$-feasible do
If P_j is feasible then stop; else do
$\sigma_{j+1} = \sigma_j / \{k\}$ where $k \in \sigma_j$ and $|r_k(\mathbf{z}_j)| > \gamma$, $j = j+1$.

Step 3. $\mathbf{y}_{j-1} = \mathbf{z}_{j-1}$ ($\mathbf{z}_{j-1}$ satisfies $|r_k(\mathbf{z}_{j-1})| > \gamma$, $k \in \bar{\sigma}_j$)
Until P_j is $\bar{\sigma}$-feasible do
find
$\mathbf{y}_j = \alpha \mathbf{z}_j + (1-\alpha)\mathbf{y}_{j-1}$, $0 \leqq \alpha < 1$ to satisfy
$|r_i(\mathbf{y}_j)| \geqq \gamma$, $i \in \bar{\sigma}_j$ and $|r_k(\mathbf{y}_j)| = \gamma$ for at least
one $k \in \bar{\sigma}_j$.
$\sigma_{j+1} = \alpha_j \cup \{k\}$
$j = j+1$.
Go to Step 2.

3.2. Finiteness. We now show that the above algorithm terminates finitely with a feasible partition. We need two further results from Clark [8], where they occur as Lemmas 2 and 5, respectively.

LEMMA 3.1. *If P_a and P_b are adjacent partitions with $\sigma_a = \sigma_b \cup \{k\}$, then for any* $\mathbf{x}$ *satisfying* $|r_k(\mathbf{x})| = \gamma$, $F_a(\mathbf{x}) = F_b(\mathbf{x})$.

LEMMA 3.2. *If $\sigma_a = \sigma_b \cup S$, then $F_a(\mathbf{x}) \geqq F_b(\mathbf{x})$, with equality holding only if $|r_i(\mathbf{x})| = \gamma$, $i \in S$.*

THEOREM 3.3. *The partitioning algorithm terminates finitely with a feasible partition.*

Proof. As the only stopping test in the algorithm is "if P_j is feasible, stop" at Step 2, we have only to show that the algorithm does terminate.

Now within Steps 1 and 3, $|\sigma|$ is strictly increasing, and within Step 2, $|\sigma|$ is strictly decreasing, so that cycling cannot occur within any single step and the limiting partitions $\sigma = \phi$ and $\sigma = \{1, \cdots, n\}$, if they occur, satisfy the conditions of the algorithm.

We will therefore concentrate on the sequence of $\bar{\sigma}$-feasible partitions which are tested at Step 2. To show that the algorithm cannot cycle, we will show that the function values (measured at the solution of (2.1)) of this sequence of partitions decreases monotonically, so that no partition can be repeated. We will denote by P_a the most recent $\bar{\sigma}$-feasible partition found in this sequence.

At Step 2, we have $\sigma_{j+1} = \sigma_j \backslash \{k\}$ with $|r_k(\mathbf{z}_j)| \neq \gamma$, so by Lemma 3.2

$$F_{j+1}(\mathbf{z}_{j+1}) < F_j(\mathbf{z}_j) \equiv F_a(\mathbf{z}_a), \tag{3.1}$$

so that the $\bar{\sigma}$-feasible solutions produced within Step 2 are strictly decreasing.

In Step 3, the first pass gives, from Lemma 3.1 and convexity,

$$F_{j+1}(\mathbf{z}_{j+1}) \leqq F_{j+1}(\mathbf{y}_j) = F_j(\mathbf{y}_j) \leqq \alpha F_j(\mathbf{z}_j) + (1-\alpha) F_j(\mathbf{y}_{j-1}) \leqq F_j(\mathbf{y}_{j-1}). \tag{3.2}$$

But as, by Lemma 3.2, $F_j(\mathbf{z}_{j-1}) < F_{j-1}(\mathbf{z}_{j-1})$,

$$F_{j+1}(\mathbf{z}_{j+1}) \leqq F_j(\mathbf{z}_{j-1}) < F_{j-1}(\mathbf{z}_{j-1}) \equiv F_a(\mathbf{z}_a). \tag{3.3}$$

If s steps are required within Step 3 to produce a $\bar{\sigma}$-feasible partition, the argument is repeated, so that

$$
\begin{aligned}
F_{j+s}(\mathbf{z}_{j+s}) \leqq F_{j+s}(\mathbf{y}_{j+s-1}) \leqq \cdots &\leqq F_{j+s-1}(\mathbf{y}_{j+s-2}) \\
\cdots &\leqq F_{j+s-2}(\mathbf{y}_{j+s-3}) \\
&\vdots \\
&\leqq F_j(\mathbf{z}_{j-1}) < F_{j-1}(\mathbf{z}_{j-1}) \equiv F_a(\mathbf{z}_a).
\end{aligned} \tag{3.4}
$$

Thus when a $\bar{\sigma}$-feasible partition is reached, its function value is less than that of the previous $\bar{\sigma}$-feasible partition.

The theorem now follows from the finiteness of the total number of possible partitions. □

3.3. Use of the partitioning algorithm to avoid cycling in the continuation algorithm. At a change of partition in the continuation algorithm it is possible to have more than one residual as candidates for changing status.

We assume we have a feasible partition P at γ and we wish to know the feasible partition P' and γ', where $\gamma' < \gamma$ but close enough to γ so that $|r_i(\gamma', P')| \neq \gamma'$ for $|r_i(\gamma, P)| \neq \gamma$. If we knew the value of γ', we could use the partitioning algorithm, starting from P, but making γ' "close" to γ is for practical purposes an ill-defined concept. However, we have already seen in Theorem 2.3 that r_i is a linear function of γ within a partition, so that performing partition changes based on residual values at γ' is equivalent to performing the changes based on residual derivatives at γ, the residuals being updated as described in § 2.3. With this modification the partitioning algorithm provides an elegant solution to the problem of degeneracy in the continuation algorithm.

Remark 3.1. This feature was not actually built into the continuation algorithm as tested, as it added significantly to the complexity of the coding. Multiple residual changes appear to be far more uncommon than degeneracy in linear programming, and in every case when it did occur (all artificially and deliberately constructed), it was handled adequately by a natural treatment, i.e. by assuming that all such residuals did indeed change status.

3.4. Implementation. The structure of the partitioning algorithm has been given in § 3.1. Most of it is straightforward to implement, but it appears that a subproblem of the type (2.1)

$$
\min F(\mathbf{x}, \gamma) = \tfrac{1}{2} \sum_{i \in \sigma} r_i^2 + \sum_{i \in \bar{\sigma}} (\gamma |r_i| - \tfrac{1}{2}\gamma^2) \tag{3.5}
$$

has to be solved at each step. However, as each partition is replaced by an adjacent one, $\sigma' = \sigma \cup \{k\}$ or $\sigma \backslash \{k\}$, $\mathbf{z}(\gamma, P')$ and $\mathbf{r}(\gamma, P')$ may be obtained by updating $\mathbf{z}(\gamma, P)$ and $\mathbf{r}(\gamma, P)$ rather than by solving (2.1) each time.

Following the analysis of § 2.3, but starting from (2.2) rather than (2.3) the updating formulae become

$$
r_j' = r_j \mp (\mathbf{w}_j')_\alpha (\mathbf{w}_k')_\alpha \theta_k (r_k - \gamma), \tag{3.6}
$$

$$
(\mathbf{z}')_j = (\mathbf{z})_j \mp (\mathbf{w}_{n+j}')_\alpha (w_k')_\alpha \theta_k (r_k - \gamma). \tag{3.7}
$$

The first solution of (2.1) will provide the initial factorisation $A_\sigma^T A_\sigma = MM^T$, and updating at each iteration may be implemented by using equations (2.10) to (2.13), (3.6) and (3.7).

The theory given above does not guarantee finite termination when τ, the scaling factor, has to be estimated at each iteration (and this is a major advantage of the continuation method). However, when there is confidence in the initial estimate of τ a reasonable approach could be to assume its accuracy for all iterations and only re-estimate it again when the optimum is found, restarting then if necessary.

4. Numerical results. Both algorithms were tested using randomly generated data. For the continuation algorithm, examples with a known l_1 estimate were generated in the following manner. An $(n-p-1)\times(p+1)$ matrix A was generated using uniformly distributed random numbers. Then a $p\times(p+1)$ matrix was prepended to it as rows 1 to m, this matrix being chosen so that $\mathbf{z}=\mathbf{e}$ solved the equations $r_i=0$, $i=1,\cdots,p$. Finally, a row was added so that $\lambda_i=\frac{1}{2}$, $i=1,\cdots,p$ satisfied the l_1 optimality criterion

$$\sum_{i=1}^{p}\lambda_i\mathbf{a}_i+\sum_{i=p+1}^{n}\theta_i\mathbf{a}_i=0,$$

where $\theta_i=\operatorname{sgn}(\sum_{j=1}^{p}A_{ji}-A_{p+1,i})$. Thus at $\gamma=0$, $\sigma_0=1,\cdots,p$ and $\mathbf{z}=\mathbf{e}$.

These examples were then solved using the continuation algorithm by allowing γ to reduce to zero, and the results checked by comparing them with the known solution of the l_1 problem. In only one case was the number of iterations greater than $n-|\sigma_0|$, and then only by two. For this reason we give only times in Table 1. The times are internal CPU times for a DEC 10 computer, and display the expected $O(np)$ and $O(np^2)$ relationships for time/iteration and total time, respectively.

For the partitioning algorithm, test data was generated in a similar manner, except that a Pareto distribution ($\alpha=1.2$) was used in order to generate a longer-tailed distribution. Then values of γ were chosen for 10% and 20% outliers. In each case two starting partitions were used, a "good" one based on the l_1 estimate, and a "bad" one derived from the LS estimate. This terminology is based on the observed performance of the algorithm and corresponds to quite striking differences in behaviour. This indicates that the l_1 estimate ranks the residuals in a more satisfactory manner, possibly reflecting its well-known robustness properties.

TABLE 1
Continuation algorithm.
Total test time for 5 test runs (seconds) and time per iteration (m sec).

p \ n	50	100	200
6	2.41	9.75	40.32
	10.20	20.15	41.08
8	3.00	12.30	49.95
	12.80	25.27	50.60
10	3.50	14.50	59.21
	15.22	30.25	60.51

In Table 2 we report average number of iterations over 5 test runs (time per iteration was identical to the continuation algorithm). The results seem to indicate that if a good initial partition is available, the partitioning algorithm is able to take advantage of it, but if not then the continuation algorithm is superior.

Finally, the much-analyzed Daniel and Wood example [9, Chap. 5] was tried. This is a 4-variable, 21-observation example, with observations 1, 3, 4 and 21 generally

being regarded as outliers [2], [11]. Estimating scale at the same time, the continuation algorithm took four iterations to identify the four outliers. The range of δ $(=\gamma\tau)$ for which $\{1, 3, 4, 21\}$ compromised the outliers was [2.35, 3.09]. This corresponded to $\gamma \in [1.08, 1.22]$, rather lower than the 1.345 suggested by Holland and Welsch [13]. One of the virtues of the continuation algorithm is its reporting of outliers at all higher values of γ, so that the stopping value can be deliberately made more conservative and residuals examined at higher values of γ. A summary of the results for the Daniel and Wood data is given in Table 3.

TABLE 2
Partitioning algorithm.
Average number of iterations over 5 test runs for good (l_1) and bad (LS) starting partition.

(a) 10% outliers				(b) 20% outliers			
p \ n	20	50	100	p \ n	20	50	100
2	2.0	2.0		2	1.6	2.4	
	1.6	2.4			14.2	26.0	
5	3.2	2.4	3.4	5	2.2	2.0	5.2
	2.2	2.0	5.2		5.6	28.2	60.8
10	2.8	1.6	2.2	10	2.6	2.0	5.0
	2.6	2.0	5.0		5.6	28.4	79.2

TABLE 3
Partitioning algorithm, Daniel and Wood data.

Lower end of range of $(\gamma\tau)$	τ (estimate of scale)	γ (tuning constant)	Observation becoming outlier
7.24	2.76	2.62	21
5.82	2.68	2.19	4
4.09	2.74	1.49	3
3.09	2.53	1.22	1
2.35	2.17	1.08	13

The partitioning algorithm on the Daniel and Wood data took one iteration for $\gamma\tau = 3.09$ and two for $\gamma\tau = 2.35$ when the l_1 partition was used, and two and eight respectively, starting from the LS partition.

5. Conclusions. The continuation algorithm presented in § 2 appears to be suitable for most situations, particularly where there are relatively few outliers expected. It has the ability to iteratively estimate scale, requires only $2p(n+p)$ multiplications and additions per iteration, and the number of iterations is almost always equal to the number of outliers. Further, it is a finite algorithm giving an exact solution. The implementation given is stable for all realistic models. This appears to compare favourably with the iterative schemes reviewed in § 1, which do not estimate scale iteratively and do not in general give exact solutions. Further, only Huber's method has a comparable number of operations per iteration, but it, like the IRLS method, suffers from slow convergence.

The partitioning algorithm of § 3 shares the efficiency and stability properties of the continuation algorithm and is again finite. However, it does rely rather heavily on

a good initial estimate. Its use would seem to be restricted to models where one is available, and where there is confidence in the initial estimate of scale.

Acknowledgments. This paper is based on part of the doctoral thesis of one of the authors (Clark) supported by an Australian National University Scholarship. The authors with to thank the referee for careful comment and a simpler proof of Theorem 2.6.

REFERENCES

[1] D. F. ANDREWS (1971), *Significance tests based on residuals*, Biometrika, 58, pp. 139–148.
[2] ——— (1974), *A robust method for multiple linear regression*, Technometrics, 16, pp. 523–531.
[3] A. E. BEATON AND J. W. TUKEY (1974), *The fitting of power series, meaning polynomials, illustrated on band-spectroscopic data*, Technometrics, 16, pp. 147–185.
[4] J. B. BIRCH (1980), *Effects on the starting value and stopping rule on robust estimates obtained by iterated weighted least squares*, Comm. Statist., B9(2), pp. 133–140.
[5] ——— (1980), *Some convergence properties of iterated reweighted least squares in the location model*, Comm. Statist., B9(4), pp. 359–369.
[6] R. H. BYRD AND D. A. PYNE (1979), *Some results on the convergence of the iteratively reweighted least squares algorithm for robust regression*, Proc. Statistical Computing Section, American Statistical Association, pp. 87–90.
[7] D. I. CLARK (1981), *Finite algorithms for linear optimization problems*, Doctoral thesis, Australian National University, Canberra.
[8] ——— (1982), *The mathematical structure of Huber's M-estimator*, this Journal, 6 (1985), pp. 209–219.
[9] C. DANIEL AND F. S. WOOD (1971), *Fitting Equations to Data*, John Wiley, New York.
[10] J. F. GENTLEMAN AND M. B. WILK (1975), *Detecting outliers* II, Biometrics, 31, pp. 387–410.
[11] P. E. GILL, G. H. GOLUB, W. MURRAY AND M. A. SAUNDERS (1974), *Methods for modifying matrix factorisations*, Math. Comp., 28, pp. 505–535.
[12] T. P. HETTMANSPERGER AND J. W. MCKEAN (1977), *A robust alternative based on ranks to least squares in analyzing linear models*, Technometrics, 19, pp. 275–284.
[13] P. W. HOLLAND AND R. E. WELSCH (1977), *Robust regression using iteratively reweighted least squares*, Comm. Statist., A6(9), pp. 813–827.
[14] P. J. HUBER (1964), *Robust estimation of a location parameter*, Ann. Math. Statist., 35, pp. 73–101.
[15] ——— (1973), *Robust regression: asymptotics, conjectures and Monte Carlo*, Ann. Statist., 1, pp. 799–821.
[16] ——— (1977), *Robust Statistical Procedures*, CBMS Regional Conference Series in Applied Mathematics 27, Society for Industrial and Applied Mathematics, Philadelphia.
[17] P. J. HUBER AND R. DUTTER (1974), *Numerical solution of robust regression problems*, COMPSTAT 1974, Proc. Symposium on Computational Statistics, G. Brushmann, ed., Physike Verlag, Berlin, pp. 165–172.
[18] J. H. WILKINSON AND C. REINSCH (1971), *Linear Algebra*, Springer-Verlag, Berlin.

SIAM J. SCI. STAT. COMPUT.
Vol. 7, No. 1, January 1986

006

NUMERICAL METHODS FOR ROBUST REGRESSION: LINEAR MODELS*

DAVID F. SHANNO† AND DAVID M. ROCKE†

Abstract. This paper considers the robust regression problem, in which observations with large residuals are given less weight in the analysis. An auxiliary scale estimate is needed in this problem to define which residuals are large. We develop variants of Newton's method which allow for simultaneous adjustment of the scale factor and regression coefficients, and which converge superlinearily to both estimates for different loss functions. Computational results are given for both the Huber and biweight loss functions with scales obtained from the median absolute residual and the Winsorized residual variance.

Key words. Huber-Dutter algorithm, iteratively reweighted least squares, Newton's method, quasi-Newton methods, robust regression

1. Introduction.

The standard linear regression model is

$$y = X\beta + \varepsilon, \tag{1}$$

where X is $m \times n$, y and ε are $m \times 1$, and β is $n \times 1$. Denoting by r_i the residual for the ith row of (1), we have

$$r_i = y_i - \sum_{j=1}^{n} x_{ij}\beta_j. \tag{2}$$

The standard least squares regression model minimizes

$$F(\beta) = \sum_{i=1}^{m} r_i^2 \tag{3}$$

as a function of β.

Since each residual in (3) enters quadratically, points with large residuals have the major controlling effect on the results. M-estimate robust regression (Andrews [1], Beaton and Tukey [3], Holland and Welsch [8], Huber [10] and [11]) provides a means of estimating B that is less sensitive to large residuals that may be unrepresentative of the majority of the data. The weight that is accorded to each data point is a function of the absolute magnitude $|r_i|$ of the residual. In order to specify what a "large" residual is, a scale factor σ must be determined as a comparison standard. For example, given a scale constant k, for a given estimate β^* of β one may downweight points for which $|r_i| > k\sigma$.

If the scale σ is known, robust regression can be viewed as minimizing

$$\sum_{i=1}^{m} \rho\left(\frac{r_i}{k\sigma}\right). \tag{4}$$

For ordinary least squares, $\rho(t) = t^2$, and for robust regression some function rising less than quadratically is chosen [8]. Two choices seem most commonly adopted, and these are selected for further consideration in this paper. The first [9] is due to Huber and is defined by

$$\rho(t) = \begin{cases} \frac{1}{2}t^2, & |t| \leqq 1, \\ |t| - \frac{1}{2}, & |t| > 1. \end{cases} \tag{5}$$

* Received by the editors November 8, 1983, and in revised form April 15, 1984.
† Graduate School of Administration, University of California Davis, Davis, California 95616.

This function, being linear in the tails, provides less vulnerability to outliers than least squares. The second is due to Beaton and Tukey [3] and is defined by

$$\rho(t)=\begin{cases}\frac{1}{6}(1-(1-t^2)^3), & |t|\leqq 1,\\ \frac{1}{6}, & |t|>1\end{cases} \tag{6}$$

and is generally called the biweight in the literature.

When the scale σ is unknown, the problem becomes significantly more complex since the estimate of β depends on σ and the estimate of σ depends on β. There are two methods for scale estimation commonly used in the robustness literature. The first [2] is based on the median of the absolute values of the residuals, called the MAD scale. The second is Huber's Proposal 2 [9], which is used only with Huber's ρ, and is defined as the solution to the estimating equation

$$\frac{1}{m-n}\left(\sum_{|r_i|\leqq k\sigma} r_i^2+pk^2\sigma^2\right)=b\sigma^2 \tag{7}$$

where p is the number of r_i with $|r_i|>k\sigma$ and where b is the asymptotic expectation of the left-hand side for standard normal residuals. A third possibility, which is introduced here, is to use (7) with $k=.6745$ for any choice of ρ, so that about half of the points would be given full weight in the scale if the data were normally distributed. This is a kind of Winsorized variance [16], so this will be called the Winsorized scale. To simultaneously estimate the Proposal 2 scale σ, while minimizing $\rho(\cdot)$ given by (5), Huber and Dutter [10] noted that this can be done by minimizing

$$Q(\beta,\sigma)=\sum_{i=1}^{m}\rho\left(\frac{r_i}{k\sigma}\right)k\sigma+\frac{a\sigma}{k}, \tag{8}$$

with

$$a=\frac{(m-n)b}{2}, \tag{9}$$

where $b=E_\Phi\chi(t)$, the expected value of $\chi(t)=2kt\rho'(t/k)-2k^2\rho(t/k)$ for a standard normal argument t. Note that, for ρ defined by (5), as it is here,

$$2kt\rho'\left(\frac{t}{k}\right)-2k^2\rho\left(\frac{t}{k}\right)=\begin{cases}t^2, & |t|\leqq k,\\ k^2, & |t|>k.\end{cases} \tag{10}$$

The first order conditions $\partial Q/\partial\beta=0$ lead to the usual normal equations for the loss function ρ, while the first order condition $\partial Q/\partial\sigma=0$ defines σ via the equation

$$-k\sum_{i=1}^{m}\rho'\left(\frac{r_i}{k\sigma}\right)\frac{r_i}{k\sigma}+k\sum_{i=1}^{m}\rho\left(\frac{r_i}{k\sigma}\right)+\frac{a}{k}=0. \tag{11}$$

To see how this defines σ, if $I=\{i|1\leqq i\leqq m \text{ and } |r_i/k\sigma|\leqq 1\}$, then (11) is equivalent to

$$k^2\sum_{i\in I}\left(\frac{r_i}{k\sigma}\right)^2+k^2p=2a, \tag{12}$$

where p is the number of residuals which satisfy $|r_i/k\sigma|>1$, and this is in turn equivalent to (7).

The Huber-Dutter algorithm is then to find initial estimates $\beta^{(0)}$ and $\sigma^{(0)}$ to $\hat{\beta}$ and σ. For these estimates the residuals are computed, and a new estimate to σ made by

$$\sigma_{j+1}^2=\frac{1}{2a}\left(\sum_{i\in I} r_i^2+pk^2\sigma_j^2\right). \tag{13}$$

The residuals are then Winsorized by

$$\hat{r}_i = \rho'\left(\frac{r_i}{k\sigma_{j+1}}\right) k\sigma_{j+1}, \tag{14}$$

and new estimates $\beta^{(j+1)}$ computed by solving the normal equations

$$X^T X\tau = X^T\hat{r}, \tag{15}$$

$$\beta^{(j+1)} = \beta^{(j)} + \tau. \tag{16}$$

The process continues until successive iterates on both β and σ have converged to desired accuracy. Huber and Dutter have shown that this algorithm always converges due to the special nature of the object function defined by (8). However, this convergence only holds for ρ defined by (5) and the Proposal 2 scale σ defined by (13).

Now, if one wished to use the Huber loss function, but the MAD scale for σ, a scheme similar to the Huber–Dutter algorithm can certainly be devised, with σ_{j+1} a multiple of median $|r_i(\beta_j)|$, but the convergence proof fails. The method encounters even more difficulty if a loss function other than the Huber loss function is desired. If the biweight ρ (6) is used in (8), inspection shows that for σ sufficiently small, $\rho(r_i/k\sigma) = \frac{1}{6}$ for all r_i, hence $\sigma = 0$ minimizes (8) for any finite β.

Also, due to the nonconvexity of the biweight loss function, even for fixed σ (i.e., $\sigma_{j+1} = \sigma$ for all j), the sequence (14)–(16) need not converge to estimates β; and if convergence occurs, it can be very slow.

If the scale were fixed, a method of accelerating convergence for a non-Huber object function is to use Newton's method to solve the first order conditions. Here the sequence of iterates is defined by

$$\beta^{(j+1)} = \beta^{(j)} + (X^T S X)^{-1} X^T R, \tag{17}$$

where R is the $m \times 1$ vector $(\rho'(r_i/k\sigma))$ and S is the $m \times m$ matrix diag $(\rho''(r_i/k\sigma)/k\sigma)$. A variant of this which avoids the calculation of $\rho''(\cdot)$ is the iteratively reweighted least squares (IRLS) method defined by

$$\beta^{(j+1)} = \beta^{(j)} + (X^T S^* X)^{-1} X^T R, \tag{18}$$

where S^* is the $m \times m$ matrix diag $(\rho'(r_i/k\sigma)/r_i)$. For a full discussion of these methods, the reader is referred to Holland and Welsch [8] and Byrd and Pyne [4].

Two difficulties arise with both of these methods. First, convergence depends on good initial estimates $\beta^{(0)}$. More important, Holland and Welsch and Byrd and Pyne both assume σ is fixed before the iterative schemes defined by (17) or (18) are begun, and never changed. As both scale factors which have been discussed in this paper depend upon β, if the initial estimate $\beta^{(0)}$ is poor, the scale factor may well be poor, badly affecting the robustness of the estimates.

To attempt to answer this latter point, Huber [11] suggests iteratively recomputing the scale by a scheme similar to (13), even though this may not converge. It is also important to know that even with the Huber object function convergence has a linear rate because the iteration (13) converges linearly. This relatively slow rate will also apply to iterative schemes for recomputing σ by (13) for other object functions.

It is the purpose of this paper to demonstrate ways to regularly reestimate σ_j for each new estimate $\beta^{(j)}$, and to show how to incorporate this estimate into algorithms which exhibit a superlinear rate of convergence in both β and σ. Further, we propose a method which has achieved convergence, and eventual superlinear convergence, for bad initial estimates $\beta^{(0)}$ and σ_0 on a variety of problems.

To this end, § 2 will deal with the explicit algorithms for computing σ as a function of β. Section 3 will discuss methods for solution of the robust regression problem in different cases. Section 4 contains computational results indicating the methods derived in § 3 are computationally robust and efficient.

2. Computation of the scale factor. In this section, we consider the problem of determining the scale factor explicitly as a function of β. The solution to this is immediate if the MAD scale is used. The algorithm used for this in the computational results contained in § 4 is, for any given β, compute the absolute values of the residuals by (2), and then to sort these absolute values using a heapsort algorithm [13], an algorithm which sorts an array of length m in $o(m \log m)$ operations. From the sorted array, the scale factor σ is calculated as

$$\sigma = (|r_{m_1}| + |r_{m_2}|)/1.349 \tag{19}$$

where $m_1 = [m/2+1]$ and $m_2 = [(m+1)/2]$ and where the divisor is chosen to guarantee asymptotic equivalence with the standard deviation for normally distributed data. (The theoretical MAD for the standard normal distribution is 1.349.) For the algorithms of § 3, it will be important to be able to calculate $\partial\sigma/\partial\beta_j$, which can be derived immediately from (19) as

$$\frac{\partial\sigma}{\partial\beta_j} = (\operatorname{sgn}(r_{m_1})x_{m_1,j} + \operatorname{sgn}(r_{m_2})x_{m_2,j})/1.349. \tag{20}$$

This derivative exists unless half or more of the residuals are exactly zero, but it is discontinuous at points at which a residual defining the median changes.

If we wish to use the Proposal 2 or Winsorized scale, however, the algorithm for explicitly calculating it is more complex. Note that if the index set I is known, (13) can be solved explicitly for the convergent value of σ as

$$\sigma^2 = \sum_{i \in I} \frac{r_i^2}{2a - pk^2}. \tag{21}$$

However, I depends on σ. In order to calculate the correct index set I and σ correspondingly, the following algorithm is used:

(i) Sort the residuals r_i in ascending order of $|r_i|$;
(ii) Set $\sigma_0^2 = \sum_{i=1}^{m} r_i^2/2a$, $p=0$, $j=m$, $l=0$;
(iii) If $|r_j/k\sigma_l| \leqq 1$, then $\sigma = \sigma_l$, else go to (iv);
(iv) Until $|r_j/k\sigma_l| \leqq 1$, $j = j-1$, $p = p+1$;
(v) $l = l+1$, $\sigma_l^2 = \sum_{i=1}^{m-p} r_i^2/(2a-k^2p) = \sum_{i=1}^{j} r_i^2/(2a-k^2p)$,
(vi) Go to iii).

To show the above algorithm converges to a scale factor σ satisfying (21), we demonstrate three propositions.

PROPOSITION 1. *Let σ_l be defined by* (v). *If* $|r_j/k\sigma_l| > 1$, *then*

$$\sum_{i=1}^{j-1} \frac{r_i^2}{2a - k^2(p+1)} > 0. \tag{22}$$

Proof. $|r_j/k\sigma_l| > 1$, implies by the definition of σ_l

$$r_j^2 > k^2\left(\sum_{i=1}^{j} \frac{r_i^2}{(2a-k^2p)}\right), \tag{23}$$

or

$$(2a-k^2p)r_j^2 > k^2\sum_{i=1}^{j-1} r_i^2 + k^2 r_j^2, \tag{24}$$

or

$$(2a-k^2(p+1))r_j^2 > k^2\sum_{i=1}^{j-1} r_i^2. \tag{25}$$

Since the right-hand side of (25) is clearly nonnegative, it follows that

$$(2a-k^2(p+1)) > 0, \tag{26}$$

which trivially proves (22).

PROPOSITION 2. $|r_j/k\sigma_l| > 1$ *implies*

$$\sum_{i=1}^{j-1} \frac{r_i^2}{2a-k^2(p+1)} < \sum_{i=1}^{j} \frac{r_i^2}{2a-k^2p}. \tag{27}$$

Proof. By Proposition 1, the denominators of both sides of (27) are positive, so it is sufficient to show

$$(2a-k^2p)\sum_{i=1}^{j-1} r_i^2 < (2a-k^2(p+1))\sum_{i=1}^{j} r_i^2. \tag{28}$$

Cancellation of terms from each side reduces (28) to

$$(2a-k^2(p+1))r_j^2 > k^2\sum_{i=1}^{j-1} r_i^2, \tag{29}$$

or

$$(2a-k^2p)r_j^2 > k^2\sum_{i=1}^{j} r_i^2, \tag{30}$$

or

$$r_j^2 > k^2\sum_{i=1}^{j} \frac{r_i^2}{(2a-k^2p)}$$

which is precisely the condition that $|r_j/k\sigma_l| > 1$.

PROPOSITION 3. $2a-(m-1)k^2 > 0$ *implies*

$$r_1^2 < k^2 \frac{r_1^2}{2a-(m-1)k^2}. \tag{31}$$

Proof. Trivially, $2a-(m-1)k^2 > 0$ implies (31) is true if

$$2a-(m-1)k^2 < k^2, \tag{32}$$

or

$$mk^2 > 2a. \tag{33}$$

From (9), (34) is true if

$$k^2 > b, \tag{34}$$

which follows from (10).

Proposition 2 shows that the iterates σ_l are monotonically decreasing, so $|r_j/k\sigma_{l-i}|>1$ implies $|r_j/k\sigma_l|>1$, so no residual need be considered twice. Proposition 3 together with Proposition 1 show that there must be a fixed point to the algorithm, so convergence must occur.

In fact, this algorithm for computing the Huber-Dutter σ is also $o(m \log m)$, as the major work is sorting the residuals. The partial sums can be stored in a vector, so this segment of the algorithm is $o(m)$.

Finally, we note that once (21) has been computed, we can again compute $\partial\sigma/\partial\beta_j$ by

$$2\sigma\frac{\partial\sigma}{\partial\beta_j}=\frac{1}{2a-pk^2}\sum_{i\in I}2r_jx_{ij}. \tag{35}$$

This derivative, as with median, is discontinuous at points where the index set I changes.

3. Methods of solution. The formulation of the robust regression problem that is used here is to find a (local) minimum of

$$\sum_{i=1}^{m}\rho\left(\frac{r_i}{k\sigma}\right) \tag{36}$$

where σ is defined by (19) or (21), by solving the first order conditions

$$\frac{1}{k\sigma}\sum_{i=1}^{m}\rho'\left(\frac{r_i}{k\sigma}\right)x_{ij}=0, \qquad j=1,2,\cdots,n, \tag{37}$$

together with a scale equation (19) or (21).

Solving the first order conditions (37) with a scale factor explicitly determined as a function of β, either by (19) or (21), is conceptually quite simple. The first order conditions (37) can be viewed as n equations in $n+1$ unknowns β_i, $i=1,\cdots,n$ and σ. The equation defining the scale factor can then be appended as the $(n+1)$st defining equation, yielding a full system of $n+1$ equations in $n+1$ unknowns.

While conceptually this is simple, computational experience has shown that it is more efficient to use the explicit formulation for σ as a function of β to eliminate σ from (37) then utilize either Newton's method or IRLS to solve the resulting system. This has proved quite efficient in practice, both for the Huber and biweight loss functions and with both scale factors discussed in § 2, provided that good initial estimates $\beta^{(0)}$ are available.

This paper considers two different loss functions and two different scales (and a third for the Huber loss function). Not surprisingly, different algorithms prove most efficient depending on the problem to be solved and the degree of contamination of the data.

The simplest problem to solve is to estimate β and σ using the Huber loss function and the Huber-Dutter scale σ. The Huber-Dutter algorithm has been described in equations (13)-(16). Its major advantage is that the normal matrix X^TX used only need be formed and factored once, while its major disadvantage is the previously mentioned linear rate of convergence to σ. Another approach is to optimize directly $Q(\beta,\sigma)$ as defined by (8), with σ treated as an independent variable. Dutter [6] compared such an approach with the Huber-Dutter method, with results uniformly favoring Huber-Dutter. The algorithm he used in the comparison was a quasi-Newton method, specifically the Davidon-Fletcher-Powell implementation described in Himmelblau [7].

As this particular quasi-Newton method is quite out of date, we tested a more up-to-date code, the Broyden-Fletcher-Goldfarb-Shanno (BFGS) code of Shanno and Phua [15], with precisely the opposite results. In view of this, we include a brief description of the algorithm. The reader interested in a fuller development, with complete theoretical details, is referred to Dennis and Schnabel [5].

Briefly, the BFGS method minimizes any object function $f(t)$ sequentially by choosing $t^{(0)}$ and an initial estimate $H^{(0)}$ (usually the identity matrix multiplied by an appropriate scalar) to the inverse Hessian matrix of f and proceeds by

$$t^{(j+1)} = t^{(j)} - \alpha^{(j)} H^{(j)} g^{(j)}, \tag{38}$$

$$H^{(j+1)} = H^{(j)} - \frac{H^{(j)} y^{(j)} \delta^{(j)T} + \delta^{(j)} y^{(j)T} H^{(j)}}{\delta^{(j)T} y^{(j)}} \left(1 + \frac{y^{(j)T} H^{(j)} y^{(j)}}{\delta^{(j)T} y^{(j)}}\right) \frac{\delta^{(j)} \delta^{(j)T}}{\delta^{(j)T} y^{(j)}}, \tag{39}$$

where $\delta^{(j)} = -\alpha^{(j)} H^{(j)} g^{(j)}$, $g^{(j)} = \nabla f(t^{(j)})$, $y^{(j)} = g^{(j+1)} - g^{(j)}$, and $\alpha^{(j)}$ is an appropriately chosen scalar.

The salient point of this method within the context of this paper is that $H^{(i)}$ is $n \times n$, where n, the number of parameters, is usually quite small compared to m, the number of data points. As both the Huber-Dutter and BFGS must compute the equivalent of the gradient of the Huber loss function, an order $m \times n$ calculation, this should dominate the overhead of the quasi-Newton method, which is admittedly higher than the single factorization of $X^T X$ of the Huber-Dutter method. The advantage of the BFGS method lies in a local superlinear rate of convergence to σ as well as β, which explains the computational results of the next section.

The case of a general object function and arbitrary scale must be handled differently. In order to solve the first order conditions with the augmented equation describing the scale factor, we note again that the first order conditions are

$$f_j(\beta, \sigma) = \sum_{i=1}^{m} \rho'\left(\frac{r_i}{k\sigma}\right) x_{ij} = 0, \qquad j = 1, \cdots, m. \tag{40}$$

Newton's method for solution of this system given the augmented equation

$$\sigma = h(\beta) \tag{41}$$

can be determined by first computing

$$\frac{\partial f_j}{\partial \beta_l} = \sum_{i=1}^{m} \rho''\left(\frac{r_i}{k\sigma}\right) x_{ij} \left(-\frac{x_{il}}{k\sigma} - \frac{r_i}{k\sigma^2} \frac{\partial \sigma}{\partial \beta_l}\right), \tag{42}$$

$$= \frac{1}{k\sigma} \sum_{i=1}^{m} \rho''\left(\frac{r_i}{k\sigma}\right) x_{ij} \left(-x_{il} - \frac{r_i}{\sigma} \frac{\partial \sigma}{\partial \beta_l}\right). \tag{43}$$

In matrix terms, this yields the expression

$$\left[\frac{\partial f_i}{\partial \beta_l}\right] = -\frac{1}{k\sigma} X^T D(X + Y), \tag{44}$$

where D is the diagonal matrix

$$D = \left[\rho''\left(\frac{r_i}{k\sigma}\right)\right], \tag{45}$$

and Y is the $m \times n$ matrix

$$Y = \left[\frac{r_i}{\sigma} \frac{\partial \sigma}{\partial \beta_l}\right]. \tag{46}$$

The terms $\partial\sigma/\partial\beta_l$ are derived from (20) or (35) depending upon which scale is used to define $h(\beta)$ in (41). The Newton sequence equivalent to (18) becomes

$$\beta^{(j+1)} = \beta^{(j)} + k\sigma(X^T D(X+Y))^{-1} X^T R, \tag{47}$$

where R is the $m \times 1$ vector $(\rho'(r_i/k\sigma))$.

Note that the IRLS method is equally applicable here, as it entails solely replacing D with the matrix

$$D = \left[\rho'\left(\frac{r_i}{k\sigma}\right) \Big/ \left(\frac{r_i}{k\sigma}\right)\right]. \tag{48}$$

A comparison of (47) and (48) with (17) and (18) shows the effect of explicitly incorporating the scale into the iteration. For the numerical tests of § 4, we tested only Newton's method, as we used only the Huber and biweight loss functions, and for these the true Newton D is as easily computed as (48). For more complex loss functions, and particularly for nonlinear models, however, (48) may well be preferable to (47) and remains for further testing.

As a final note on this section, in order to prevent divergence of Newton's method, the sequence (47) was modified to

$$\beta^{(j+1)} = \beta^{(j)} + (\tfrac{1}{2})^l k\sigma(X^T D(X+Y))^{-1} X^T R, \tag{49}$$

where $l \geqq 0$ was the smallest integer which guaranteed $\sum_{j=1}^{m} f_j^2(\beta, \sigma)$ was reduced at each step. Simplistic line searches such as this are known not to converge for Newton's method with bad initial estimates, but are a useful safeguard. A more complex trust region approach could be implemented (Moré and Sorensen [14], for example, or Welsch and Becker [17]), but may not be necessary in view of the computational results of the next section.

4. Computational results. To test the various methods discussed in this paper, a set of 30 randomly generated test problems of varying dimensions were generated. Data were generated from the model

$$y = X\beta \tag{50}$$

where $x_{i1} = 1$, $i = 1, \cdots, m$, and x_{ij} are uniformly randomly distributed, $-1 \leqq x_{ij} \leqq 1$, $i = 1, \cdots, m$, $j = 2, \cdots, n$. Examination of the sequence of Newton iterates (47), quasi-Newton iterates (38)-(39), and Huber-Dutter iterates (15)-(16) shows that the sequence of iterates is independent of the optimal β if the initial estimates $\beta^{(0)}$ are chosen so as to assure the initial residuals, which can be made solely a function of the error term ε, remain constant for translations of β. Hence, the value $\beta = 0$, $y = 0$, was always used. The error vector ε consisted of independent standard normal data of which some fraction were multiplied by an expansion factor. Both the expansion factor and the fraction of points affected were program input variables that will be discussed in more detail for specific cases. The sequence of iterates was independent of scale in the sense that multiplying all elements of the error vector by the same factor did not change the iterates.

In order to ensure that the starting residuals remained independent of the β chosen in the model (50), the least squares estimate to β,

$$\beta^{(0)} = (X^T X)^{-1} X^T y \tag{51}$$

was used as a starting value.

The thirty problems tested were divided into three sets of ten. The first had $n = 5$, $m = 30$, the second had $n = 10$, $m = 100$, and the third had $n = 15$, $m = 200$. For each set of ten, five problems were considered "good," which meant 10% of the data points were outliers, and the standard deviation of the outliers was 5. The remaining five problems in each set were "bad," meaning that 20% of the points were outliers and the outliers had a standard deviation of 15. As the computational results will show, this differential was sufficient to produce significantly different computational behavior on the two classes of problems.

All five problems within each size and division were identical except for the actual random deviates ε and random coefficient matrices X. The computational results we report in this section consequently summarize the results into six groups of five runs each, designated GOOD 5, BAD 5, GOOD 10, BAD 10, GOOD 15, and BAD 15 to identify the number of parameters determined and the distortion of the data.

All tests were run on a VAX 11/750, using FORTRAN 77. The CPU times quoted were obtained by running in batch mode, with all calculations in double precision. The relatively high CPU times are accounted for by the fact that the particular machine did not have a floating point accelerator.

For the Huber-Dutter algorithm and the BFGS algorithm applied to (8) convergence was achieved when the norm of the gradient of the object function satisfied

$$|g| \leqq \varepsilon \max(1, \tau).$$

where $\tau = (\sum_{i=1}^{m} \beta_i^2 + \sigma^2)^{1/2}$ and $\varepsilon = .00001$. Hence, identical convergence criteria were used for all comparable cases.

When the first order conditions were solved, convergence was achieved when

$$\left(\sum_{i=1}^{m} f_i^2\right)^{1/2} < \varepsilon, \quad \text{with again } \varepsilon = .00001.$$

The first test was the BFGS code of Shanno and Phua [15] versus the Huber-Dutter algorithm to minimize $Q(\beta, \sigma)$ of (8). Here k was always .6745, $b = .2988$. The least-squares estimates were used as starting values. The results are summarized in Table 1, where ITER is the number of search directions computed, IFUN the number of times $Q(\beta, \sigma)$ was evaluated, and CPU is the CPU time in seconds.

TABLE 1
Comparison of Huber-Dutter and BFGS for minimizing $Q(\beta, \sigma)$ of (8) with least squares start, $k = .6745$.

	BFGS			Huber-Dutter		
Problem	ITER	IFUN	CPU	ITER	IFUN	CPU
GOOD 5	74	83	14.22	178	178	17.27
BAD 5	83	95	15.00	284	284	23.24
GOOD 10	76	85	36.86	192	192	62.09
BAD 10	77	87	37.57	240	240	99.35
GOOD 15	99	109	102.04	220	220	178.43
BAD 15	84	99	94.19	248	248	199.08

Examination of the results verifies that the BFGS requires substantially fewer iterations and function evaluations, a point also noted by Dutter [6]. However, contrary to Dutter's results, the BFGS also requires less CPU time, with the difference in favor of the BFGS growing with problem size. Further, the BFGS efficiency appears indepen-

dent of the degree of contamination of the data, whereas the Huber-Dutter efficiency clearly decreases with increased data contamination. Finally, as all CPU times include problem generation times, constant in both cases, the relative efficiencies solely of the algorithms would be even more favorable to the BFGS, a point attributable almost entirely to the superlinear rate of convergence to σ as opposed to the Huber-Dutter linear rate of convergence.

Table 2 examines the performance of Newton's method to solve the first order conditions with the Huber object function, with both the median and Winsorized scales. Again the least squares start, $k = .6745$, $b = .2988$ were used in all cases. Here ITER is the number of Newton directions computed, IFUN the number of function evaluations needed to assure $\sum_{i=1}^{n} f_i^2$ was reduced.

TABLE 2
Solution of first order conditions for Huber object functions by Newton's method with least squares start, $k = .6745$, and two different scales.

	Winsorized scale			Median scale		
Problem	ITER	IFUN	CPU	ITER	IFUN	CPU
GOOD 5	18	31	18.87	19	32	18.05
BAD 5	20	50	20.87	30	74	23.10
GOOD 10	15	25	102.64	22	33	118.83
BAD 10	20	41	126.66	23	44	125.96
GOOD 15	15	25	325.70	20	30	437.46
BAD 15	21	42	492.79	22	43	477.02

A striking initial conclusion can be drawn from the first half of this table. The β computed using the Winsorized scale is identical in this case to the β computed in Table 1 since $k = .6745$, yet the execution times are uniformly larger and rise much more quickly with problem size. This result is attributable to the fact that while the Newton matrix, like the quasi-Newton matrix, is $n \times n$, it is of necessity computed in order mn^2 operations, while the need only to compute gradients for quasi-Newton methods make the method of order mn. Similar results are reported by Dutter [6].

Unfortunately, the inability to derive a function $Q(\beta, \sigma)$ similar to (8) for arbitrary loss functions and scale factors leaves no alternative to Newton's method for other than the Huber loss function with the Proposal 2 scale.

The remaining results of interest in Table 2 are that despite the nondifferentiability of both the median and Winsorized scales at select points, this caused no problems computationally, and all cases with both scales converged. The computation times are indistinguishable, and the relative efficiency of the scales remains for further testing. Also, Newton's method definitely shows that its efficiency is dependent on the initial estimates, which will be worse in the BAD than in the GOOD cases; this has long been known in optimization literature.

Finally, Newton's method was tested on the biweight object function, with two methods of obtaining the starting value: with least squares starting estimates directly to Newton's method, and with estimates obtained in Table 1 using the BFGS as starting estimates. For the biweight object function, $k = 6$ was used (this corresponds to $c = 9$ in [3]).

The results are summarized in Tables 3 and 4.

TABLE 3

Solution of first order conditions for biweight object functions by Newton's method with least squares start, $k = 6.0$, and two different scales.

	Winsorized scale			MAD scale		
Problem	ITER	IFUN	CPU	ITER	IFUN	CPU
GOOD 5	43	43	21.90	23	63	25.19
BAD 5	174[1]	174[1]	37.51	29[3]	153[3]	33.68
GOOD 10	24	24	124.96	19	30	150.30
BAD 10	90[2]	90[2]	279.98	20[4]	76[4]	161.39
GOOD 15	24	24	512.71	18	29	586.35
BAD 15	54	54	784.45	27	62	794.71

[1] 3 of 5 cases failed to converge.
[2] 2 of 5 cases failed to converge.
[3] 2 of 5 cases failed to converge.
[4] 2 of 5 cases failed to converge.
[5] 1 of 5 cases failed to converge.

TABLE 4

Solutions of first order conditions for biweight object function by Newton's method using $k = 6.0$ and two different scales. Starting values were Proposal 2 estimates as in Table 1 using $k = .6745$.

	Winsorized scale			MAD scale		
Problem	ITER	IFUN	CPU	ITER	IFUN	CPU
GOOD 5	9	9	23.00	13	23	29.17
BAD 5	19	37	27.96	13	22	29.83
GOOD 10	10	20	135.44	14	25	172.47
BAD 10	10	20	127.54	11	21	147.42
GOOD 15	10	20	505.98	12	23	602.58
BAD 15	10	20	474.14	17	27	677.08

* CPU times are total times, including the time to obtain the starting point, which is summarized in Table 1.

As Table 3 shows, for least squares starting estimates, 8 out of 30 cases failed to converge within the allotted time, and indeed would not have converged within any determinable time unit. All were for badly contaminated data, where least squares starting estimates are likely to be poor.

Table 4 shows that with better starting values, convergence was achieved in all cases. Results here indicate the Proposal 2 start should always be used in preference to the least squares start, both for safety and efficiency.

In conclusion, the algorithms proposed by this paper appear to show strong promise for providing efficient, convergent methods of estimating coefficients in robust regression for a variety of loss functions with a variety of dynamically chosen scales.

REFERENCES

[1] D. F. ANDREWS, (1974), *A robust method for multiple linear regression*, Technometrics, 16, pp. 523-531.

[2] D. F. ANDREWS, P. J. BICKEL, F. R. HAMPEL, P. J. HUBER, W. H. ROGERS AND J. W. TUKEY (1972), *Robust Estimates of Location: Survey and Advances*, Princeton Univ. Press, Princeton, NJ.

[3] A. E. BEATON AND J. W. TUKEY (1974), *The fitting of power series, meaning polynomials, illustrated on band-spectroscopic data*, Technometrics, 16, pp. 147-185.

[4] R. H. BYRD AND D. A. PYNE (1979), *Convergence of the iteratively reweighted least squares algorithm for robust regression*, Tech. Rept. 313, Mathematical Sciences Dept., Johns Hopkins Univ., Baltimore, MD.

[5] J. G. DENNIS, JR. AND R. B. SCHNABEL (1983), *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*, Prentice-Hall, Englewood Cliffs, NJ.

[6] R. DUTTER (1977), *Numerical solution of robust regression problems: computational aspects, a comparison*, J. Statist. Comput., Simul., 5, pp. 207–238.

[7] D. M. HIMMELBLAU (1972), *Applied Nonlinear Programming*, McGraw-Hill, New York.

[8] P. W. HOLLAND, AND R. E. WELSCH (1977), *Robust regression using iteratively reweighted least squares*, Comm. Statist., A6, pp. 813–827.

[9] P. J. HUBER (1964), *Robust estimation of a location parameter*, Ann. Math. Stat., 35, pp. 73–101.

[10] ———, (1973), *Robust regression: asymptotics, conjectures, and Monte Carlo*, Ann. Stat., 1, pp. 799–821.

[11] ———, (1981), *Robust Statistics*, John Wiley, New York.

[12] P. J. HUBER AND R. DUTTER (1974), *Numerical solution of robust regression problems*, COMPSTAT 1974 Proc. Symposium on Computational Statistics, G. Brushmann, ed., Physike Verlag, Berlin, pp. 165–172.

[13] D. E. KNUTH (1973), *The Art of Computer Programming, Vol. 3, Sorting and Searching*, Addison-Wesley, Reading, MA, pp. 145–149.

[14] J. J. MORÉ AND D. C. SORENSEN (1981), *Computing a trust region step*, ANL-81-83, Argonne National Labs., Argonne, IL, this Journal, 4 (1983), pp. 553–572.

[15] D. F. SHANNO AND K. H. PHUA (1980), *Research on algorithm 500, Minimization of unconstrained multivariate functions*, ACM Trans. Math. Software, 6, pp. 618–622.

[16] J. W. TUKEY AND D. H. MCLAUGHLIN (1963), *Less vulnerable confidence and significance procedures based on a single sample* (Trimming/Winsorization 1), Sankhyā, Ser. A, pp. 331–352.

[17] R. E. WELSCH AND R. A. BECKER (1975), *Robust regression using the dog leg algorithm*, Proc. Computer Science and Statistics Eighth Annual Symposium on the Interface, J. W. Frane, ed., Health Sciences Computing Facility, Univ. California, Los Angeles.

SIAM J. SCI. STAT. COMPUT.
Vol. 7, No. 1, January 1986

007

A NEW METHOD IN ORDER TO DETERMINE THE MOST SIGNIFICANT MEMBERS WITHIN A LARGE SAMPLE, IN PROBLEMS OF SURFACE APPROXIMATION*

MIRA BOZZINI†, FLAVIA DE TISI† AND LICIA LENARDUZZI‡

Abstract. From an initial large sample a smaller, yet representative subsample is to be determined. As an alternative to the principal component method, a new method is proposed: "domains of influence" of each datum relative to nearby data and a hierarchy of significance are determined. The properties are discussed and numerical examples are given.

Key words. subset, significance estimation, local analysis, surface approximation

1. Introduction. A problem of increasing interest that can be found in many areas of scientific applications (as for example geographic, geological, meteorological and engineering information) is this one: from an initial large sample to determine a subsample not too large and suitable to represent the examined phenomenon properly.

In [1], [7], [8] this problem has usually been solved by the principal component method. This method requires the covariance matrix, which is not always available or easily computed.

We present a new method for solving the problem. The method has been tested extensively, and § 6 gives the results of two numerical examples. The former is of academic character for illustrative purposes, the latter deals with a biomedical problem.

2. Definitions. Given $f=f(P)$: $D\subset R^q \to R$, D compact, μ a measure on D and $f\in L(\mu)$, let us consider a set $\{f(P_i)\}_1^N$ corresponding to the set S of points $\{P_i\}_1^N$. Let $f(P_i)=f_i$, $i=1,\cdots,N$.

Consider a generic point $P_i\in S$.

DEFINITION. For $K_i\in R^+$, let the K_i-neighborhood $\subseteq D$ be the largest local region $I(P_i)$ centered at P_i, with an arbitrary prefixed shape, such that

$$0<\int_{I(P_i)} |f-f_i|\, d\mu/\mu(I(P_i))<K_i.$$

This choice is made in order to account for local homogeneous variability of the function.

DEFINITION. Let

$$\delta_i=\int_{I(P_i)} |f-f_i|\phi_i(P)\, d\mu \Big/ \int_{I(P_i)} \phi_i(P)\, d\mu,$$

δ_i being an average of $|f-f_i|$ in the K_i-neighborhood and $\phi_i(P)$ a suitable smooth weight function. ϕ_i is a function with support I_i, $0\leqq\phi_i(P)\leqq 1$, decreasing with the distance from P_i. This is for taking more in account the regions closest to P_i. δ_i is an average variability.

DEFINITION 3. Let us define the quantities

$$s_{ij}=\frac{|f_i-f_j|}{\delta_j}$$

* Received by the editors July 29, 1983, and in revised form August 8, 1984.

† Istituto Matematico Università di Milano, via Saldini 50, 21033 Milano, Italy.
‡ Istituto Applicazioni Matematica ed Informatica del CNR, via Cicognara 7, 20129 Milano, Italy.

as P_j varies in $(S \cap I(P_i))$, $j \neq i$. s_{ij} gives a measure of the change toward P_i relative to the average variability of f near P_j.

3. Method. Let there be given a sample S of points $P_i \in D$, $i = 1, \cdots, N$ such that the corresponding set of function values $\{f_i\}_1^N$ well represents the behavior of the function f: $D \subseteq R^q \to R$.

The K_i-neighborhood is built for each of the points $P_i \in S$ in order to subdivide the domain D into parts (which have nonnull intersection in general) that can be thought of as being homogeneous with respect to the function behavior. An evaluation of the level of homogeneity is obtained for each neighborhood, by the function $\delta = \delta(K\text{-neighborhood}, f, \phi)$. Thus a covering of the domain D is reached by associating the value δ_i of its K_i-neighborhood to each point $P_i \in S$. It follows that each point $P_i \in S$ gets to represent its K_i-neighborhood and that the quantities s_{ij} supply information about how much the value of the function at a point can be "different" from those falling in its neighborhood. By averaging the quantities s_{ij} for the values of j with $P_j \in K_i$-neighborhood and then repeating for each $P_i \in S$, a hierarchy is given, by which it is possible to characterize the most significant points.

4. Algorithm. Call n the dimension of the subsample we want to determine; the proposed algorithm is summarized by the following steps.

1. Estimate[1] the K_i-neighborhood by a $\hat{I}(P_i)$ (with the same character as $I(P_i)$, a maximum neighborhood centered at P_i and with the same shape) such that

$$\sum_h |f_h - f_i| / m_i < K_i \quad \forall i = 1, \cdots, N,$$

where $h = h(\hat{I}_i)$ points to $P_h \in (S \cap \hat{I}(P_i))$, $h \neq i$; m_i is the number of points P_h of $\hat{I}(P_i)$, $h \neq i$; K_i is an empirical constant related to the specified application and, of course, depending on the size order of the functional values.

2. Estimate[1]

$$\hat{\delta}_i = \sum_h |f_h - f_i| \phi_i(P_h) \Big/ \sum_h \phi_i(P_h), \qquad h \neq i \quad \forall i = 1, \cdots, N,$$

ϕ with support $\hat{I}(P_i)$, $h = h(\hat{I}_i)$.

3. For each $i = 1, \cdots, N$ estimate the quantities

$$\hat{s}_{ij} = \frac{|f_i - f_j|}{\hat{\delta}_j}, \qquad j = j(\hat{I}_i), \quad j \neq i.$$

4. For each $i = 1, \cdots, N$, calculate the mean value of the $\hat{s}_{ij}$

$$\hat{s}_i = \sum_j \hat{s}_{ij} / m_i, \qquad j = j(\hat{I}_i), \quad j \neq i.$$

5. Choose the point to be included into the subsample; this is the point P_h such that:

$$\hat{s}_h = \max_{1 \leqq i \leqq N} \{\hat{s}_i\}.$$

6. Take away the point P_h from the sample: put $N = N - 1$ and repeat steps 4, 5, 6 for $n - 1$ times. Of course, as the number of points in the sample changes, also the number within $\hat{I}(P_i)$ changes.

[1] In the Appendix one can find convergence theorems.

The reason why we take away the most significant datum and update the hierarchy is that we remove the influence the most significant point has on the other ones.

Remarks. The above method has the following desirable properties.

a) At the beginning of the algorithm no sample point has a priority.

b) The different local density of the points does not affect the choice too much.

c) The method is such that the inclusion of a datum into the subsample does not change the hierarchy of the importance levels for the remaining points in an essential way, on condition that the subsample size is not too small a fraction of the original set. So, in general, if it should be computationally costly to update the hierarchy after the inclusion of a datum in the subsample, one might avoid it.

For the reader's convenience, we give here a trivial example for the algorithm. Let the following $\{(P_i, f_i), i = 1, \cdots, 14\}$ be given (see Fig. 1).

$$\{(0.1, 1), (0.2, -0.3), (0.3, 0), (0.4, 0.3), (0.5, 0.), (0.6, -3.),$$
$$(0.7, 0), (0.8, 2.), (0.9, 1.9), (1., 2.), (1.1, 2.5), (1.2, 2.), (1.3, 1.5), (1.4, 2.)\}.$$

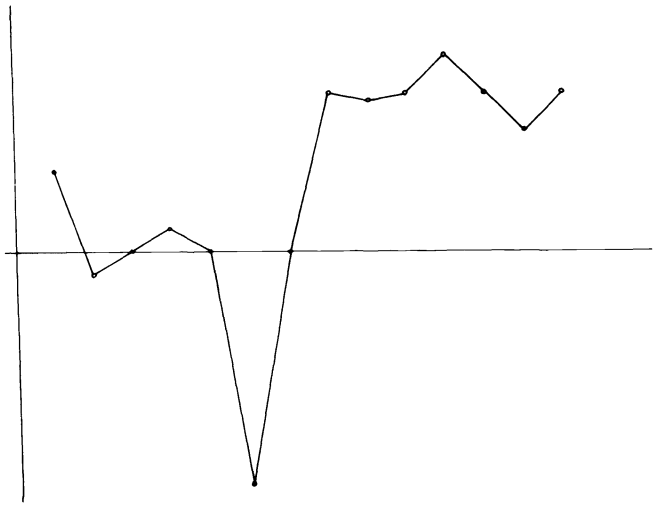


FIG. 1.

1. We set $\hat{I}(P_i)$ as a symmetric interval centered at P_i with fixed radius $R = 0.25$ as a rough choice. The first five data included in the subsample by the algorithm are in the order

$$(P_6, f_6), (P_7, f_7), (P_1, f_1), (P_{11}, f_{11}), (P_{13}, f_{13}).$$

The same subsample with the same order of inclusion is reached also if the hierarchy is not updated each time.

2. We set $\hat{I}(P_i)$ as a symmetric interval centered at P_i with variable radius R_i to include more homogeneous functional behavior:

$$R_1 = R_2 = R_3 = 0.25, \quad R_4 = R_5 = R_6 = R_7 = 0.15,$$
$$R_8 = R_9 = R_{10} = R_{11} = R_{12} = R_{13} = R_{14} = 0.25.$$

The first five data included in the subsample by the algorithm are in the order

$$(P_1, f_1), \quad (P_{11}, f_{11}), \quad (P_6, f_6), \quad (P_7, f_7), \quad (P_2, f_2).$$

If the hierarchy is not updated at each inclusion, we get order

$$(P_1, f_1), \quad (P_{11}, f_{11}), \quad (P_6, f_6), \quad (P_{13}, f_{13}), \quad (P_2, f_2).$$

As previously stated, the purpose of the method is to choose data suitable for a good surface reconstruction by a sample of size n.

For $n = 1$, the approximating function is a constant which is close to the mean value of the functional values; for n greater than one, still, a reasonable subset is taken.

5. Observations. The K_i-neighborhood can be given according to the nature of the problem and in different ways. It may be convenient to give the neighborhood $\hat{I}(P_i)$; when the assignment is made carefully, it is formally equivalent to choosing a suitable K_i. Different examples for the choice of $\hat{I}(P_i)$ can be found in the references, see [2], [4], [5], [9].

The particular flexibility of the proposed method makes it desirable for solving problems of a particular nature, with suitable modifications.

Finally we observe that the choice of the subsample can be made by working separately (maybe with different modalities) on the different subdomains of a partition of D.

6. Numerical examples.

Example I. Let be given $f(x, y) = \exp\{-(x+y-11)^2 - (x-y)^2/10\}$ a long and narrow hill rising from a plain (see [4]) and let be given the values $f(P_i)$ at N points homogeneously distributed in $[0, 11] * [0, 11]$ and including also a few important points (see [4]). We found the n most important data obtained while using the function $\phi_i(P) = 1 - 3(d/R)^2 + 2(d/R)^3$, $d = \text{dist}(P_i, P)$ and as a rough choice fixed as K_i-neighborhoods circular neighborhoods with fixed radius R. Two surfaces are built: the surface $m_1(x, y)$ from the N data and the surface $m_2(x, y)$ from the subsample of n data, in both cases by the same interpolation method of a local type.

A comparison between the goodness of fit of m_1 and m_2 is given by the indices (see [3])

$$(1) \qquad \begin{aligned} Q_1 &= \frac{\sum_1^l (f_h - m_{1h})^2}{\sum_i^l f_h^2}, \qquad q_1 = \frac{\sum_1^l f_h m_{1h}}{\{\sum f_h^2 \cdot \sum m_{1h}^2\}^{1/2}}, \\ Q_2 &= \frac{\sum_1^l (f_h - m_{2h})^2}{\sum_1^l f_h^2}, \qquad q_2 = \frac{\sum_1^l f_h m_{2h}}{\{\sum f_h^2 \cdot \sum m_{2h}^2\}^{1/2}} \end{aligned}$$

where the following points $(x, y)_h$, $h = 1, \cdots, 1$ have been used for the evaluations; they are placed at the grid knots $[1, 1.5, 2., \cdots, 10] * [1, 1.5, 2., \cdots, 10] = G$.

An index of type Q measures a quantitative difference while an index of type q measures a difference in the behavior of the surfaces.

Case 1a). The interpolation method is the modified Shepard's method [5]. The $N = 45$ points are shown in Fig. 2. The $n = 22$ most important points are circled in Fig. 2. R, parameter of ϕ_i, is given the value $R = 3$ and the same value is given to the radius for the local interpolation. The results are

$$Q_1 = .348, \qquad q_1 = .858,$$
$$Q_2 = .364, \qquad q_2 = .881.$$

As can be seen, the two surfaces m_1 and m_2 are very similar; this can be shown better by the values at the same grid points of

$$(2) \qquad Q_{12} = \frac{\sum_1^l (m_{1h} - m_{2h})^2}{\sum m_{1h}^2}, \qquad q_{12} = \frac{\sum_1^l m_{1h} \cdot m_{2h}}{\{\sum m_{1h}^2 \sum m_{2h}^2\}^{1/2}}.$$

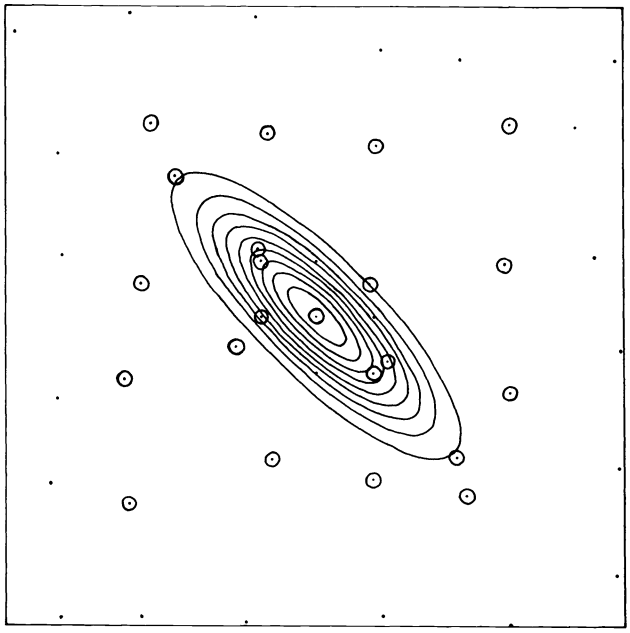


FIG. 2.

These values are

$$Q_{12} = .034, \qquad q_{12} = .988.$$

Incidentally, we observe that because of the simple interpolation method, m_1 and m_2 do not give a satisfactory quantitative fit.

Case 1b). The interpolation method is local least squares interpolation (Method I in [6]). For this method we chose a larger N, $N = 100$ (so as not to have too large a radius for the local neighborhood) and a subset of $n = 50$. $m_1(x, y)$ is built with $R_q = 3$, the radius of the weight for the quadratic function, $m_2(x, y)$ with $R_q = 3.5$. R_w, the radius to weight of the quadratic functions, is given the value $R_w = 1.5$ for both m_1 and m_2; R, the radius of the neighborhoods $\hat{I}(P_i)$ for the subset selection is set to $R = 1.5$.

Using again the indices (1) and (2) for the points of the grid G, we get these values:

$$Q_1 = .032, \qquad q_1 = .984,$$

$$Q_2 = .152, \qquad q_2 = .936,$$

$$Q_{12} = .122, \qquad q_{12} = .951.$$

The results obtained by the method are satisfying, and a better fit by the quadratic method is reached, as expected.

Example II. Two surfaces are given for the behavior of the heart potential on the body for one person at one instant; these surfaces are represented by contour lines with a step = 500 microvolts; the surfaces are rebuilt from a set of function values, making use of a local technique [2].

Because of the nature of the problem the neighborhoods $\hat{I}(P_i)$ were chosen with a quadrangular section; the size of a section is determined from an increasing sequence of sets on the basis of the function variation between the points of the border of a set and those of the border of a contiguous larger set.

For Fig. 3 we used a set of $N = 219$ values, and for Fig. 4 we chose $n = 108$ most important data.

The phenomenon is still reproduced in a satisfactory way.

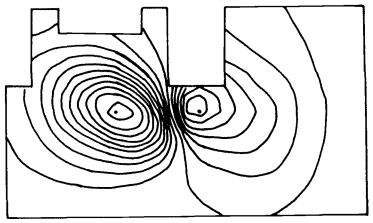


FIG. 3.

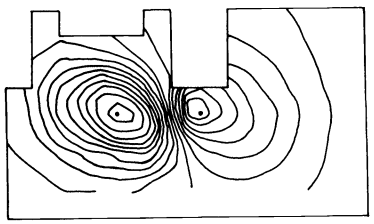


FIG. 4.

Appendix.

THEOREM 1. *Let D be a compact set, $D \in R^q$ and μ a measure defined on D; let $P_0 \in D$ and $I(P_0)$ be its K_0-neighborhood with $\mu(I(P_0)) \neq 0$. Let us take a random sample of points $\{P_i\}_1^M$, $P_i \in D$, according to the measure μ.*

As $M \to \infty$ the sequence of estimators $\{\hat{I}(P_0)\}$ converges to the K_0-neighborhood $I(P_0)$ almost everywhere.

Proof. Let $m = m(M)$ be the number of points of the sample falling into the corresponding $\hat{I}(P_0)$. As $M \to \infty$, $m \to \infty$ also. Indeed, after fixing $\varepsilon > 0$ it is possible to determine $\bar{n} = \bar{n}(\varepsilon, M)$ such that, for $n(M) \geqq \bar{n}$,

$$\left| \frac{1}{\mu(I)} \int_I |f - f_0| \, d\mu - \frac{1}{n} \sum_1^n |f_i - f_0| \right| < \varepsilon \quad \text{a.e.} \tag{1}$$

as $M \to \infty$, with $f_i = f(P_i)$, $P_i \in I$, $i = 1, \cdots, n$. This comes from the law of large numbers. Hence, for n large enough,

$$\frac{1}{n} \sum_1^n |f_i - f_0| < K_0.$$

Let $I_n(P_0)$ be the minimum neighborhood containing the n points; then $I_n(P_0) \subseteq \hat{I}(P_0)$ and as $M \to \infty$, also $n \to \infty$ and hence $m \to \infty$.

The above also proves that the set a of the $\{\hat{I}_{m(M)}(P_0)\}$ has a limit class which does not coincide with the empty set.

Let us call I^* the maximum limit of a and take a sequence $a_1 = \{\hat{I}_{m_i}(P_0)\}$ of members in a and which converges to I^*. After fixing an arbitrary small $\gamma > 0$ the measures of the neighborhoods of a_1 are definitely between $\mu(I^*) - \gamma$, $\mu(I^*) + \gamma$.

Let us take the subsequence of neighborhoods of a_1 containing a nondecreasing number of points h.

As $h \to +\infty$, from the law of large numbers one has

$$\frac{1}{h}\sum |f_i - f_0| \to \frac{1}{\mu(I^*)} \int_{I^*} |f - f_0| \, d\mu \quad \text{a.e.} \tag{2}$$

Since, by definition, I is the maximum neighborhood for which

$$\frac{1}{\mu(I)} \int_I |f - f_0| \, d\mu < K_0,$$

it follows from (1) and (2) that $I^* \equiv I$ except for a set of measure zero.

Remark. With the same hypothesis as Theorem 1, as $M \to \infty$ from the law of large numbers, the sequence of estimators $\hat{\delta}_i \to \delta$ a.e.

REFERENCES

[1] R. BARR, M. S. SPACH AND G. SCOTT HERMAN-GIDDENS, *Selection of the number and positions of measuring locations for electrocardiography*, IEEE, Trans. Bio. Med. Engng, BME-18 (1971), pp. 125-138.

[2] M. BOZZINI, F. DE TISI AND L. LENARDUZZI, *An approximation method of the local type. Application to a problem of heart potential mapping*, Computing, 32 (1984), pp. 69-80.

[3] ———, *Indices to evaluate the agreement between functions in problems of numerical approximation*, manuscript.

[4] R. FRANKE, *Locally determined smooth interpolation at irregularly spaced points in several variables*, J. Inst. Maths. Applics., 19 (1977), pp. 471-482.

[5] ———, *A critical comparison of some methods for interpolation of scattered data*, Naval Postgrad. Sch. Tech. Rep., NPS53-79-003 (1979).

[6] R. FRANKE AND G. NIELSON, *Smooth interpolation of large sets of scattered data*, Internat. J. Numer. Methods Engrg., 15, 1980, pp. 1691-1704.

[7] L. S. GANDIN, *The planning of meteorological station networks*, WMO-No 265.TP.149 (1970).

[8] A. M. KSHIRSAGAR, *Multivariate Analysis*, Dekker, New York, 1972.

[9] C. J. STONE, *Consistent nonparametric regression*, Ann. Statist., 5 (1977), pp. 595-645.

SIAM J. SCI. STAT. COMPUT.
Vol. 7, No. 1, January 1986

008

CENSORED DISCRETE LINEAR l_1 APPROXIMATION*

R. S. WOMERSLEY†

Abstract. The censored linear l_1 approximation problem is to minimize the nonconvex piecewise linear function $F(x)=\sum_{i=1}^{m}|y_i-\max(z_i, x^Ta_i)|$. The problem arises in regression models where the range of the dependent variable is restricted. Unlike the maximum likelihood and least squares estimators the censored l_1 estimator provides a consistent estimator without an assumption that the errors are normally distributed.

This paper presents a compact characterization of the generalized gradient of F, and necessary and sufficient conditions for a (strict) local minimizer of F. A reduced gradient algorithm for linear programming and l_1 approximation is extended to provide a stable finite direct descent method for calculating a local minimizer of F. This provides an efficient method of calculating the censored l_1 estimator.

Key words. censored l_1 approximation, censored LAD estimation, generalized gradient, reduced gradient algorithm

AMS(MOS) 1980 subject classification. Primary 90

1. Introduction.

The censored discrete linear l_1 approximation problem is to minimize over $x\in\mathbb{R}^n$ the nonconvex piecewise linear function $F:\mathbb{R}^n\to\mathbb{R}$ defined by

$$F(x)=\sum_{i=1}^{m}|y_i-\max(z_i, x^Ta_i)|, \tag{1.1}$$

where $y_i\in\mathbb{R}$, $z_i\in\mathbb{R}$ and $a_i\in\mathbb{R}^n$ for $i=1,\cdots,m$.

The interest in this problem arises from the censored and truncated least absolute deviation (LAD) estimators proposed by Powell [8] and [10] for regression coefficients in two models with limited dependent variables—that is, regression models where the range of the dependent variable is restricted to some subset of the real line. Consider a linear process, where the dependent variable is restricted to be nonnegative, namely

$$y_i=\max(0, \tilde{x}^Ta_i+\varepsilon_i). \tag{1.2}$$

Here $\tilde{x}$ is the underlying vector of parameters to be estimated, ε_i are unobservable errors and $y_i\in\mathbb{R}$ and $a_i\in\mathbb{R}^n$ are observable data. The consistency of the maximum likelihood and least squares estimators depends critically upon the assumption that the errors are normally distributed. However the censored LAD estimator, that is the global minimizer of (1.1) with $z_i=0\ \forall i=1,\cdots,m$, provides a consistent estimator of $\tilde{x}$ which does not depend upon the functional form of the distribution of the errors.

An important feature of (1.1) is the generality obtained by taking z_i arbitrary. Thus the results of this paper apply to censored l_1 estimation ($z_i=0\ \forall i$), truncated l_1 estimation ($z_i=y_i/2\ \forall i$), general lower and upper bounds on the observed dependent variable, as well as to standard l_1 estimation ($z_i=-\infty\ \forall i$). The results could also be extended to regression quantile estimation [9].

One aim of this paper is to provide an efficient algorithm for computing the censored LAD estimator, and thus remove a major obstacle to its use. For instance Paarsch [7] published a simulation study of the censored l_1 estimator which uses a grid-search method to minimize (1.1)! Although Powell's proof [10] of the consistency and asymptotic normality of the censored LAD estimator assumes the unique global

* Received by the editors February 27, 1984 and in revised form October 15, 1984.

† School of Mathematics, University of New South Wales, P.O. Box 1, Kensington NSW 2033, Australia. This work was done while the author was at the Mathematical Sciences Research Centre, Australian National University.

minimizer of F is calculated, in practice only a local minimizer can be calculated as F is not convex. However on some randomly generated test problems this seemed to be a major difficulty only when m is of the same order of magnitude as n. When $m \gg n$, as in most practical applications, the algorithm usually found the global minimizer of F (see § 5 for further comments).

The following example [11] of temperature accelerated life tests on electrical insulation in 40 motorettes illustrates the type of problem on which the algorithm can be used. Ten motorettes were tested at each of four temperatures. Testing was terminated at different times at each temperature giving the data in Table 1. The model fitted in [11] is

$$\log_{10} H = \tilde{x}_1 + 1000\tilde{x}_2/(T+273.2) + \varepsilon,$$

where H is the failure time and T is the temperature. At each temperature there is an upper bound H_t (the time at which testing was stopped) on the observed failure times, so the logarithms of the observed failure times are given by

$$\min(\log_{10} H_t, x_1 + 1000\tilde{x}_2/(T+273.2) + \varepsilon).$$

A problem with upper bounds on the observed variable can be converted into one with lower bounds by changing the sign of the data y_i, z_i and a_i, as

$$|y_i - \max(z_i, x^T a_i)| = |-y_i - \min(-z_i, -x^T a_i)|.$$

TABLE 1
Data for motorette example.

	Test temperature T °C			
	150	170	190	200
Failure times H in hours		1764	408	408
		2722	408	408
		3444	1344	504
		3542	1344	504
		3780	1440	504
		4860		
		5196		
Termination time H_t	8064	5448	1680	528
	10 units	3 units	5 units	5 units

The next section considers the differential properties of F and establishes a concise characterization of the generalized gradient $\partial F(x)$ of $F(x)$. Section 3 strengthens the usual necessary conditions $0 \in \partial F(x)$ of nonsmooth optimization (Clarke [5]) to provide necessary and sufficient conditions for a (strict) local minimizer of F. The interpolation result for l_1 approximation (see for example Watson [12, p. 119]) extends to censored l_1 approximation. That is a global minimizer of F is characterized by n (when the vectors a_i, $i = 1, \cdots, m$ have rank n) linear equations $r_i(x) \equiv \max(y_i, z_i) - a_i^T x = 0$. This reduces the search for a minimizer of F to a finite number of points.

A finite direct descent algorithm is developed in § 4. This algorithm is a generalization of a reduced gradient algorithm for l_1 approximation, which in turn is simple an extension of a reduced gradient algorithm for linear programming (see Osborne [6]). The algorithm which is numerically stable, can be implemented in a convenient tableau form for relatively small dense problems. An efficient algorithm, where for

large m the number of operations per iteration is dominated by $nm \log m$, can be developed. For fixed n the number of iterations needed to find a solution appears to grow slowly with m.

2. Differential properties. The function F defined by (1.1) can be written as the sum of a nonconvex function and a convex function, namely

$$F(x) = \sum_{i:y_i > z_i} |y_i - \max(z_i, x^T a_i)| + \sum_{i:y_i \leqq z_i} -y_i + \max(z_i, x^T a_i). \tag{2.1}$$

The character of the component functions is illustrated by the following one-dimensional examples. The situation where $y_i > z_i$ is typified by the function $f_1(x) = |1 - \max(0, x)|$, where $y_1 = 1$, $z_1 = 0$ and $a_1 = 1$, which is sketched in Fig. 1. The situation where $y_i \leqq z_i$ is typified by the function $f_2(x) = \max(0, x)$ where $y_1 = 0$, $z_1 = 0$ and $a_1 = 1$, which is sketched in Fig. 2.

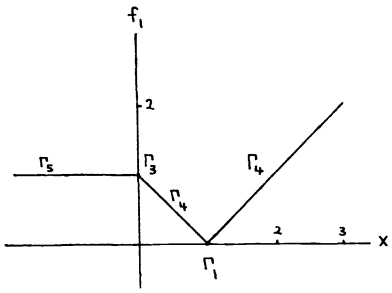


FIG. 1. *A function with* $y_i > z_i$.

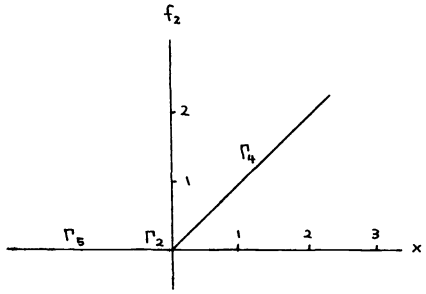


FIG. 2. *A function with* $y_i \leqq z_i$.

It should be noted that for censored estimation problems with a lower bound z_i on the dependent variable

$$y_i = \max(z_i, \tilde{x}^T a_i + \varepsilon_i) \geqq z_i.$$

Thus the inequalities $y_i \leqq z_i$ could be replaced by $y_i = z_i$ in the definition of the sets $\Gamma_i(x)$ below. However this does not produce any further simplification, so is not used in the rest of the paper. Define the functions $r_i(x)$ by

$$r_i(x) = \max(y_i, z_i) - x^T a_i \quad \text{for } i = 1, \cdots, m, \tag{2.2}$$

and the index sets $\Gamma_j(x)$, $j = 1, \cdots, 5$ by

$$\begin{aligned}
\Gamma_1(x) &= \{i \in 1, \cdots, m\colon y_i > z_i \text{ and } x^T a_i = y_i\},\\
\Gamma_2(x) &= \{i \in 1, \cdots, m\colon y_i \leqq z_i \text{ and } x^T a_i = z_i\},\\
\Gamma_3(x) &= \{i \in 1, \cdots, m\colon y_i > z_i \text{ and } x^T a_i = z_i\},\\
\Gamma_4(x) &= \{i \in 1, \cdots, m\colon x^T a_i > z_i \text{ and } x^T a_i \neq y_i\},\\
\Gamma_5(x) &= \{i \in 1, \cdots, m\colon x^T a_i < z_i\}.
\end{aligned} \tag{2.3}$$

As an example the nonempty sets $\Gamma_i(x)$ are marked on Figs. 1 and 2. Also define the index set $\mathcal{A}(x)$ by

$$\mathcal{A}(x) = \Gamma_i(x) \cup \Gamma_2(x) = \{i \in 1, \cdots, m\colon r_i(x) = 0\}. \tag{2.4}$$

The sets $\Gamma_j(x)$, $j=1,\cdots,5$ form a disjoint partition of $\{1,\cdots,m\}$, that is

$$\bigcup_{j=1}^{5}\Gamma_j(x)=\{1,\cdots,m\} \quad \forall x\in\mathbb{R}^n,$$

$$\Gamma_i(x)\cap\Gamma_j(x)=\phi \quad \forall i\neq j, \quad \forall x\in\mathbb{R}^n.$$

A definition of the generalized gradient $\partial F(x)$ (Clarke [5]) of a piecewise smooth function F at the point x is

$$\partial F(x)=\operatorname{conv}\{u\in\mathbb{R}^n:\ \exists \text{ a sequence } \{x^{(k)}\} \text{ such that } x^{(k)}\to x,\ \nabla F(x^{(k)}) \text{ exists } \forall k, \text{ and } \nabla F(x^{(k)})\to u \text{ as } k\to\infty\}, \tag{2.5}$$

where conv G denotes the convex hull of G. For functions $F:\mathbb{R}^n\to\mathbb{R}$, $\partial F(x)$ is a nonempty compact convex set in $\mathbb{R}^n$.

For $i\in\Gamma_4(x)$ the component functions $|y_i-\max(z_i, x^Ta_i)|$ are smooth (continuously differentiable) in a neighbourhood of x, with gradient $-\theta_i a_i$ where $\theta_i=\operatorname{sign}(r_i(x))$. For $i\in\Gamma_5(x)$ the component functions $|y_i-\max(z_i, x^Ta_i)|=|y_i-z_i|$ are also smooth in a neighbourhood of x, but with zero gradient. The gradient $g\equiv g(x)$ of the smooth component functions is thus

$$g(x)=-\sum_{i\in\Gamma_4(x)}\theta_i a_i. \tag{2.6}$$

For $i\in\mathscr{A}(x)$ the component functions $|y_i-\max(z_i, x^Ta_i)|$ are nonsmooth, but convex is a neighbourhood of x, with generalized gradients (subdifferentials) given by, for $i\in\Gamma_1(x)$

$$\partial|y_i-\max(z_i, x^Ta_i)|=\operatorname{conv}\{-a_i, a_i\}=\{u\in\mathbb{R}^n:\ u=-\lambda_i a_i, -1\leqq\lambda_i\leqq 1\},$$

and for $i\in\Gamma_2(x)$

$$\partial(-y_i+\max(z_i, x^Ta_i))=\operatorname{conv}\{0, a_i\}=\{u\in\mathbb{R}^n:\ u=-\lambda_i a_i, -1\leqq\lambda_i\leqq 0\}.$$

For $i\in\Gamma_3(x)$ the component functions are nonsmooth and nonconvex at x with generalized gradient

$$\partial|y_i-\max(z_i, x^Ta_i)|=\operatorname{conv}\{-a_i, 0\}=\{u\in\mathbb{R}^n:\ u=-\lambda_i a_i, 0\leqq\lambda_i\leqq 1\} \quad \text{for } i\in\Gamma_3(x).$$

As generalized gradients satisfy $\partial(F_1(x)+F_2(x))\subseteq\partial F_1(x)+\partial F_2(x)$, one has $\partial F(x)\subseteq G(x)$ where $G(x)$ is the nonempty compact convex polytope defined by

$$G(x)=\Big\{v\in\mathbb{R}^n:\ v=g(x)-\sum_{i\in\mathscr{A}(x)\cup\Gamma_3(x)}\lambda_i a_i \quad \text{where} \quad |\lambda_i|\leqq 1,\ i\in\Gamma_1(x),\ -1\leqq\lambda_i\leqq 0,\ i\in\Gamma_2(x),\ 0\leqq\lambda_i\leqq 1,\ i\in\Gamma_3(x)\Big\}. \tag{2.7}$$

F may not be convex, so the inclusion $\partial F(x)\subseteq G(x)$ can be strict as the following example illustrates. Let $n=2$, $m=3$, $z^T=(0,0,0)$, $y^T=(1,2,2)$, $a_1^T=(1,-1)$, $a_2^T=(1,1)$ and $a_3^T=(-1,3)$. The contours of F are sketched in Fig. 3, whilst the sets $\partial F(x)$ and $G(x)$ at the point $\bar{x}^T=(3/2,1/2)$ are sketched in Fig. 4. At the point $\bar{x}^T=(3/2,1/2)\Gamma_1=\{1,2\}$, $\Gamma_3=\{3\}$ and $\Gamma_2=\Gamma_4=\Gamma_5=\phi$. Thus the possible extreme values of λ in (2.7) are

$$\begin{bmatrix}1\\1\\1\end{bmatrix},\ \begin{bmatrix}1\\1\\0\end{bmatrix},\ \begin{bmatrix}1\\-1\\1\end{bmatrix},\ \begin{bmatrix}1\\-1\\0\end{bmatrix},\ \begin{bmatrix}-1\\1\\1\end{bmatrix},\ \begin{bmatrix}-1\\1\\0\end{bmatrix},\ \begin{bmatrix}-1\\-1\\1\end{bmatrix},\ \begin{bmatrix}-1\\-1\\0\end{bmatrix},$$

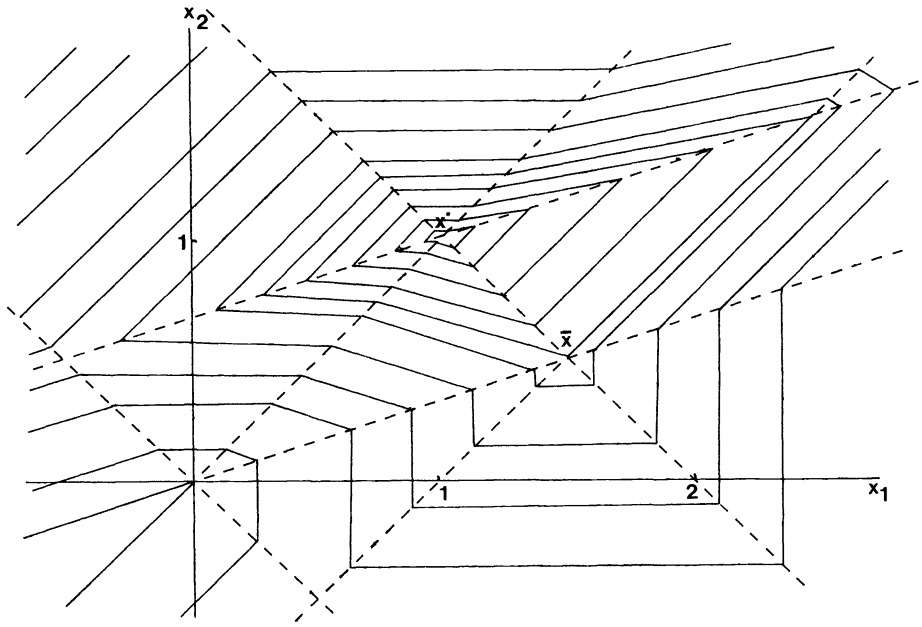


FIG. 3. *Contours of Example* 1.

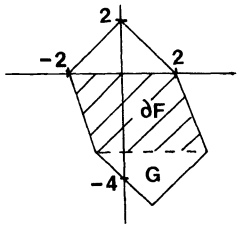


FIG. 4. $G(\bar{x})$ *and* $\partial F(\bar{x})$.

giving

$$G(\bar{x}) = \operatorname{conv}\left\{\begin{bmatrix}-1\\3\end{bmatrix}\begin{bmatrix}-2\\0\end{bmatrix}\begin{bmatrix}1\\-1\end{bmatrix}\begin{bmatrix}0\\2\end{bmatrix}\begin{bmatrix}1\\-5\end{bmatrix}\begin{bmatrix}0\\-2\end{bmatrix}\begin{bmatrix}3\\-3\end{bmatrix}\begin{bmatrix}2\\0\end{bmatrix}\right\}.$$

However from (2.5) and Fig. 3 one can see that

$$\partial F(\bar{x}) = \operatorname{conv}\left\{\begin{bmatrix}-1\\-3\end{bmatrix}\begin{bmatrix}-2\\0\end{bmatrix}\begin{bmatrix}1\\-1\end{bmatrix}\begin{bmatrix}0\\-2\end{bmatrix}\begin{bmatrix}-3\\3\end{bmatrix}\begin{bmatrix}2\\0\end{bmatrix}\right\},$$

which is a strict subset of $G(\bar{x})$.

The following result gives conditions which ensure the equality of the sets $\partial F(x)$ and $G(x)$.

LEMMA 1. *Let* $\hat{\lambda}_i = \pm 1$ for $i \in \Gamma_1(x)$, $\hat{\lambda}_i = -1$ *or* 0 *for* $i \in \Gamma_2(x)$ *and* $\hat{\lambda}_i = 0$ *or* 1 *for* $i \in \Gamma_3(x)$. *Define the set* $\mathcal{F}(\hat{\lambda})$ *by*

$$\begin{aligned}\mathcal{F}(\hat{\lambda}) = \{\omega \in \mathbb{R}^n\colon\ & z_i \leqq \omega^T a_i \leqq y_i \text{ for } i \in \Gamma_1(x) \text{ with } \hat{\lambda}_i = 1\\ & y_i \leqq \omega^T a_i \text{ for } i \in \Gamma_1(x) \text{ with } \hat{\lambda}_i = -1\\ & z_i \leqq \omega^T a_i \text{ for } i \in \Gamma_2(x) \text{ with } \hat{\lambda}_i = -1 \text{ and } i \in \Gamma_3(x) \text{ with } \hat{\lambda}_i = 1\\ & \omega^T a_i \leqq z_i \text{ for } i \in \Gamma_2(x) \cup \Gamma_3(x) \text{ with } \hat{\lambda}_i = 0\}.\end{aligned}$$

Then $\partial F(x) = G(x)$ if and only if $\mathcal{F}(\hat{\lambda})$ has nonempty interior for every $\hat{\lambda}$ which corresponds to an extreme point of $G(x)$.

Proof. (a) Assume $\mathcal{F}(\hat{\lambda})$ has nonempty interior for every $\hat{\lambda}$ corresponding to an extreme point of $G(x)$. As $\partial F(x) \subseteq G(x)$ one only has to show $G(x) \subseteq \partial F(x)$. Note that $\mathcal{F}(\hat{\lambda})$ is the subset of $\mathbb{R}^n$ in which the linear function specified by $\hat{\lambda}$ is active, so the gradient of this linear function is

$$\hat{v} = g - \sum_{i \in \mathcal{A} \cup \Gamma_3} \hat{\lambda}_i a_i.$$

Let $\bar{x} \in \operatorname{int} \mathcal{F}(\hat{\lambda})$, and set $s = \bar{x} - x$. Then $F(x + \alpha s) = F(x) + \alpha \hat{v} \forall \alpha \in [0, 1]$; moreover $\nabla F(x + \alpha s)$ exists and $\nabla F(x + \alpha s) = \hat{v}$ for all α in $(0, 1]$. Thus from (2.5) $\hat{v} \in \partial F(x)$. This holds for all extreme points $\hat{v}$ of $G(x)$ giving $G(x) \subseteq \partial F(x)$.

(b) Assume $\partial F(x) = G(x)$. Let $\hat{\lambda}$ correspond to an extreme point $\hat{v}$ of $G(x)$, and hence $\partial F(x)$. Then from (2.5) there exists a sequence $\{x^{(k)}\}$ such that $x^{(k)} \to x$, $\nabla F(x^{(k)})$ exists for all k and $\nabla F(x^{(k)}) \to \hat{v}$ as $k \to \infty$. As F is piecewise linear, this implies $\hat{v} = \nabla F(x^{(k)})$ for all k sufficiently large ($x^{(k)} \neq x$). As $\nabla F(x^{(k)})$ exists, one has $x^{(k)} \in \operatorname{int} \mathcal{F}(\hat{\lambda})$ for these k. □

If $\operatorname{rank} \{a_i : i \in \mathcal{A}(x) \cup \Gamma_3(x)\} = n$, then there are vectors $\hat{\lambda}$ for which $\mathcal{F}(\hat{\lambda})$ has an empty interior. The sets $\partial F(x)$ and $G(x)$ are equal if and only if these $\hat{\lambda}$ correspond to points $\hat{v}$ which are not extreme points of $G(x)$. If $\Gamma_3(x)$ is empty, then F is convex in a neighbourhood of x, so $\partial F(x) = G(x)$. Thus difficulties only arise at points where $\Gamma_3(x)$ is nonempty.

A key tool in the development of optimality conditions in nonsmooth optimization is the one-sided directional derivative $F'(x; s)$ defined by

$$F'(x; s) = \lim_{\alpha \to 0^+} \frac{F(x + \alpha s) - F(x)}{\alpha},$$

which exists for all x, $s \in \mathbb{R}^n$ as F is a continuous piecewise linear function. For the function (1.1) one has

$$F'(x; s) = s^T g + \sum_{i \in \Gamma_1(x)} |s^T a_i| + \sum_{i \in \Gamma_2(x)} \max(0, s^T a_i) - \sum_{i \in \Gamma_3(x)} \max(0, s^T a_i). \tag{2.8}$$

As F may be nonconvex at points x where $\Gamma_3(x)$ is nonempty, one only has

$$F'(x; s) \leqq \max_{u \in \partial F(x)} u^T s. \tag{2.9}$$

If $\Gamma_3(x)$ is empty, then (2.9) holds with equality.

3. Characterization of minimizers. The inclusion $\partial F(x) \subseteq G(x)$ means that the well-known necessary conditions $0 \in \partial F(x^*)$ [5] for x^* to be a local minimizer of F carry through to $0 \in G(x^*)$. However sufficient conditions are only immediately available when $\Gamma_3(x^*)$ is empty, as then F is convex in a neighbourhood of x^*. In that case $\partial F(x^*) = G(x^*)$ and $0 \in G(x^*)$ is both necessary and sufficient, whilst $0 \in \operatorname{int} G(x^*)$ ensures that x^* is a strict local minimizer of F. Two one-dimensional examples; F_1 with $y = (1, 1/2)^T$, $z = (0, 0)^T$, $a_1 = 1$, $a_2 = -1/2$ and F_2 with $y = (1, 1, 2)^T$, $z = (0, 0, 0)^T$, $a_1 = 2$, $a_2 = 1$, $a_3 = 1$, are sketched in Figs. 5 and 6. F_1 has strict local minima at $x = -1$ with $F_1 = 1$ and at $x = 1$ with $F_1 = 1/2$, whilst any $x \in [1/2, 1]$ is a local minimizer of F_2 with $F_2 = 1$.

When $\Gamma_3(x)$ is nonempty, the situation is more complicated. However one can obtain an interpolation result similar to the discrete l_1 case where a solution is

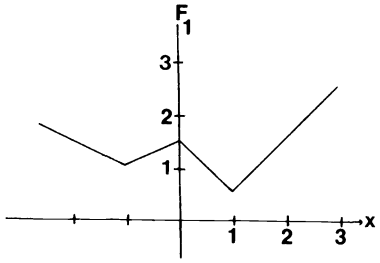


FIG. 5. *Distinct local minimizers.*

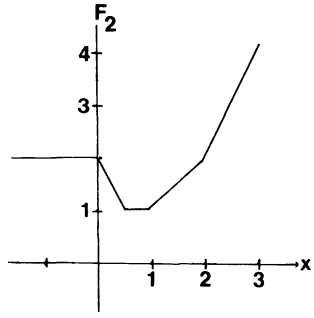


FIG. 6. *Multiple minimizers.*

characterized in terms of the residual functions $r_i(x)$ which are zero. Define

$$A=[a_i\colon i\in\mathcal{A}(x)].$$

LEMMA 2. *Let x be a point with $0\in G(x)$ and rank$(A)<n$. Then $F'(x;s)\leqq 0$ for all directions s satisfying $s^TA=0$. Moreover if there exists an index $k\in\Gamma_3(x)$ with $a_k\notin\mathcal{R}(A)$, the range of space of A, then there exists a descent direction s with $s^TA=0$.*

Proof. As $0\in G(x)$ there exists a vector λ satisfying $|\lambda_i|\leqq 1$, $i\in\Gamma_1(x)$, $-1\leqq\lambda_i\leqq 0$, $i\in\Gamma_2(x)$, $0\leqq\lambda_i\leqq 1$, $i\in\Gamma_3(x)$ and

$$g=\sum_{i\in\mathcal{A}(x)\cup\Gamma_3(x)}\lambda_i a_i.$$

Let s satisfy $s^TA=0$. If $\Gamma_3(x)$ is nonempty, then

$$s^Tg=\sum_{i\in\Gamma_3(x)}\lambda_i s^Ta_i.$$

From (2.8)

$$\begin{aligned}F'(x;s)&=s^Tg-\sum_{i\in\Gamma_3(x)}\max(0,s^Ta_i)\\&=\sum_{i\in\Gamma_3(x)}[\lambda_i s^Ta_i-\max(0,s^Ta_i)]\\&=\sum_{\substack{i\in\Gamma_3(x)\\ s^Ta_i\leqq 0}}\lambda_i s^Ta_i+\sum_{\substack{i\in\Gamma_3(x)\\ s^Ta_i>0}}(\lambda_i-1)s^Ta_i\\&\leqq 0,\end{aligned}$$

as $0\leqq\lambda_i\leqq 1$ for all $i\in\Gamma_3(x)$. If $\Gamma_3(x)$ is empty, then $s^TA=0$ implies $s^Tg=0$ and hence $F'(x;s)=0$. Now let $k\in\Gamma_3(x)$ be such that $a_k\notin\mathcal{R}(A)$. If $0<\lambda_k\leqq 1$ and s is chosen so that $s^Ta_k<0$, then $F'(x;s)<0$. Alternatively if $0\leqq\lambda_k<1$ and s is chosen so that $s^Ta_k>0$, then $F'(x;s)<0$. □

Note that if $k\in\Gamma_3(x)$, $0<\lambda_k<1$, and $a_k\notin\mathcal{R}(A)$ then any direction s such that $s^TA=0$ is a descent direction. It is only if λ_k is at one of its bounds that an additional restriction needs to be placed on s to obtain a descent direction.

The significance of the above result is that in searching for a minimizer one need only consider points where $a_i\in\mathcal{R}(A)$ for all $i\in\Gamma_3(x)$. Typically methods also involve a line search, and the following result shows that only points where a new residual $r_i(x)$ becomes zero need be considered in minimizing F along a line. Define

$$I(x;s)=\{i\in 1,\cdots,m\colon r_i(x)(s^Ta_i)>0\}. \tag{3.1}$$

LEMMA 3. *Let s be a direction such that $F'(x; s) \leqq 0$. If $I(x; s)$ is empty, then $F(x+\alpha s) = F(x)\ \forall \alpha \geqq 0$. Otherwise any local minimum of $F(x+\alpha s)$ over $\alpha \geqq 0$ is attained at a point $\alpha_i > 0$ satisfying $r_i(x+\alpha_i s) = 0$ for some $i \in I(x; s)$.*

Proof. Define

$$F'(x+\alpha^- s; s) \equiv \lim_{\zeta \to \alpha^-} F'(x+\zeta s; s) = -F'(x+\alpha s; -s).$$

The only points α where $F'(x+\alpha s; s)$ changes are those that satisfy either $r_i(x+\alpha s) = 0$ or $(x+\alpha s)^T a_i = z_i$ for some $i \in \{1, \cdots, m\}$. Let α be a point at which only the second of these conditions is satisfied, then from (2.8)

$$F'(x+\alpha s; s) = F'(x+\alpha^- s; s) - \sum_{\substack{i \in \Gamma_3(x+\alpha s) \\ s^T a_i > 0}} s^T a_i + \sum_{\substack{i \in \Gamma_3(x+\alpha s) \\ s^T a_i < 0}} s^T a_i$$

$$< F'(x+\alpha^- s; s).$$

Thus points α satisfying just $(x+\alpha s)^T a_i = z_i$ need not be considered in looking for minimizers of $F(x+\alpha s)$. It is easily verified that there exists an $\alpha > 0$ with $r_i(x+\alpha s) = 0$ if and only if $i \in I(x; s)$. If $I(x; s) = \phi$ then $F'(x+\alpha s; s)$ is a nonincreasing function of $\alpha \geqq 0$. As F is continuous and bounded below by zero the only possibility is that $F'(x+\alpha s; s) = 0$ for all $\alpha \geqq 0$. □

Solving $r_i(x+\alpha_i s) = 0$ gives

$$\alpha_i = \frac{\max(y_i, z_i) - x^T a_i}{s^T a_i}, \qquad i \in I(x; s),$$

as the points to be considered in a line search. These may include separate local minima as well as points which do not correspond to minima of $F(x+\alpha s)$ (see Figs. 5 and 6). The global minimum may also be attained at other points.

Lemmas 2 and 3 enable one to obtain the following result, which states that when the vectors a_i, $i = 1, \cdots, m$ have rank n (which is usually true as $m \gg n$ in practice) the global minimum of F is attained at a point characterized by n linear equations $r_i(x) = 0$. This reduces the search for a global minimizer of F to a finite number of points, which may also include nonglobal local minimizers of F (for example $x = -1$ in Fig. 5.). However certain nonglobal local minima are excluded (for example any $x < 0$ in Fig. 6). This result corresponds to the interpolation result for discrete l_1 approximation (see for example Watson [12, p. 119]), remembering that $r_i(x) = 0$ if and only if $i \in \mathscr{A}(x)$.

THEOREM 1. *If the vectors a_i, $i = 1, \cdots, m$ have rank m_r, then there exists a global minimizer x^* of $F(x)$ with rank $(A^*) = m_r$, and hence $\mathscr{A}(x^*) \equiv \Gamma_1(x^*) \cup \Gamma_2(x^*)$ contains at least m_r indices.*

Proof. Suppose $\bar{x}$ is a global minimizer of F with rank $(\bar{A}) < m_r$. Then $0 \in G(\bar{x})$. Let $s \neq 0$ be a direction satisfying $s^T \bar{A} = 0$. As rank $(\bar{A}) < m_r$ there exists an index k with $k \notin \mathscr{A}(\bar{x})$ and $s^T a_k \neq 0$. The sign of s can then be chosen so that $r_k(\bar{x})(s^T a_k) > 0$, and hence $k \in I(\bar{x}; s)$. Then $F'(\bar{x}; s) \leqq 0$ by Lemma 2, and by Lemma 3 the minimum of $F(\bar{x}+\alpha s)$ over $\alpha \geqq 0$ is attained at a point $\alpha_i > 0$ where $r_i(\bar{x}+\alpha_i s) = 0$ for some $i \in I(\bar{x}; s)$. This process can be repeated until rank $(A) = m_r$. □

The following theorem strengthens the condition $0 \in G(x)$ to provide necessary and sufficient conditions for local minimizers and strict local minimizers of F. The proof also illustrates how descent directions may be calculated. Let $\mathscr{A}^*$ denote $\mathscr{A}(x^*)$, $g^* = g(x^*)$ and so on.

THEOREM 2. *The point x^* is a local minimizer of F if and only if there exist multipliers λ and $\mu^{(j)}$, $j \in \Gamma_3^*$ satisfying*

(1)

$$g^* = \sum_{i \in \mathscr{A}^*} \lambda_i a_i, \qquad a_j = \sum_{i \in \mathscr{A}^*} \mu_i^{(j)} a_i \quad \forall j \in \Gamma_3^*; \tag{3.2}$$

(2)

$$\begin{aligned} -\lambda_i + \sum_{j \in \Gamma_3^*} \max(0, \mu_i^{(j)}) &\leqq 1 \quad \forall i \in \mathscr{A}^*, \\ \lambda_i + \sum_{j \in \Gamma_3^*} \max(0, -\mu_i^{(j)}) &\leqq 1 \quad \forall i \in \Gamma_1^*, \\ \lambda_i + \sum_{j \in \Gamma_3^*} \max(0, -\mu_i^{(j)}) &\leqq 0 \quad \forall i \in \Gamma_2^*. \end{aligned} \tag{3.3}$$

Moreover the point x^* is a strict local minimizer of F if and only if the following condition also holds.

(3) *The set of vectors a_i, $i \in \mathscr{A}^*$ such that all the inequalities* (3.3) *are strict has rank n.*

Proof. (a) Suppose there exist multipliers λ and $\mu^{(j)}$, $j \in \Gamma_3(x^*)$ satisfying conditions (1) and (2). Now x^* is a local minimizer of F if and only if $F'(x^*; s) \geqq 0 \forall s \in \mathbb{R}^n$. Moreover x^* is a strict local minimizer of F if and only if $F'(x^*, s) > 0 \ \forall s \neq 0$, $s \in \mathbb{R}^n$. Now from (2.8).

$$\begin{aligned} F'(x^*; s) &= \sum_{i \in \Gamma_1^*} [\lambda_i s^T a_i + |s^T a_i|] + \sum_{i \in \Gamma_2^*} [\lambda_i s^T a_i + \max(0, s^T a_i)] \\ &\quad - \sum_{j \in \Gamma_3^*} \max\left(0, \sum_{i \in \mathscr{A}^*} \mu_i^{(j)} s^T a_i\right) \\ &\geqq \sum_{i \in \mathscr{A}^*} \left[\lambda_i \operatorname{sign}(s^T a_i) + \eta_i - \sum_{j \in \Gamma_3^*} \max(0, \mu_i^{(j)} \operatorname{sign}(s^T a_i))\right] |s^T a_i| \\ &= \sum_{i \in \mathscr{A}^*} \zeta_i |s^T a_i| \geqq 0, \end{aligned} \tag{3.4}$$

where $\eta_i = 1$ for $i \in \Gamma_1^*$ and $\eta_i = \max(0, \operatorname{sign}(s^T a_i))$ for $i \in \Gamma_2^*$. The last inequality follows as $\zeta_i \geqq 0 \ \forall i \in \mathscr{A}^*$ from condition (2). Moreover if condition (3) holds, then $\zeta_i > 0$ for a set of indices i such that the corresponding vectors a_i have rank n, and one obtains $F'(x^*; s) > 0 \ \forall s \neq 0$.

(b) Suppose x^* is a local minimizer of F. Then $0 \in G^*$, so there exist multipliers $\tilde{\lambda}$ such that

$$g^* = \sum_{i \in \mathscr{A}^* \cup \Gamma_3^*} \tilde{\lambda}_i a_i.$$

As a descent direction cannot exist at x^*, Lemma 2 implies there exist $\mu^{(j)}$ satisfying

$$a_j = \sum_{i \in \mathscr{A}^*} \mu_i^{(j)} a_i \quad \forall j \in \Gamma_3^*.$$

Hence there exist multipliers λ and $\mu^{(j)}$, $j \in \Gamma_3^*$ satisfying (3.2).

Now suppose there exists an index $k \in \mathscr{A}^*$ such that

$$-\sigma\lambda_k + \sum_{j \in \Gamma_3^*} \max(0, \sigma\mu_k^{(j)}) > \begin{cases} 1, & k \in \Gamma_1^*, \\ \max(0, \sigma), & k \in \Gamma_2^*, \end{cases}$$

where $\sigma = +1$ or -1. Let Λ be a subset of $\mathscr{A}^*$ such that $k \in \Lambda$ and the vectors a_i, $i \in \Lambda$ form a basis for $\mathscr{R}(A)$. For $i \in \mathscr{A}^*$ such that $i \notin \Lambda$ the multipliers λ_i and $\mu_i^{(j)}$, $j \in \Gamma_3^*$

can be chosen arbitrarily whilst still satisfying (3.2). Let s be a direction satisfying $s^T a_k = \sigma$ and $s^T a_i = 0$ for $i \neq k$, $i \in \Lambda$. Also for all $j \in \Gamma_3^*$ and $i \in \mathscr{A}^*/\Lambda$ let $\mu_i^{(j)} = 0$ and $\lambda_i = -\text{sign}\,(s^T a_i)$. Then from (3.4)

$$F'(x^*; s) = \sigma\lambda_k + \begin{cases} 1, & k \in \Gamma_1^* \\ \max(0, \sigma), & k \in \Gamma_2^* \end{cases} - \sum_{j \in \Gamma_3^*} \max(0, \sigma\mu_k^{(j)}) < 0.$$

This contradicts the fact that x^* is a local minimizer of F, so establishing the inequalities (3.3).

Finally suppose x^* is a strict local minimizer of F. Then exactly as above one establishes conditions (1) and (2). Suppose that condition (3) does not hold. Then there exists an index $k \in \mathscr{A}^*$ which satisfies (3.5) with equality, and a direction s such that $s^T a_k = \sigma$ and $s^T a_i = 0$ for all indices i for which the inequalities in (3.3) are strict. One can then choose a subset Λ of $\mathscr{A}^*$ such that $k \in \Lambda$, the vectors a_i, $i \in \Lambda$ form a basis for $\mathscr{R}(A)$, and $s^T a_i = 0$ for $i \in \Lambda/\{k\}$ implies $s^T a_i = 0$ for all the indices i for which the inequalities (3.3) are strict. Then following the proof above one has a direction s such that $F'(x^*; s) = 0$, contradicting the fact that x^* is a strict local minimizer of F. □

An immediate consequence of the inequalities (3.3) is that

$$|\lambda_i| \leqq 1, \qquad i \in \Gamma_i^*,$$

$$-1 \leqq \lambda_i \leqq 0, \qquad i \in \Gamma_2^*,$$

$$\sum_{j \in \Gamma_3^*} |\mu_i^{(j)}| \leqq \begin{cases} 2, & i \in \Gamma_1^*, \\ 1, & i \in \Gamma_2^*. \end{cases}$$

This theorem includes the degenerate situation when the vectors a_i, $i \in \mathscr{A}^*$ are linearly dependent, in which case the multipliers λ and $\mu^{(j)}$ are not uniquely determined.

4. A reduced gradient algorithm. As F is nonconvex, the censored l_1 approximation problem cannot be posed as a linear programming problem, as the discrete linear l_1 approximation problem can. Thus this section presents a direct descent method based on the reduced gradient method for l_1 approximation. A detailed discussion of the reduced gradient method, and its equivalence to the modified simplex algorithm of Barrodale and Roberts [1], can be found in Osborne [6]. An equally viable alternative, which is not considered here, is to develop a projected gradient algorithm based on the work of Bartels, Conn and Sinclair [2] for the l_1 problem. Osborne [6] also gives a comparison of the reduced and projected gradient methods for linear programming and discrete linear l_1 approximation. Note that the distinction between the reduced and projected gradient algorithms is only of importance whilst a complete active set (one with n elements) is being built up.

Let vectors a_i, $i = 1, \cdots, m$ have rank m_r (in most practical situations $m_r = n$ as $m \gg n$). The basic idea is to generate a sequence of points satisfying sets of equations $r_i(x) = 0$ for $i \in \mathscr{M}$ (an approximation to $\mathscr{A}^*$). The number of elements t in $\mathscr{M}$ increases until $t = m_r$, and thereafter one element at a time is changed until $\mathscr{M} = \mathscr{A}^*$ and the optimality conditions are satisfied. At each point x a step is made in a direction s satisfying $F'(x; s) \leqq 0$ (usually $F'(x; s) < 0$) and $r_i(x + \alpha s) = 0$ for all or all but one $i \in \mathscr{M}$.

Let $\mathscr{M}$ be an index set with t elements such that $\mathscr{M} \subseteq \mathscr{A}$, and let

$$A = [a_i : i \in \mathscr{M}] = [a_{\mathscr{M}(1)} a_{\mathscr{M}(2)} \cdots a_{\mathscr{M}(t)}].$$

It will be shown later that, because of the way elements are added to $\mathscr{M}$, A always has full rank. Define the $n \times n$ matrix B by

$$B = [A|E],$$

where E comprises the columns of the $n \times n$ identity matrix chosen to make B nonsingular. Let

$$H = \text{span}\{a_i : i \in \mathcal{M}\}.$$

The reduced gradient algorithm is based on the fact that the vectors $B^{-T}e_j$ for $j = t+1, \cdots, n$, where $e_j \in \mathbb{R}^n$ is the jth unit vector, form a basis for the orthogonal complement of H, and so provide a suitable basis from which to choose a search direction.

Using the above notation a typical iteration of the basic algorithm can now be given. Each step will then be discussed along with possible improvements. The implementation of the algorithm along with its numerical properties is considered in § 5. The sets Γ_i, $i = 1, \cdots, 5$ are defined by (2.3).

Step 1. Calculate

$$g = \sum_{i \in \Gamma_4} \theta_i a_i, \tag{4.1}$$

where $\theta_i = \text{sign}(r_i(x))$ and $r_i(x) = \max(y_i, z_i) - x^T a_i$.

Step 2. Calculate multipliers: solve

$$Bu = g \quad \text{for } u, \tag{4.2}$$

and

$$B_v^{(j)} = a_j \quad \text{for } v^{(j)}, j \in \Gamma_3. \tag{4.3}$$

Step 3. Check optimality

a) Calculate

$$\delta_i(\sigma) = -\sigma u_i + \sum_{j \in \Gamma_3} \max(0, \sigma v_i^{(j)}) - \eta_i(\sigma), \qquad i = 1, \cdots, n, \tag{4.4}$$

where $\sigma = +1$ and -1, and

$$\eta_i(\sigma) = \begin{cases} 1 & \text{if } \mathcal{M}(i) \in \Gamma_1 \text{ and } 1 \leqq i \leqq t, \\ \max(0, \sigma) & \text{if } \mathcal{M}(i) \in \Gamma_2 \text{ and } 1 \leqq i \leqq t, \\ 0 & \text{if } t+1 \leqq i \leqq n. \end{cases} \tag{4.5}$$

$$\gamma_i = \max(\delta_i(1), \delta_i(-1)), \qquad i = 1, \cdots, n, \tag{4.6}$$

$$p = \text{argmax}\{\gamma_i, i = 1, \cdots, t, \tau\gamma_i, i = t+1, \cdots, n\}, \tag{4.7}$$

$$\sigma_p = \begin{cases} 1 & \text{if } \gamma_i = \eta_i(1), \\ -1 & \text{if } \gamma_i = \eta_i(-1) \end{cases} \tag{4.8}$$

If $\gamma_p > 0$ go to step 4.

If $t = n$ STOP: Rank n termination.

b) If $t < n$ and $J = \{i \in 1, \cdots, m : y_i > z_i > x^T a_i\} \neq \varnothing$ set $j = J(1)$ and

(i) Solve $B\bar{v}^{(j)} = a_j$. (4.9)

(ii) Let

$$p = \text{argmax}\{|\bar{v}_i^{(j)}|, i = t+1, \cdots, n\}, \tag{4.10}$$

$$\sigma_p = \text{sign}(\bar{v}_p^{(j)}). \tag{4.11}$$

(iii) If $|\bar{v}_p^{(j)}| > 0$ go to step 4

Otherwise select next $j \in J$ and go to (i).

If $\bar{v}_i^{(j)} = 0 \; \forall i = t+1, \cdots, n$ and $\forall j \in J$ STOP: Rank deficient termination.

Step 4. Compute search direction

$$s = \sigma_p B^{-T} e_p. \tag{4.12}$$

Step 5. Line search

$$I = \{i \in 1, \cdots, m: i \notin \mathcal{M}, r_i(x)(a_i^T s) > 0\}, \tag{4.13}$$

$$\alpha_i = \frac{r_i(x)}{a_i^T s} \quad \text{for } i \in I. \tag{4.14}$$

$$q = \operatorname{argmin} \{F(x + \alpha_i s), i \in I\}. \tag{4.15}$$

Step 6. Update $x_{\text{new}} = x + \alpha_q s$.
If $p \leqq t$ then $\mathcal{M}_{\text{new}} = \mathcal{M} + \{q\} - \{p\}$
otherwise $\mathcal{M}_{\text{new}} = \mathcal{M} + \{q\}$.
Update B and go to Step 1.

Given a starting point x the algorithm is initialized by setting $\mathcal{M} = \varnothing$, so $t = 0$ and $B = I_n$ (the $n \times n$ identity matrix). Step 1 calculates the gradient of the component functions $|y_i - \max(z_i, x^T a_i)|$ which are smooth at x, whilst step 2 calculates the multipliers u and $v^{(j)}$ for $j \in \Gamma_3$. The usual situation is that Γ_3 is empty, in which case steps 2 and 3 simplify considerably. Step 3 then checks the optimality conditions (3.2) and (3.3), and if they are not satisfied calculates the index p by an unnormalized steepest edge test so that the direction s in step 4 is a descent direction. Step 5 then calculates the next point by a line search along s, where from Lemma 3 the minimum is known to lie at one of a finite number of points.

Step 3 needs further discussion. Consider the nondegenerate case where $\mathcal{M} = \mathcal{A}$, that is $\mathcal{A}(x^{(0)}) = \varnothing$ where $x^{(0)}$ is the starting point and the index q calculated in step 5 is uniquely determined. For the search directions $s^{(i)}$ defined by

$$s^{(i)} = \sigma B^{-T} e_i, \qquad i = 1, \cdots, n, \tag{4.16}$$

where $\sigma = \pm 1$, (2.8), (4.2)-(4.5) yield

$$F'(x; s^{(i)}) = -\delta_i(\sigma) \quad \text{for } i = 1, \cdots, n.$$

If the index p is chosen by

$$p = \operatorname{argmax} \{\gamma_i, i = 1, \cdots, n\},$$

with σ defined by (4.11), then the search direction $s^{(p)}$ minimizes $F'(x; s^{(i)})$ over $i = 1, \cdots, n$ and $\sigma = \pm 1$. If $\gamma_p > 0$, then $F'(x; s^{(p)}) < 0$ so $s^{(p)}$ corresponds to the steepest descent edge direction.

The first part of the optimality conditions is that there exist multipliers satisfying (3.2), that is $g \in H$ and $a_j \in H$ for all $j \in \Gamma_3$. This is true if and only if

$$u_i = 0 \quad \text{and } v_i^{(j)} = 0 \quad \forall j \in \Gamma_3, \quad \text{for } i = t+1, \cdots, n,$$

or equivalently if and only if

$$\gamma_i = 0, \qquad i = t+1, \cdots, n.$$

Note that $\gamma_i \geqq |u_i|$ for $i = t+1, \cdots, n$ with equality if and only if $\Gamma_3 = \phi$. The factor τ in (4.7) is a positive weight (typically $\tau = 100$) to ensure that the columns of E are favoured for deletion until $\gamma_i\, i = t+1, \cdots, n$ are all small or $t = n$ (the usual situation). This permits relaxing off active equations $r_i(x) = 0$ when a very large negative directional derivative would result, before (3.3) is satisfied.

If $t=n$ and $\gamma_i<0$, $i=1,\cdots,n$ then the sufficient conditions are satisfied and x is a strict local minimizer of F. However if $t<n$ then x can be a local minimizer, but there exists a nearby point with the same function value and at which there is a descent direction. This is only likely to occur with relatively small values of m or poor choices of starting point, a typical example being any point $x<0$ in Fig. 4.

Step 3b) thus checks to see if there are any directions along which a line search will result in an increase in t without increasing F. This ensures that $t=m_r=\operatorname{rank}(a_i, i=1,\cdots,m)$, so excluding certain types of nonglobal local minima. Let

$$J=\{j\in 1,\cdots,m: y_i>z_j>x^Ta_j\},$$

and for every $j\in J$ let

$$K^{(j)}=\{i\in t+1,\cdots,n: a_j^Ts^{(i)}\neq 0\}.$$

For any $i\in K^{(j)}$ a line search in the direction $s^{(i)}$ defined in (4.16) with the sign σ chosen so that $a_j^Ts^{(i)}>0$ will increase t without increasing F. For $i\in K^{(j)}$ let

$$\beta^{(i)}=\min\left\{\frac{z_j-x^Ta_j}{a_j^Ts^{(i)}}, j\in J\right\}, \tag{4.17}$$

where σ is always chosen so that $a_j^Ts^{(i)}>0$. If

$$\beta^{(i)}<\min\{\alpha_l: l\in I(x; s^{(i)})\},$$

where I is defined by (3.1), then as $t+1\leqq i\leqq n$ $F'(x; s^{(i)})=0$, moreover

$$F(x+\alpha s^{(i)})=F(x)\quad \forall\alpha\in[0,\beta^{(i)}]$$

and

$$F'(x+\beta^{(i)}s^{(i)}; s^{(i)})=a_k^Ts^{(i)}<0, \tag{4.18}$$

where k is the index which achieves the minimum in (4.17) (if k is not uniquely defined, there will simply be more negative contributions to (4.18)). As the set J can be large, step 3b) simply finds the first $j\in J$ for which $K^{(j)}$ is nonempty. The search direction chosen is that which would give the most negative directional derivative (4.18).

Finally the updating of B must be considered. Let

$$\bar{B}=B+[a_q-Be_p]e_p^T.$$

In order that the structure of B is preserved

$$B_{\text{new}}=\bar{B}P,$$

where if $t+1\leqq p\leqq n$ then P is the permutation matrix which corresponds to swapping the pth column of $\bar{B}$ with the $(t+1)$st. Otherwise P is the identity matrix.

LEMMA 4. *If B is nonsingular, then B_{new} is nonsingular.*

Proof. B_{new} is nonsingular if and only if $\bar{B}$ is nonsingular. Now

$$\bar{B}=B(I+[B^{-1}a_q-e_p]e_p^T).$$

The result follows as

$$\det(I+[B^{-1}a_q-e_p]e_p^T)=1-e_p^T(B^{-1}a_q-e_p)=e_p^TB^{-1}a_q=\operatorname{sign}(u_p)s^Ta_q\neq 0. \qquad \square$$

As B is initially the identity matrix, all matrices B are nonsingular.

In the nondegenerate case the above algorithm is finite, as on every iteration either F decreases or if F does not decrease t (the number of elements in $\mathcal{M}$) increases. Thus

as t never decreases, the sets $\mathcal{M}$ never repeat. As there are only a finite number of possible sets $\mathcal{M}$ the algorithm must terminate in a finite number of iterations.

The degenerate case arises if $\mathcal{M}$ is a strict subset of $\mathcal{A}$, in which case the direction s generated in steps 3a) and 4 may not be a descent direction. Some allowance for this can be made by changing the definition of I in (4.13) to

$$I = \{i \in 1, \cdots, m: i \notin \mathcal{M}, s^T a_i \neq 0 \text{ and } r_i(x)(s^T a_i) \geqq 0\}. \tag{4.19}$$

It is then possible for (4.14) and (4.15) to produce $\alpha_q = 0$, so that an element in $\mathcal{M}$ is exchanged without F decreasing. In this case it is theoretically possible for the sets $\mathcal{M}$ to cycle. However the various techniques available for resolving degeneracy in linear programming can be extended to remove this difficulty.

Some remarks should be made on the use of the unnormalized steepest edge tests (4.4)-(4.7) used to calculate a descent direction. The use of (4.7) assumes that γ_i, $i = 1, \cdots, n$ are of comparable magnitude, and calculates p by computing a minimum over unnormalized directional derivatives $F'(x; s)$. If these are normalized so that $\|s\| = 1$ one obtains

$$p = \operatorname{argmax} \left\{ \frac{\gamma_i}{\|B^{-T} e_i\|} \; i = 1, \cdots, t, \frac{\tau \gamma_i}{\|B^{-T} e_i\|} \; i = t+1, \cdots, n \right\}. \tag{4.20}$$

This test can be computed economically by setting up a recursion for the quantities

$$\chi_i = \|B^{-T} e_i\|^2, \qquad i = 1, \cdots, n. \tag{4.21}$$

One obtains

$$\chi_i^{\text{new}} = \chi_i - 2\zeta_{ip}\phi_i + \zeta^2{}_{ip}\chi_p \tag{4.22}$$

where

$$\phi = B^{-1}(B^{-T} e_p), \quad \psi = B^{-1} a_q \quad \text{and} \quad \zeta_{ip} = (\psi_i - \delta_{ip})/\psi_p \tag{4.23}$$

where δ_{ip} is the Kronecker delta. A complete discussion of the relative merits of unnormalized and normalized steepest edge tests in the context of linear programming and l_1 approximation can be found in Osborne [6].

Although conceptually one of the simplest, step 5 of the algorithm, the line search, is one of the most time-consuming steps for large m, as the number of elements in I in (4.13) is often a significant fraction of m (see § 5 for more comments). A more efficient line search algorithm can be developed as one knows the points at which the slope of $F(x + \alpha s)$ changes and the amount by which it changes. These are the points α_i, $i \in I$ given by (4.14) and (4.19) where a minimum may occur, and at which the slope changes by

$$\Delta g(\alpha_i) = \begin{cases} 2|s^T a_i| & \text{if } y_i > z_i, \\ |s^T a_i| & \text{if } y_i \leqq z_i. \end{cases} \tag{4.24}$$

The only other points where the slope changes are given by

$$\beta_i = \frac{z_i - x^T a_i}{s^T a_i} \quad \text{for } i \in \tilde{I}, \tag{4.25}$$

where

$$\tilde{I} = \{i \in 1, \cdots, m: y_i > z_i \text{ and } (z_i - x^T a_i) s^T a_i > 0\}. \tag{4.26}$$

At these points the change in slope is given by

$$\Delta g(\beta_i) = -|s^T a_i|. \tag{4.27}$$

A local minimizer along the line is characterized by the slope changing from nonpositive to positive upon passing through that point. Let n_I denote the number of elements in the set I. The partition sort techniques of Clark and Osborne [4] can be used to find an α_i corresponding to a local minimum of $F(x+\alpha s)$ in $O(n_I)$ operations. This has the disadvantage that the algorithm is then slightly more likely to converge to a local minimum rather than a global minimum of F. Alternatively if one requires the global minimum along the line, as in step 5 of the algorithm, the α_i, $i \in I$ must be completely sorted. This can be done by a QUICKSORT algorithm taking $O(n_I \log n_I)$ operations. In either case one only needs a partition sort of the β_i, $i \in \tilde{I}$, as for any $k \in I$

$$F'(x+\alpha_k s; s) = -\gamma_p + \sum_{i:\alpha_i \leqq \alpha_k} \Delta g(\alpha_i) + \sum_{i:\beta_i \leqq \alpha_k} \Delta g(\beta_i). \tag{4.28}$$

5. Implementation and numerical experience. Osborne [6] gives a convenient tableau representation for relatively small dense l_1 problems, which can be extended to censored l_1 problems in the following way.

$$W = [I_n | A_F], \tag{5.1}$$

where

$$A_F = [a_1, a_2, \cdots, a_m],$$

and let

$$w^{(k)^T} = [x^{(k)^T} | r^{(k)^T}],$$

where $x^{(k)}$ is the point on the kth iteration and $r^{(k)}$ is the corresponding vector of residuals $r_i^{(k)} - \max(y_i, z_i) - a_i^T x^{(k)}$. Also let

$$B^{(k)} = L^{(k)} U^{(k)},$$

where $L^{(k)}$ is unit lower triangular and $U^{(k)}$ is upper triangular. Define

$$W^{(k)} = L^{(k)^{-1}} W.$$

The key point is that at each stage one works with the tableau $W^{(k)}$. The multipliers $u^{(k)}$ satisfying (4.2) are calculated by solving

$$U^{(k)} u^{(k)} = g^{(k)}$$

by backsubstitution, where

$$g^{(k)} = \sum_{i \in \Gamma_4^{(k)}} \theta_i^{(k)} L^{(k)^{-1}} a_i.$$

Note that the vectors $L^{(k)^{-1}} a_i$ are just certain columns of $W^{(k)}$. Also as $U^{(k)} = L^{(k)^{-1}} B^{(k)}$, the columns of $U^{(k)}$ are simply those columns of $W^{(k)}$ which correspond to the columns of W forming $B^{(k)}$. The vectors $v^{(j,k)}$ satisfying (4.3) are similarly given by a backsubstitution to solve

$$U^{(k)} v^{(j,k)} = L^{(k)^{-1}} a_j.$$

To calculate the search direction, one computes

$$\hat{s}^{(k)} \equiv \sigma_p U^{(k)^{-T}} e_p = L^{(k)^T} s^{(k)},$$

which involves a forward substitution with special structure from the unit vector e_p. Then

$$\Delta w^{(k)^T} \equiv \hat{s}^{(k)^T} W^{(k)} = s^{(k)^T} W = [s^{(k)^T} | s^{(k)^T} A_F].$$

The line search (4.13)-(4.15) requiring $r_i^{(k)} \equiv w_{n+i}^{(k)}$ and $a_i^T s^{(k)} \equiv \Delta w_{n+i}^{(k)}$ provides the steplength $\alpha_q^{(k)}$ and the column a_q to be pivoted into $B^{(k)}$. The vector $w^{(k)}$ is updated by

$$w^{(k+1)} = w^{(k)} + \alpha_q^{(k)} \Delta w^{(k)}.$$

The factors of $B^{(k)}$ are updated by the Bartels-Golub scheme [3]. $W^{(k)}$ must be updated when the pth column of $U^{(k)}$ is deleted and $L^{(k)^{-1}} a_q$ is added. After the appropriate column shifts one has

$$\bar{U}^{(k)} = [U_1^{(k)} \cdots U_{p-1}^{(k)} U_{p+1}^{(k)} \cdots U_n^{(k)} L^{(k)^{-1}} a_q].$$

The subdiagonal elements $\bar{U}_{i+1,i}^{(k)}$ for $i = p+1, \cdots, n$ are zeroed by row operations to produce $U^{(k+1)}$. If necessary rows are interchanged to ensure that the multipliers in the elimination do not exceed one in magnitude. This ensures there is no unnecessary growth of rounding error. It also means that $L^{(k)^{-1}}$ can lose its lower triangular structure, which causes no difficulties as $L^{(k)}$ is never explicitly used.

The quantities (4.22) and (4.23) required to implement the normalized steepest edge test (4.20) can also be efficiently calculated. The details are omitted as the following numerical results refer to the basic algorithm with the unnormalized steepest edge test (4.4)-(4.8) and the basic line search on function values given by (4.13)-(4.15).

To test the numerical performance of the algorithm, some pseudo-random censored estimation problems are generated by the following procedure. A vector $\tilde{x}$ is produced by taking $\tilde{x}_i \in \mathcal{R}[a, b]$, $i = 1, \cdots, n$, where $\mathcal{R}[a, b]$ generates a sequence of numbers uniformly distributed on $[a, b]$. The input invectors a_i, $i = 1, \cdots, m$ are generated in a similar manner. With error terms $\varepsilon_i \in \mathcal{R}[a_e, b_e]$, the observed data values y_i are then generated by (1.2). The algorithm described in § 4 is then used to calculate a local minimizer of (1.1) with $z_i = 0$, $i = 1, \cdots, m$.

For each value of n and m 10 different problems were generated, and for each problem the algorithm was applied from 10 different starting points generated by $x_i^{(0)} \in \mathcal{R}[a, b]$. The results with $a = -10$, $b = 10$, $a_e = -5$ and $b_e = 5$ are collected in Table 1. The reported figures are the minimum number of iterations, the median number of iterations and the maximum number of iterations taken to converge to a local minimum. A figure in brackets indicates the number of nonglobal local minima found and the number of times the algorithm converged to a nonglobal local minimum out of the total of 100 runs for each value of n and m. Only problems with $m > n$ were solved, whilst time restrictions limited the results available for the larger values of n and m.

If the problem has a nonglobal local minimum, then it was usual for several of the starting points to produce convergence to it. Unfortunately there seems to be no way of verifying that a point is a global minimizer, even for the very special case where $a_i \geqq 0$ and $y_i \geqq z_i$ for $i = 1, \cdots, m$.

It was previously remarked that the line search (4.13)-(4.15) based on function values is expensive as the number of points α_i to be checked is usually a significant fraction of m. For the problems in Table 2 the average number of points in the line search is around $m/2$. Thus the line search requires $O(m^2)$ operations. For large m the rest of the algorithm requires $O(nm)$ operations, which is dominated by the line search. Hence a more efficient line search based on a sorting algorithm and the changes

TABLE 2
Results for some random censored estimation problems.

m \ n	2	5	10	15	20	25
10	2 3 5	5 6 9 (7, 14)				
20	2 4 7	5 8 13 (1, 2)	10 13 18 (14, 24)	15 15 20 (39, 43)		
40	2 4 7	6 11 21 (1, 4)	13 20 28	19 25 29 (2, 4)	20 28 46 (32, 38)	25 27 39 (62, 67)
60	2 4 7	6 13 21 (4, 18)	18 24 34 (2, 4)	21 33 44 (2, 4)	25 39 54 (5, 18)	31 43 71 (12, 21)
80	2 4 8 (1, 2)	8 13 20 (3, 12)	18 27 37 (1, 8)	27 38 49 (1, 2)	32 46 63 (4, 13)	41 50 69 (3, 10)
100	2 4 9 (1, 1)	8 14 25 (3, 15)	20 29 41	29 41 58 (2, 7)	38 52 71 (3, 6)	45 59 79 (1, 1)
200	2 5 10	11 18 27 (2, 6)	22 36 51 (2, 12)	37 52 66 (3, 8)	51 69 90 (1, 3)	72 90 113 (2, 8)
400	3 7 12	11 20 36	31 43 59 (1, 3)	48 65 89 (1, 3)	68 87 113 (2, 14)	80 103 126 (4, 14)
600	3 6 11	9 22 30	32 45 63 (1, 3)	55 70 93 (1, 2)	74 95 116 (1, 1)	
800	2 6 12	14 23 32 (1, 2)	33 49 70 (9, 23)			
1000	3 7 10	13 23 33 (1, 4)				

(4.24) to (4.28) in the directional derivative, which would require $O(m)$ or $O(m \log m)$ operations, should be used.

Finally to check consistency ($x_m^* \to \tilde{x}$ as $m \to \infty$, where x_m^* is the global minimizer of (1.1)) and $\sqrt{m}$-consistency ($\sqrt{m}\,\|\tilde{x} - x_m^*\|$ is bounded in probability) the average values of $\|\tilde{x} - x_m^*\|$ for $n = 5$ (which is typical of the other values of n) for the 10 different problems of Table 2 are listed in Table 3.

TABLE 3
Average values of $\|x_m^* - \tilde{x}\|$ *for* $n = 5$.

m	10	20	40	60	80	100
$\|x_m^* - \tilde{x}\|$	4.324	.679	.389	.277	.290	.255
$\sqrt{m}\,\|x_m^* - \tilde{x}\|$	13.7	3.04	2.46	2.15	2.59	2.55
m	200	400	600	800	1000	
$\|x_m^* - \tilde{x}\|$	.179	.147	.130	.099	.095	
$\sqrt{m}\,\|x_m^* - \tilde{x}\|$	2.53	2.94	3.18	2.80	3.00	

Finally Table 4 gives the results for the motorette example given in § 1. For this problem the function (1.1) has a nonunique global minimizer, so the points listed in Table 4 are the global minimizers x^* characterized by $r_i(x^*) = 0$ for $i \in \mathcal{A}(x^*)$. From all starting points tried the algorithm converged to one of these points in 2 or 3 iterations. Note that observations with identical data values z_i, y_i and a_i were grouped together to minimize the effects of degeneracy. The iterated least squares solution

reported in [11] is $x^* = (-5.818, 4.204)^T$, whilst the maximum likelihood solution is $x^* = (-6.027, 4.314)$. Both these points are very close to the convex hull of the points in Table 4.

TABLE 4
Global minima for motorette example.

$\mathscr{A}^*$	x_1^*	x_2^*
1, 6	−3.386	3.086
1, 7	−.967	2.062
6, 13	−4.578	3.615
6, 15	−5.054	2.826
6, 16	−4.855	3.737
7, 15	−6.022	4.303
7, 16	−5.822	4.214
13, 15	−5.371	3.982
13, 16	−5.039	3.828

Acknowledgment. The author wishes to thank M. R. Osborne for bringing this problem to his attention, and for many useful discussions.

REFERENCES

[1] I. BARRODALE AND F. D. K. ROBERTS, *An improved algorithm for discrete L_1 linear approximation*, SIAM J. Numer. Anal., 10 (1973), pp. 839-848.

[2] R. H. BARTELS, A. R. CONN AND J. W. SINCLAIR, *Minimization techniques for piecewise differentiable functions: the L_1 solution of an overdetermined linear system*, SIAM J. Numer. Anal., 15 (1978), pp. 224-241.

[3] R. H. BARTELS, G. H. GOLUB AND M. A. SAUNDERS, *Numerical techniques in mathematical programming*, Nonlinear Programming, J. B. Rosen, O. L. Mangasarian and K. Ritter, eds., Academic Press, New York and London, 1970.

[4] D. I. CLARK AND M. R. OSBORNE, *A descent algorithm for minimizing polyhedral convex functions*, this Journal, 4 (1983), pp. 757-786.

[5] F. H. CLARKE, *Generalized gradients and applications*, Trans. Amer. Math. Soc., 205 (1975), pp. 247-262.

[6] M. R. OSBORNE, *Finite Algorithms in Optimization and Data Analysis*, John Wiley, New York, to appear.

[7] H. J. PAARSCH, *A Monte Carlo comparison of estimators for censored regression models*, Ann. Econometrics, 24 (1984), pp. 197-214.

[8] J. L. POWELL, *Least absolute deviations estimation for censored and truncated regression models*, Institute for Mathematical Studies in the Social Sciences, the Economic Series, Technical Report 356, Stanford Univ., Stanford, CA, 1981.

[9] ———, *Generalizations of the censored and truncated least absolute deviations estimators.* Institute for Mathematical Studies in the Social Sciences, the Economic Series, Technical Report 398, Stanford Univ., Stanford, CA, 1983.

[10] ———, *Least absolute deviations estimation for the censored regression model*, J. Econometrics, 25 (1984), pp. 303-325.

[11] J. SCHMEE AND G. J. HAHN, *A simple method for regression analysis with censored data*, Technometrics, 21 (1979), pp. 417-432.

[12] G. A. WATSON, *Approximation Theory and Numerical Methods*, John Wiley, Chichester and New York, 1980.

SIAM J. SCI. STAT. COMPUT.
Vol. 7, No. 1, January 1986

009

COMPUTING THE MINIMUM EIGENVALUE OF A SYMMETRIC POSITIVE DEFINITE TOEPLITZ MATRIX*

GEORGE CYBENKO† AND CHARLES VAN LOAN‡

Abstract. A method for computing the smallest eigenvalue of a symmetric positive definite Toeplitz matrix is given. It relies solely upon the Levinson-Durbin algorithm. The procedure involves a combination of bisection and Newton's method. Good starting values are also shown to be obtainable from the Levinson-Durbin algorithm.

Key words. Toeplitz, eigenvalue, signal processing

1. Introduction. Recent progress in signal processing and estimation has generated considerable interest in the problem of computing the minimal eigenvalue of a Toeplitz matrix. The fundamental modeling and solution that led to this are due to Pisarenko [P73], while more recently numerous authors have discussed the computational aspects of the problem [F83], [H80], [H83a], [H83c].

In this paper we shall not discuss the underlying assumptions, merits, or potential applications of the model—instead pointing the interested reader to the literature concerned with these issues [H83a], [P73]. We hasten to add that the quantities of ultimate interest in applications are the roots of the polynomial whose coefficients are given by the eigenvector associated with the minimal eigenvalue of the Toeplitz matrix. We shall only discuss the computation of the minimal eigenvalue noting that the associated eigenvector can be obtained as a by-product. Furthermore, methods exist for computing the roots that altogether avoid the explicit formation of the eigenvector [C84b].

The essence of our minimum eigenvalue procedure involves solving systems of shifted Yule-Walker (YW) systems. Initially, the solutions to these systems are used in a bisection scheme that repeatedly halves a bracketing subinterval. Subsequently, a Newton iteration takes over that quadratically converges to the desired eigenvalue. We stress the fact that only YW systems are involved—an important point since extremely efficient methods for YW systems exist. (They require half the computational resources needed by general symmetric Toeplitz system solvers.)

In an absolute sense, only modest use is made of Toeplitz structure. Indeed, this is true of all currently known Toeplitz eigenvalue solvers. The study of the eigenstructure of finite Toeplitz matrices is proceeding rather slowly. Recent developments include [C84a], [C84b], [D83]. An indication of the collective ignorance about Toeplitz eigenstructure is that the inverse eigenvalue problem for real symmetric Toeplitz matrices is currently unsolved. We suspect that the process of designing efficient algorithms for this problem will go hand in hand with the uncovering of Toeplitz eigenstructure properties.

Our paper is organized as follows. Section 2 describes a rational function intimately related to the eigenvalue problem for Hermitian matrices. Section 3 specializes the

* Received by the editors May 9, 1984.

† Department of Mathematics, Tufts University, Medford, Massachusetts 02155 and the Statistics Center, Massachusetts Institute of Technology, Cambridge, Massachusetts 02135. The work of this author was supported in part by the National Science Foundation under contract NSF MCS 8003364.

‡ Department of Computer Science, Cornell University, Ithaca, New York 14853. The work of this author was supported in part by the National Science Foundation under contract NSF MCS 8004106, and by the Office of Naval Research under contract ONR N00014-83-K-0640.

discussion to real symmetric Toeplitz matrices and then develops our bisection/Newton scheme. In the last section we discuss the numerical behavior of our procedure.

2. A rational eigenvalue equation. In this section we derive a rational function from a given Hermitian matrix that has the property that its zeros are eigenvalues of the original matrix. A feature of this rational function is that both it and its derivatives are easily evaluated thereby making Newton-type schemes feasible. Strictly speaking, parts of our derivation are not new and can be found in [W65], but we present the details for the sake of completeness.

Let T be an $n \times n$ Hermitian matrix partitioned as follows:

$$T = \begin{bmatrix} \gamma & r^* \\ r & G \end{bmatrix} \begin{matrix} 1 \\ n-1 \end{matrix}.$$

$$\begin{matrix} 1 & n-1 \end{matrix}$$

Here, r^* denotes the conjugate transpose of r. It is well known [G83] that the eigenvalues of T and G are real and satisfy an interlacing property. In particular, if $\lambda_i(T)$ and $\lambda_i(G)$ are the ith largest eigenvalues of T and G respectively then

$$\lambda_n(T) \leqq \lambda_{n-1}(G) \leqq \lambda_{n-1}(T) \leqq \cdots \leqq \lambda_2(T) \leqq \lambda_1(G) \leqq \lambda_1(T).$$

Note that if T has a repeated eigenvalue, then the repeated value is also an eigenvalue of G. Adopting the notation

$$\lambda_{\min} \equiv \lambda_n(T),$$

we shall assume throughout this paper that

$$d = \lambda_{n-1}(G) = \lambda_{\min} > 0. \tag{2.1}$$

The strict separation of $\lambda_{n-1}(G)$ and $\lambda_{\min}$ guarantees that the eigenvector associated with $\lambda_{\min}$ is unique up to scalar premultiplication. It is a realistic assumption in many important problems such as the estimation of Pisarenko frequencies. See [C84b].

Suppose

$$\begin{bmatrix} \gamma & r^* \\ r & G \end{bmatrix} \begin{bmatrix} \alpha \\ y \end{bmatrix} = \lambda_{\min} \begin{bmatrix} \alpha \\ y \end{bmatrix} \tag{2.2}$$

where it is assumed that α and y are not both zero. From this equation we obtain

$$\begin{aligned} \gamma\alpha + r^* y &= \lambda_{\min}\alpha, \\ \alpha r + Gy &= \lambda_{\min} y. \end{aligned} \tag{2.3}$$

We must have $\alpha \neq 0$ for otherwise $Gy = y_{\min} y$ contradicting (2.1). Noting that $G - \lambda_{\min} I$ is positive definite we obtain the following rational equation for $\lambda_{\min}$:

$$\gamma - r^*(G - \lambda_{\min} I)^{-1} r - \lambda_{\min} = 0. \tag{2.4}$$

Thus, the smallest eigenvalue of T is the smallest root of the rational function

$$f(\lambda) = \gamma - \lambda - r^*(G - \lambda I)^{-1} r. \tag{2.5}$$

In addition, $f(\lambda)$ has the following important properties if $0 \leqq \lambda < \lambda_{n-1}(G)$:

$$f'(\lambda) = -1 - \|(G - \lambda I)^{-1} r\|_2^2 \leqq -1, \tag{2.6}$$

$$f''(\lambda) = -2r^*(G - \lambda I)^{-3} r \leqq 0. \tag{2.7}$$

Now consider the following Newton iteration:

ALGORITHM 2.1. Let $\lambda \in [\lambda_{\min}, \lambda_{n-1}(G))$ be given along with a tolerance $\delta > 0$.
Do Until $(|f(\lambda)| < \delta/(1+\|w\|_2^2)^{1/2})$

Solve $(G-\lambda I)w = -r$ for w.

$$\lambda := \lambda + \frac{\gamma + r^*w - \lambda}{1+w^*w} = \lambda - \frac{f(\lambda)}{f'(\lambda)}.$$

Properties (2.6) and (2.7) ensure that the iteration converges to $\lambda_{\min}$. To see this assume that $\lambda \in (\lambda_{\min}, \lambda_{n-1}(G))$ and set

$$\lambda_+ = \lambda - \frac{f(\lambda)}{f'(\lambda)}.$$

Since f is monotone decreasing in this interval, it follows that both $f(\lambda)$ and $f'(\lambda)$ are negative. Thus, $\lambda_+ < \lambda$. On the other hand, from truncated Taylor series we have

$$0 = f(\lambda_{\min}) = f(\lambda) + f'(\lambda)(\lambda_{\min} - \lambda) + \frac{f''(\zeta)}{2}(\lambda_{\min} - \lambda)^2$$

with $\zeta \in [\lambda_{\min}, \lambda]$. It follows that

$$\lambda_+ - \lambda_{\min} = \frac{f''(\zeta)}{2f'(\lambda)}(\lambda_{\min} - \lambda)^2 > 0. \tag{2.8}$$

Thus, the iterates in the algorithm converge monotonically to $\lambda_{\min}$ from the right and at a rate that is ultimately quadratic. Note from (2.8) that in the limit we have

$$(\text{error in new } \lambda) \cong C \cdot (\text{error in old } \lambda)^2$$

where

$$C = \frac{f''(\lambda_{\min})}{2f'(\lambda_{\min})} = \frac{w^*(G-\lambda_{\min}I)^{-1}w}{1+w^*w}$$

and $w = -(G-\lambda_{\min}I)^{-1}r$. Since $\|(G-\lambda_{\min}I)^{-1}\|_2 = 1/d$ it is easy to show that $C \leqq 1/d$. It follows that Algorithm 2.1 may converge slowly in problems where the separation d is small. We return to this point later.

The termination criteria in Algorithm 2.1 gives good absolute error in the final λ provided the tolerance δ is small enough. This follows from

$$\left\| \begin{bmatrix} \gamma & r^* \\ r & G \end{bmatrix} \begin{bmatrix} 1 \\ w \end{bmatrix} - \lambda \begin{bmatrix} 1 \\ w \end{bmatrix} \right\|_2 = \left\| \begin{bmatrix} f(\lambda) \\ 0 \end{bmatrix} \right\|_2 = |f(\lambda)|.$$

Applying standard Hermitian matrix perturbation theory (see [G83]), we may conclude that there exists an exact eigenvalue λ_e of T that satisfies

$$|\lambda - \lambda_e| < |f(\lambda)|(1+\|w\|_2^2)^{1/2} < \delta.$$

If δ is sufficiently small compared to the separation d, then one can ensure that $\lambda_e = \lambda_{\min}$.

Despite the nice mathematical properties of Algorithm 2.1, its practical value hinges on two critical factors: how is the starting value determined and how is the linear system $(G-\lambda I)w = -r$ to be solved? We address these questions in the next section for the case when T is symmetric, positive definite, and Toeplitz.

3. The symmetric positive definite Toeplitz case. Let $(t_0, t_1, \cdots, t_{n-1})$ be the first row of a symmetric positive definite Toeplitz matrix $T = (t_{ij})$, i.e., $t_{ij} = t_{|i-j|}$. Assume

that T is normalized so that $t_0 = 1$ and partition it as follows:

$$T = \begin{bmatrix} 1 & r^T \\ r & G \end{bmatrix}, \qquad r^T = (t_1, \cdots, t_{n-1}).$$

Recall that in order to apply Algorithm 2.1 we must find a starting value λ that belongs to the interval $[\lambda_n(T), \lambda_{n-1}(G))$. This requirement can be couched in the language of signatures. The signature sig (A) of a symmetric matrix A is a triplet of integers (neg, z, pos) where neg, z, and pos are the number of negative, zero and positive eigenvalues of A. Our starting value problem is to find λ such that sig $(G - \lambda I) = (0, 0, n-1)$ while sig $(T - \lambda I) = (1, 0, n-1)$ or $(0, 1, n-1)$.

This problem can be solved by exploiting the well-known Levinson-Durbin algorithm:

ALGORITHM 3.1.

$E_0 = 1$

For $i = 1$ to $n-1$

$$k_i = -\left(t_i + \sum_{j=1}^{i-1} a_{i-1,j} t_j \right) \Big/ E_{i-1}$$

For $j = 1$ to $i-1$

$$a_{ij} = a_{i-1,j} + k_i a_{i-1,i-j}$$

$$a_{ii} = k_i$$

$$E_i = E_{i-1}(1 - k_i^2)$$

The a_{ij} satisfy the Yule-Walker (YW) systems

$$\begin{bmatrix} 1 & t_1 & \cdots & t_{i-1} \\ t_1 & 1 & & \vdots \\ \vdots & & \ddots & \\ t_{i-1} & \cdots & & 1 \end{bmatrix} \begin{bmatrix} a_{i1} \\ \vdots \\ a_{ii} \end{bmatrix} = - \begin{bmatrix} t_1 \\ \vdots \\ t_i \end{bmatrix}$$

for $i = 1, \cdots, n-1$. The quantities k_i and E_i are referred to as the ith partial correlation coefficient and the ith prediction error respectively. (k_i is also known as the ith reflection coefficient.) See [G83] for a discussion of Algorithm 3.1.

In [C80] it is shown that if

$$L = \begin{bmatrix} 1 & & & & \\ a_{11} & 1 & & 0 & \\ a_{22} & a_{21} & 1 & & \\ \vdots & & & \ddots & \\ a_{n-1,n-1} & & \cdots & & 1 \end{bmatrix}$$

then

$$LTL^T = \text{diag}\,(1, E_1, \cdots, E_{n-1}). \tag{3.1}$$

Since signature is preserved under congruence transformations by the Sylvester Law of Inertia, all of the E_i are positive since T is positive definite.

However, if we apply Algorithm 3.1 to the normalized Toeplitz matrix $(T - \lambda I)/(1 - \lambda)$ and if the algorithm runs to completion, then the number of negative E_i that are generated equals the number of eigenvalues of $T - \lambda I$ that are negative, i.e., the number of T's eigenvalues that are strictly less than λ. The caveat "runs to completion" must be added because it is possible for one of the E_i to be zero if Algorithm 3.1 is applied to an indefinite T. Adapting the algorithm so that computes

(3.1) for $T := (T - \lambda I)/(1-\lambda)$ gives

ALGORITHM 3.2.

$i = 0$

$E_0 = 1$

Do While $(E_i > 0 \ \& \ i < n-1)$

$$i := i+1$$

$$k_i = -\left(t_i + \sum_{j=1}^{i-1} a_j t_j\right) \Big/ [(1-\lambda)E_{i-1}]$$

$$\begin{bmatrix} a_1 \\ \vdots \\ a_{i-1} \\ a_i \end{bmatrix} := \begin{bmatrix} a_1 \\ \vdots \\ a_{i-1} \\ 0 \end{bmatrix} + k_i \begin{bmatrix} a_{i-1} \\ \vdots \\ a_1 \\ 1 \end{bmatrix}$$

$$E_i = E_{i-1}(1-k_i^2)$$

We have dropped the double subscripting of the a's since we need only be in possession of the most recent YW solution at any one time.

Note that if the loop terminates because $i = n-1$, then we have

$$(G - \lambda I)\begin{bmatrix} a_1 \\ \vdots \\ a_{n-1} \end{bmatrix} = -\begin{bmatrix} t_1 \\ \vdots \\ t_{n-1} \end{bmatrix}.$$

Recall that being able to solve this shifted YW system is critical to Algorithm 2.1, the Newton iteration for $f(\lambda)$.

Equally important, the final value of i in Algorithm 3.2 enables us to determine the position of λ with respect to $\lambda_{\min}$ and $\lambda_{n-1}(G)$:

(a) If $i = n-1$ and $E_{n-1} > 0$, then $\lambda < \lambda_{\min}$.

(b) If $i = n-1$ and $E_{n-1} \leqq 0$, then $\lambda_{\min} \leqq \lambda < \lambda_{n-1}(G)$.

(c) If $i < n-1$ then $\lambda_{n-1}(G) \leqq \lambda$.

Hence, Algorithm 3.2 can be used in a bisection scheme to position λ eventually in the interval $[\lambda_{\min}, \lambda_{n-1}(G))$. Thereafter, it can be used to carry out the Newton iteration. All we need is an initial interval $[\alpha, \beta]$ with the property that

$$\alpha \leqq \lambda_{\min} \leqq \beta. \tag{3.2}$$

ALGORITHM 3.3.

Compute α and β satisfying (3.2) and let $\delta > 0$ be a given tolerance.

$k = 0$

$\lambda^{(0)} = (\alpha + \beta)/2$

Do Until $(|\lambda^{(k)} - \lambda^{(k-1)}| \leqq \delta |\lambda^{(k-1)}|)$

Apply Algorithm 3.2 with $\lambda = \lambda^{(k)}$ to generate i and $a_1, \cdots, a_i$.

$k := k+1$

If $(i < n-1)$

then

$\beta = \lambda$; $\lambda^{(k)} = (\alpha + \beta)/2$

else

If $(E_{n-1} > 0)$

then

$\alpha = \lambda$; $\lambda^{(k)} = (\alpha + \beta)/2$

else

$$\lambda^{(k)} = \lambda + \frac{1 - \lambda + t_1 a_1 + \cdots + t_{n-1} a_{n-1}}{1 + a_1^2 + \cdots + a_{n-1}^2}$$

The last expression for $\lambda^{(k)}$ above is the same as the λ update expression in Algorithm 2.1 with $\gamma = 1$, $r^T = (t_1, \cdots, t_{n-1})^T$ and $w^T = (a_1, \cdots, a_{n-1})^T$.

There are several possible ways to choose the initial bracketing interval $[\alpha, \beta]$. For example, we could set $[\alpha, \beta] = [0, 1]$. The choice for β follows from the inequality $\lambda_{\min} \leqq e_1^T T e_1 = 1$ where $e_1 = (1, 0, \cdots, 0)^T$. There are, however, more refined ways to get the initial interval:

Method 1. See $[\alpha, \beta] = [0, 1 - |t_1|]$. Since the smallest eigenvalue of

$$T_1 = \begin{bmatrix} 1 & t_1 \\ t_1 & 1 \end{bmatrix}$$

is given by $1 - |t_1|$, we have from separation theory that $1 - |t_1| = \lambda_2(T_1) \geqq \lambda_n(T)$.

Method 2. Set $[\alpha, \beta] = [0, \min_i \{1 - |t_i|\}]$. The reasoning is the same as for Method 1 with t_1 replaced by t_i. Note that $\begin{bmatrix} 1 & t_i \\ t_i & 1 \end{bmatrix}$ is a principal submatrix of T.

Method 3. Set $[\alpha, \beta] = [0, E_{n-2}(1 - |k_{n-1}|)]$ where E_{n-2} and k_{n-1} are generated by Algorithm 3.3 with λ_0. To understand the choice for β, consider the effect of replacing t_{n-1} with $\tilde{t}_{n-1} = t_{n-1} + \varepsilon$ in Algorithm 3.3 and that we set $\lambda = 0$. Nothing changes except during the last pass through the loop when we compute

$$\tilde{k}_{n-1} = -\left(\tilde{t}_{n-1} + \sum_{j=1}^{n-2} a_j t_j\right) \Big/ E_{n-2} = -\frac{\varepsilon}{E_{n-2}} + k_{n-1}.$$

Note that if we choose ε so that $1 - \tilde{k}_{n-1}^2 = 0$, then the resulting $\tilde{E}_{n-1}$ will be zero. Thus, a perturbation of size ε transforms T into a singular matrix. It follows that $\lambda_n(T) \leqq \varepsilon$. The choice for β is the smaller of the two ε values that render $\tilde{k}_{n-1}^2 = 1$.

Method 4. Set

$$[\alpha, \beta] = \left[\frac{1}{\sqrt{n}\,\|T^{-1}\|_\infty}, \frac{\sqrt{n-1}}{\|G^{-1}\|_\infty}\right].$$

The value for α follows from the inequality

$$\frac{1}{\lambda_n(T)} = \|T^{-1}\|_2 \leqq \sqrt{n}\,\|T^{-1}\|_\infty.$$

Here, $\|\cdot\|_\infty$ denotes maximum row sum. The value for β follows from

$$\frac{1}{\lambda_n(T)} > \frac{1}{\lambda_{n-1}(G)} = \|G^{-1}\|_2 \geqq \sqrt{n-1}\,\|G^{-1}\|_\infty.$$

The quantities $\|T^{-1}\|_\infty$ and $\|G^{-1}\|_\infty$ can be calculated in $O(n^2)$ operations and $O(n)$ storage using the Trench algorithm [T64].

4. Analysis, discussion and numerical experiments. Another method for finding $\lambda_{\min}$ via the Levinson-Durbin algorithm is presented in [H80]. They propose solving $f(\lambda) = 0$ ($E_n(\mu) = 0$ in their notation) using a linear interpolation scheme. A key aspect of our work and what distinguishes it from [H80] is the recognition that one can apply Newton's method using by-products from the Levinson-Durbin algorithm. In addition, we have attempted to handle the problem of starting values more rigorously than [H80].

The importance of only having to solve Yule-Walker systems should be stressed. Methods based on inverse iteration, for example, require the solution of general Toeplitz systems. This doubles the amount of work per step. Moreover, there currently exist highly concurrent algorithms and VLSI architectures for solving Yule-Walker systems in $O(n)$ time. See [K83].

The total amount of work required by Algorithm 3.3 is determined by the number of YW systems that must be solved. The number N_B of bisection steps is bounded above by

$$N_B \leqq -\log_2 [d/(\beta - \alpha)] + 1.$$

Note that during this phase of the algorithm, calls to Algorithm 3.2 do not require a full $n-1$ steps so it is a little hard to quantify the overall work. As a function of n, N_B appears to grow as $\log(n)$. A simple explanation of this is possible if we assume that the eigenvalues of T are uniformly distributed. In this case, the distance $\lambda_{n-1}(G) - \lambda_{\min}$ is roughly $1/n^2$. Hence the worst case limit on N_B is proportional to $\log(n)$.

The number of Newton steps N_N tends to be around 5 or 6 based on our experience with numerous examples a subset of which we now describe. For each $n = 11, 21, \cdots, 91$ we generated 25 random positive definite symmetric Toeplitz matrices. These matrices had the form

$$T = m \sum_{k=1}^{n} w_k T_{2\pi\theta_k}$$

where n is the dimension, m is chosen so that T is normalized,

$$T_\theta = (t_{ij}) = (\cos(\theta(i-j))),$$

and the w_k and θ_k are uniformly distributed random numbers taken from $[0, 1]$. It can be shown that T_θ is rank two, symmetric, semidefinite, and Toeplitz.

Table 1 summarizes the results of these experiments. Only initial interval Methods 1 and 3 were tabulated. Methods 2 and 4 were too similar in performance to Methods 1 and 3 for us to report. (Note: in recording the work associated with Method 3 the single call to Algorithm 3.2 required to compute β is accounted for in the table.)

TABLE 1
Behavior of Algorithm 3.3 ($\delta = 10^{-6}$) *based on* 25 *random examples per dimension.*

	Starting values via Method 1		Starting values via Method 3	
Order	Bisection steps	Newton steps	Bisection steps	Newton steps
11	4.8	5.0	5.7	4.8
21	8.5	5.7	4.8	5.4
31	9.7	5.3	6.6	5.0
41	10.0	4.6	6.7	5.2
51	11.7	5.8	8.1	5.5
61	12.1	5.0	9.2	5.3
71	12.0	5.3	9.2	5.2
81	13.6	5.4	9.8	5.0
91	11.6	5.0	8.4	5.7

The matrices were generated by a Fortran program and the eigenvalues $\lambda_{\min}$ and $\lambda_{n-1}(G)$ were computed by the EISPACK routine RS [S76]. Although our generation technique was guaranteed to generate at least a semi-definite matrix (definite with probability one) rounding errors led to a generation of some isolated slightly indefinite cases. Although indefinite matrices (due to finite arithmetic) ought to be expected in practice, they provide no realistic test for our procedure. In fact, if we know that our

data has t significant bits (floating point), then more than t calls to the bisection step is useless. Given the quality of our data, after t steps of bisection we must conclude that the matrix is either not definite or that the condition $|\lambda_{n-1}(G) - \lambda_{\min}| < 2^{-t}$ holds and so to the precision of our data, $\lambda_n(G) = \lambda_{\min}$. This follows from standard eigenvalue perturbation arguments [P80].

While on the subject of small separations, it is interesting to point out that $1/d$ measures the sensitivity or "condition" of $\lambda_{\min}$'s eigenvector $x_{\min}$. If this quantity is large then small perturbations in T can induce large changes in $x_{\min}$. (See [G83, p. 271].) As we mentioned in the introduction, the computation of $\lambda_{\min}$ is frequently just the first step in computing $x_{\min}$, the "real" quantity of interest. Thus, slow convergence in Algorithm 3.3 goes hand in hand with ill-conditioning in the underlying $x_{\min}$ problem.

The actual procedure was implemented in C on a DEC-10. Computations were done in the "double" data type. The tolerance δ in Algorithm 3.3 was set to 10^{-6}. The Newton iteration terminated successfully on all strictly separated trials and gave six significant digit agreement with EISPACK generated solutions.

Finally, we recommend Method 3 among the various procedures that we described for obtaining an initial interval. However, it is conceivable that the simplicity of Method 1 might make it more appealing than Method 3 in real time processing situations with elementary processors.

Acknowledgment. We are each indebted to the useful recommendations of the referees.

REFERENCES

[B80] R. BITMEAD AND B. D. O. ANDERSON, *Asymptotically fast solutions of Toeplitz and related systems of equations*, Lin. Alg. Appl., 34 (1980), pp. 103-116.

[B82] R. BRENT AND F. LUK, *A systolic array for linear-time solution of Toeplitz systems of equations*, Cornell Computer Science Technical Report TR82-526, Ithaca, NY, 1982.

[C80] G. CYBENKO, *The numerical stability of the Levinson-Durbin algorithm for Toeplitz systems of equations*, this Journal, 1 (1980), pp. 303-319.

[C84a] ———, *On the eigenstructure of Toeplitz matrices*, IEEE Trans. Acoust., Speech, SIgnal Proc., ASSP, 31 (1984), pp. 918-920.

[C84b] ———, *Computing Pisarenko frequency estimates*, Proc. 1984 Princeton Conference on Information Systems and Sciences, March 14-16, 1984, pp. 587-591.

[D82] T. S. DURRANI AND K. C. SHARMAN, *Excitation of an eigenvector oriented spectrum from the MESA coefficients*, IEEE Trans. Acoust., Speech, Signal Proc., ASSP, 30 (1982), pp. 649-651.

[D83] P. DELSARTE AND Y. GENIN, *Spectral properties of finite Toeplitz matrices*, Proc. 1983 Mathematical Theory of Networks and Systems, Beer-Sheva, Israel, 1983.

[F83] D. FUHRMAN AND BEDE LIU, *Approximating the eigenvectors of a symmetric Toeplitz matrix*, 1983, preprint.

[G83] G. H. GOLUB AND C. VAN LOAN, *Matrix Computations*, Johns Hopkins Univ. Press, Baltimore, 1983.

[H80] Y. H. HU AND S. Y. KUNG, *Computation of the minimum eigenvalue of a Toeplitz matrix by the Levinson algorithm*, Proceedings SPIE 25th International Conference (Real Time Signal Processing), San Diego, August 1980, pp. 40-45.

[H83a] ———, *A Toeplitz eigensystem solver*, 1983, preprint.

[H83b] Y. H. HU, *Resolution enhanced spectrum estimation via Toeplitz eigensystem solvers*, Proceedings 22nd IEEE Conference on Decision and Control, San Antonio, TX, Dec. 14-16, 1983, pp. 1345-1346.

[H83c] Y. H. HU AND S. Y. KUNG, *Highly concurrent Toeplitz eigensystem solver for high resolution spectral estimation*, Proc. IEEE 1983 ICASSP, Boston, 1983, pp. 1422-1425.

[K83] S. Y. KUNG AND Y. H. HU, *A highly concurrent algorithm and architecture for solving Toeplitz systems*, IEEE Trans. Acoust., Speech, Signal Proc., ASSP-31 (1983), pp. 66-76.

[P73] V. F. PISARENKO, *The retrieval of harmonics from a covariance function*, Geophys. J. Royal Astron. Soc., 33 (1973), pp. 347–366.

[P80] B. N. PARLETT, *The Symmetric Eigenvalue Problem*, Prentice-Hall, Englewood-Cliffs, NJ.

[S76] B. T. SMITH et al., *Matrix Eigensystem Routines*—EISPACK *Guide*, Springer-Verlag, New York, 1976.

[T64] W. J. TRENCH, *An algorithm for the inversion of finite Toeplitz matrices*, J. Soc. Ind. Appl. Math., 12 (1964), pp. 515–522.

[W65] J. WILKINSON, *The Algebraic Eigenvalue Problem*, Oxford, New York, 1965.

SIAM J. SCI. STAT. COMPUT.
Vol. 7, No. 1, January 1986

010

SOLUTION OF SYSTEMS OF COMPLEX LINEAR EQUATIONS IN THE l_∞ NORM WITH CONSTRAINTS ON THE UNKNOWNS*

ROY L. STREIT†

Abstract. An algorithm for the numerical solution of general systems of complex linear equations in the l_∞, or Chebyshev, norm is presented. The objective is to find complex values for the unknowns so that the maximum magnitude residual of the system is a minimum. The unknowns are required to satisfy certain convex constraints; in particular, bounds on the magnitudes of the unknowns are imposed. In the algorithm presented here, this problem is replaced by a linear program generated in such a way that the relative error between its solution and a solution of the original problem can be estimated. The maximum relative error can easily be made as small as desired by selecting an appropriate linear program. Order of magnitude improvements in both computation time and computer storage requirements in an implementation of the simplex algorithm to this linear program are presented. Three numerical examples are included, one of which is a complex function approximation problem.

Key words. complex linear equations, Chebyshev solution, convex constraints, complex approximation, semi-infinite programming, l_∞ norm

1. Introduction. The numerical solution of general systems of complex linear equations in the l_∞, or Chebyshev, norm is a mathematical problem that arises in several applications. The objective is to find complex values for the unknowns so that the maximum magnitude residual of the system of equations is minimized. The unknowns are not allowed to assume any complex value whatever; instead, they are required to satisfy convex constraints of the form that can occur in the applications.

Let n, m, and r be positive integers. Let the matrices $A \in C^{n\times m}$, $B \in C^{n\times r}$, and the *row* vectors $f \in C^m$, $g \in C^r$, $a \in C^n$, $d \in R^n$, and $c \in R^r$ be given. The vector of unknowns, $z \in C^n$, is also taken to be a *row* vector. (Row instead of column vectors are used for notational convenience in § 2.) The problem is stated as follows.

Problem.

$$\min_{z\in C^n} \|zA - f\|_\infty \tag{1}$$

subject to:

$$|z - a| \leqq d, \tag{2}$$

$$|zB - g| \leqq c, \tag{3}$$

where the Chebyshev norm $\|\cdot\|_\infty$ of a vector is the maximum modulus of its components, and where the modulus $|\cdot|$ of a vector is defined to be the vector consisting of the modulii of its components. The simple constraints (2) are essential to the solution algorithm presented in this paper, but the more general constraints (3) are optional.

It is assumed that $c > 0$ and $d > 0$. Zero components of c and d are equivalent to equality constraints of the form $zH = e$. If H has rank $q \leqq n$, then q of the unknowns can be solved for explicitly in terms of the remaining unknowns and substituted out of the problem. The reduced problem has $n - q$ unknowns and the same mathematical form as (1)-(3).

In this paper the problem (1)-(3) is replaced by a discretized problem. The discretized problem is a linear program which is generated in such a way that the

* Received by the editors July 5, 1983, and in revised form October 1, 1984. This work was supported by the Office of Naval Research Project RR014-07-01 and by the Independent Research Program of the Naval Underwater Systems Center. This paper was written during the author's stay as a visiting scholar in the Department of Operations Research, Stanford University, Stanford, CA 94305.

† Naval Underwater Systems Center, New London, Connecticut 06320.

relative error between its solution and a solution of the problem (1)-(3) can be estimated without knowing the solution of either. Furthermore, the maximum relative error can easily be made as small as desired by selecting an appropriate discretized problem. See Theorem 1 below.

The starting point for the discretized problem is the following simple observation. Let u be a complex number whose real and imaginary parts are u^R and u^I, respectively. It is easy to show that

$$|u| = \max_{0 \leqq \theta < 2\pi} (u^R \cos\theta + u^I \sin\theta). \tag{4}$$

Let p be a positive integer, and let $D = \{\theta_1, \cdots, \theta_p\}$ be a subset of the interval $[0, 2\pi)$. The discretized absolute value is defined by

$$|u|_D = \max_{\theta \in D} (u^R \cos\theta + u^I \sin\theta). \tag{5}$$

Although the set D can be arbitrary, it is convenient to assume that D consists of the pth roots of unity, that is,

$$\theta_k = (k-1)2\pi/p, \qquad k = 1, 2, \cdots, p, \tag{6}$$

and to assume that $p = 2^K$, $K \geqq 2$. It follows that

$$|u|_D \leqq |u| \leqq |u|_D \sec(\pi/p). \tag{7}$$

No other choice of D can give a tighter upper bound in (7). The requirement that p be a power of 2 facilitates computational efficiencies in solving the optimization problem (5) and is discussed in § 2. With these two assumptions, a relative accuracy of 5 significant digits in (7) requires that $p \geqq 1024$. Other properties of the discretized absolute value are given in [13].

The discretized version of (1)-(3) is developed by first transforming it into an optimization problem and then replacing all absolute values with discretized absolute values. The discretized problem can be written in the following manner.

Discretized problem.

$$\min_{\varepsilon \in R, z \in C^n} \varepsilon \tag{8}$$

subject to:

$$|zA_j - f_j|_D \leqq \varepsilon, \qquad j = 1, \cdots, m, \tag{9}$$

$$|zB_j - g_j|_D \leqq c_j, \qquad j = 1, \cdots, r, \tag{10}$$

$$|z_j - a_j|_D \leqq d_j, \qquad j = 1, \cdots, n, \tag{11}$$

where A_j and B_j denote the jth columns of the matrices A and B, respectively. It is shown in § 2 that the discretized problem is a linear program in $2n+1$ unknowns with $(m+n+r)p$ inequalities. This linear program cannot be assumed to be sparse since the matrices A and B are completely dense in many applications.

The discretized problem is most easily solved by solving its dual. The revised simplex method applied in a straightforward manner to the dual problem requires $O((m+n+r)np)$ storage locations and $O((m+r)np)$ multiplications per simplex iteration. It is shown in this paper that the factor of p can be eliminated from these estimates by successfully exploiting the special structure of the dual. These economies leave unaltered the sequence of basic feasible solutions (vertices) which the simplex method generates enroute to the solution of the dual. Thus the impact of the parameter p is limited to its effect on the total number of simplex iterations required to reach the solution. As will be seen, p affects the number of columns in the dual constraints

and not the number of rows, so the growth of total computational effort as a function of p is not great.

A Fortran program for solving the discretized problem has been written and documented [11]. This program does not implement all of the economies which are possible because of practical considerations discussed in § 2. The program as written requires $O((m+r)n)+O(p)$ storage locations and $O((m+r)n)+O((m+n+r)\log_2 p)$ multiplications per simplex iteration. Also, for reasons stated in § 3, the solution of the discretized problem for large values of p is approached via smaller values of p. The discretized problem for $p=4$ is first solved and its solution used as an advanced start for the $p=8$ discretized problem. The program continues doubling p at each stage until a specified value is attained. This program is practical for modest values of m, n, and r for large values of p.

The following theorem proves that a solution of the discretized problem is an approximate solution of the original problem. It also proves that the maximum relative error in this approximate solution can be made as small as desired by appropriate choice of p. Similar results for the unconstrained problem are given in [12], [13], and [9].

THEOREM 1. *Let $z^* \in C^n$ solve problem* (1)-(3), *and let $\varepsilon^{**} \in R$ and $z^{**} \in C^n$ solve the discretized problem* (8)-(11). *Then*

$$\varepsilon^{**} \leqq \|z^*A-f\|_\infty \leqq \|z^{**}A-f\|_\infty \leqq \varepsilon^{**}\sec(\pi/p) \tag{12}$$

$$|z^{**}B-g| \leqq c\sec(\pi/p), \tag{13}$$

$$|z^{**}-a| \leqq d\sec(\pi/p). \tag{14}$$

Proof. Since $|z_j^{**}-a_j|_D \leqq d_j$ for each j, it follows from (7) that

$$|z_j^{**}-a_j| \leqq |z_j^{**}-a_j|_D \sec(\pi/p) \leqq d_j \sec(\pi/p).$$

This proves (14), and (13) is proved the same way. The following sequence of inequalities establishes (12):

$$\begin{aligned}
\varepsilon^{**} &= \max|z^{**}A_j-f_j|_D \\
&\leqq \max|z^*A_j-f_j|_D \\
&\leqq \max|z^*A_j-f_j| = \|z^*A-f\|_\infty \\
&\leqq \max|z^{**}A_j-f_j| = \|z^{**}A-f\|_\infty \\
&\leqq \max|z^{**}A_j-f_j|_D \sec(\pi/p) \\
&= \varepsilon^{**}\sec(\pi/p)
\end{aligned}$$

where the max in all cases is over $j=1,\cdots,m$. This concludes the proof.

There is one hazard in replacing the original problem with the discretized problem. The constraints of the discretized problem have a larger feasible region than the original constraints, so it is possible that the discretized problem has solutions when the original problem is infeasible. The feasible region of the original problem is approximated more and more closely as p is increased, so the discretized problem ultimately fails to have a solution for sufficiently large p when the original problem is infeasible. If the original problem is in some sense "nearly" feasible, but in reality is infeasible, the discretized problem may possess solutions for very large values of p. Thus one may be deceived in certain problems. An alternative viewpoint is that any false solution obtained in this manner to infeasible problems actually represents a "reasonable"

solution to a poorly defined problem. Whether or not this view is sensible depends on the application. An example is given in § 5.

The problem (1)-(3) has a mathematically straightforward solution when all the quantities are real valued instead of complex. The real valued problem is exactly equivalent to a linear program in $n+1$ variables with $2(m+n+r)$ inequality constraints and can therefore be solved in a finite number of steps. The complex valued problem is less simple. Eliminating complex arithmetic by substituting in the real and imaginary parts of all complex quantities yields, after squaring the constraints, a mathematical programming problem in $2n+1$ variables having a linear objective function and $m+n+r$ quadratic constraints. No method is available for solving problems of this kind in a finite number of steps. Since it is a convex programming problem and the functions involved have easily obtained derivatives of all orders, many different algorithms are potentially applicable for its approximate solution. The only reference [14] known to the author which explicitly studies the constrained complex problem (1)-(3) uses a feasible directions method. At each step, a linear program is solved to determine the steepest feasible descent direction, a line search determines the step length, and special precautions are taken to prevent zigzagging, or jamming. A convergence proof is supplied.

The problem (1)-(3) can be viewed as a semiinfinite program (SIP). The SIP formulation of the unconstrained problem, that is, the problem consisting of only the objective function (1), has been studied elsewhere [12], [13], [4] in the context of complex function approximation and it is not difficult to extend that formulation to the constrained problem (1)-(3). None of these references, however, show that the special structure of the discretized problem can be used to significantly reduce the computational effort in its solution. Theorems 3, 4 and 5 of the next section are also new and are unique to the complex valued problem. The relationship between SIP and real valued approximation is presented in [3].

2. Solution of the discretized problem. An algorithm for solving the discretized problem for fixed p is discussed in this section. Attention is directed to special structures of the discretized problem which permit order of magnitude reductions in both storage requirements and multiplications per simplex iteration. Several useful theoretical results are interspersed.

It is first established that the discretized problem (8)-(11) is a linear program. Denote the real and imaginary parts of any quantity u by u^R and u^I, respectively, whether u be a number, a row or column vector, or a matrix. By definition (5),

$$|zA_j - f_j|_D = \max_{\theta \in D} [(zA_j - f_j)^R \cos\theta + (zA_j - f_j)^I \sin\theta], \tag{15}$$

so the m inequalities (9) are equivalent to the system of mp inequalities

$$(zA_j - f_j)^R \cos\theta + (zA_j - f_j)^I \sin\theta \leqq \varepsilon, \quad \theta \in D, \quad j = 1, \cdots, m. \tag{16}$$

Since

$$(zA_j - f_j)^R = z^R A_j^R - z^I A_j^I - f_j^R, \qquad (zA_j - f_j)^I = z^R A_j^I + z^I A_j^R - f_j^I,$$

it is convenient to write (16) in the form

$$[z^R z^I \varepsilon] \begin{bmatrix} A^R \cos\theta + A^I \sin\theta \\ A^R \sin\theta - A^I \cos\theta \\ -1_m \end{bmatrix} \leqq [f^R \cos\theta + f^I \sin\theta], \qquad \theta \in D, \tag{17}$$

where $1_m \in R^m$ is a row vector whose components all equal one. The inequalities (10)

and (11) are treated similarly, so the discretized problem is a linear program in $2n+1$ variables and $(m+n+r)p$ inequalities. The linear program can be written explicitly as follows.

Primal problem.

$$\min_{[z^R z^I \varepsilon]\in R^{2n+1}} [z^R \quad z^I \quad \varepsilon][0_n \quad 0_n \quad 1]^T \tag{18}$$

subject to: $\varepsilon \geqq 0$ and, for each $\theta \in D$,

$$[z^R \quad z^I \quad \varepsilon]\begin{bmatrix} A^R\cos\theta + A^I\sin\theta & B^R\cos\theta + B^I\sin\theta & I\cos\theta \\ A^R\sin\theta - A^I\cos\theta & B^R\sin\theta - B^I\cos\theta & I\sin\theta \\ -1_m & 0_r & 0_n \end{bmatrix} \tag{19}$$

$$\leqq [f^R\cos\theta + f^I\sin\theta \quad c + g^R\cos\theta + g^I\sin\theta \quad d + a^R\cos\theta + a^I\sin\theta],$$

where I denotes the $n\times n$ identity matrix and 0_k denotes a zero row (or column, depending on context) of length $k \geqq 1$.

The primal problem is solved by solving its dual using the revised simplex method. The simplex (Lagrange) multipliers for an optimal basic solution of the dual solve the primal, assuming the primal to be feasible. The dual can be written in one of the standard linear programming formats by explicitly adding a slack variable, denoted Q, which arises naturally in this problem.

Dual problem.

$$\min_{\substack{S\in R^{m\times p}, T\in R^{r\times p} \\ W\in R^{n\times p}, Q\in R}} \sum_{k=1}^{p} \begin{Bmatrix} (f^R\cos\theta_k + f^I\sin\theta_k)S_k \\ +(c + g^R\cos\theta_k + g^I\sin\theta_k)T_k \\ +(d + a^R\cos\theta_k + a^I\sin\theta_k)W_k \end{Bmatrix} \tag{20}$$

subject to: $S \geqq 0$, $T \geqq 0$, $W \geqq 0$, $Q \geqq 0$, and

(21)

$$\sum_{k=1}^{p} \begin{bmatrix} A^R\cos\theta_k + A^I\sin\theta_k & B^R\cos\theta_k + B^I\sin\theta_k & I\cos\theta_k \\ A^R\sin\theta_k - A^I\cos\theta_k & B^R\sin\theta_k - B^I\cos\theta_k & I\sin\theta_k \\ 1_m & 0_r & 0_n \end{bmatrix} \begin{bmatrix} S_k \\ T_k \\ W_k \end{bmatrix} + \begin{bmatrix} 0_n \\ 0_n \\ 1 \end{bmatrix} Q = \begin{bmatrix} 0_n \\ 0_n \\ 1 \end{bmatrix}.$$

An alternative statement of the dual is given at the end of this section.

The slack variable Q plays a special role, as seen in the next result.

THEOREM 2. *Let the matrices* $S^{**} \geqq 0$, $W^{**} \geqq 0$, $T^{**} \geqq 0$, *and the real number* $Q^{**} \geqq 0$ *denote an optimal basic feasible solution of the dual problem* (20)-(21). *If* $Q^{**} > 0$, *then the optimal value of the objective function in the primal problem* (18)-(19) *is zero.*

Proof. Let $[z^{**R} z^{**I} \varepsilon^{**}] \in R^{2n+1}$ denote the simplex multipliers of the optimal basic solution S^{**}, W^{**}, T^{**}, Q^{**}. Applying the complementary slackness theorem [8, p. 77], $Q^{**} > 0$ implies $\varepsilon^{**} = 0$ as claimed.

Except for the slack variable Q, every basic variable of the dual is uniquely identified by specifying the matrix to which it belongs together with its location (row and column number) in this matrix. The matrix names S, T, and W correspond to the inequality systems (9), (10), and (11), respectively. The row number of a basic variable identifies the particular constraint which gives rise to it. For example, all the dual variables in row q of matrix T are eliminated from the dual problem if the qth inequality in (10) is deleted from the discretized problem. Similarly, the column number of a basic variable identifies the angle in the set D to which it corresponds.

The revised simplex algorithm, as applied to the dual, is defined in general terms as follows:

Step 1. Determine an initial basic feasible solution of the dual problem.
Step 2. Compute the simplex multipliers corresponding to the current basic feasible solution.
Step 3. Determine the incoming variable by selecting the variable having the most negative reduced cost coefficient; terminate if all reduced cost coefficients are nonnegative—the primal problem is solved by the current simplex multipliers.
Step 4. Compute the column of the incoming variable in terms of the current basis.
Step 5. Determine the outgoing basic variable by a ratio test; terminate if the dual objective function is unbounded below—the primal problem is infeasible.
Step 6. Update the basis inverse and current basic feasible solution by pivoting, and return to Step 2.

The special structure of the dual problem has its strongest influence on Steps 1, 3, and 4. These effects are outlined next. More detailed aspects of the algorithm are postponed to § 4.

The dual problem is already in canonical form for initiating the second phase of the simplex algorithm. In other words, Step 1 is trivial because an identity matrix of order $2n+1$ can be assembled from the columns of the coefficient matrix of (21). One readily available column is the column corresponding to the slack variable Q. The remaining $2n$ columns correspond to dual variables which are the components of two particular W columns. From (6), $\theta_1 = 0$ so that $\cos\theta_1 = 1$ and $\sin\theta_1 = 0$. Hence one of the W columns can be taken to be W_1. Similarly, the other is $W_{1+p/4}$ since $\theta_{1+p/4} = \pi/2$. The initial basic feasible solution is therefore

$$W_1 = W_{1+p/4} = 0_m, \qquad Q = 1. \tag{22}$$

The simplex multipliers corresponding to (22) are derived in a special way later in this section.

The initial basic feasible solution (22) is highly degenerate. As discussed in [2], it is in problems of this general kind that cycling in the simplex algorithm is occasionally observed in practice. Such cycling was observed in an example given in this paper. However, a modification of the tie-breaking rule in the ratio test for the outgoing basic variable, together with "preferential treatment" of certain incoming variables, seems to avoid the difficulty. Further discussion of cycling in the dual problem is postponed to § 4.

The cost coefficients and the columns of any dual variable can be found by inspecting (20)-(21). They are given in a complex arithmetic format in Table 1. Explicitly computing and storing all $(m+n+r)p$ columns of the dual problem is unnecessary (and impractical) since the column of any dual variable can be constructed directly from the matrices A and B. Not counting the necessary sine and cosine, this requires only n complex multiplications and reduces the storage from $(2n+1)(m+n+r)p$ words to only $2n(m+n+r)$ words. The columns of the dual variables W_{jk} are merely columns of the identity matrix I, which need not be explicitly stored. Therefore the total storage necessary for constructing the column of any dual variable is only $2n(m+r)$ words. In practice, it is convenient to compute the cosines and sines once and for all to reduce the computational overhead. If this is done, as it is in [11], the storage requirements are $2n(m+r)+2p$.

TABLE 1
Dual variable cost coefficients and columns in complex format.

dual variable	cost coefficient	column, in R^{2n+1}
S_{jk}	$(f_j e^{-i\theta_k})^R$	$[(A_j e^{-i\theta_k})^R \quad -(A_j e^{-i\theta_k})^I \quad 1]^T$
T_{jk}	$(c_j + g_j e^{-i\theta_k})^R$	$[(B_j e^{-i\theta_k})^R \quad -(B_j e^{-i\theta_k})^I \quad 0]^T$
W_{jk}	$(d_j + a_j e^{-i\theta_k})^R$	$[(I_j e^{-i\theta_k})^R \quad -(I_j e^{-i\theta_k})^I \quad 0]^T$

An efficient method of computing the smallest reduced cost coefficient in Step 3 of the revised simplex algorithm is now discussed. This method is particularly interesting because the columns of the dual variables are not explicitly needed. The only data required are the original complex matrices A and B and the sines and cosines of the angles in D. Let λ be any real row vector of simplex multipliers for the dual problem; thus, λ is of length $2n+1$. The vector λ defines a complex row vector $z \in C^n$ and a real number ε by the identification

$$\lambda = [z^R \; z^I \; -\varepsilon] \in R^{2n+1}. \tag{23}$$

The reduced cost of the dual variable S_{jk} is the cost coefficient of S_{jk} minus the product of λ with the column of S_{jk}. Using (23) and Table 1 gives

$$\begin{aligned} C_S^{jk} &= (f_j e^{-i\theta_k})^R - [z^R \quad z^I \quad -\varepsilon][(A_j e^{-i\theta_k})^R \quad -(A_j e^{-i\theta_k})^I \quad 1]^T \\ &= \varepsilon - [(zA_j - f_j)\, e^{-i\theta_k}]^R, \end{aligned} \tag{24}$$

so the minimum reduced cost coefficient of the p variables in row j of S is

$$C_S^j = \min_{1 \le k \le p} C_S^{jk} = \varepsilon - |zA_j - f_j|_D, \qquad j = 1, 2, \cdots, m. \tag{25}$$

The smallest reduced cost coefficient of all the dual variables of S is then

$$C_S = \min_{1 \le j \le m} C_S^j = \varepsilon - \max_{1 \le j \le m} |zA_j - f_j|_D. \tag{26}$$

Similarly, the minimum reduced cost coefficients over all the dual variables of T and W are

$$C_T = \min_{1 \le j \le r} (c_j - |zB_j - g_j|_D) \tag{27}$$

and

$$C_W = \min_{1 \le j \le n} (d_j - |z_j - a_j|_D), \tag{28}$$

respectively. The smallest reduced cost of all the variables of the dual problem is $\min\{C_S, C_W, C_T\}$.

The smallest of the three quantities C_S, C_W, and C_T and the index j for which the minimum value is attained determine the row number and the correct matrix name of the incoming dual variable. The column number is determined by the angle $\theta_k \in D$ giving the largest projection (i.e., the discretized absolute value) at the minimal index j. The angle θ_k may not be unique because of possible ties in (5), so a tie-breaking rule called the minimal clockwise index (MCI) rule is used to determine unambiguously the incoming dual variable.

The MCI rule is defined for all $u \in C$. Let u_D be the set of those angles $\theta \in D$ for which the maximum in (5) is attained. There are three cases. First, if u_D has precisely one element, the MCI of u is defined to be the index of that element. Second, if u_D has precisely two elements, say θ_k and θ_j, and neither k or j equals p, then the MCI of u is defined to be $\min\{k, j\}$; on the other hand, if either $k = p$ or $j = p$, then the

MCI of u is taken to be p. Third, if u_D has more than two elements, then it must be that $u = 0$ and $u_D = D$, so the MCI of u is defined to be 1.

The computation of the discretized absolute value and corresponding MCI must be undertaken for $m+n+r$ complex numbers to compute (26)-(28) during each iteration of the simplex algorithm in Step 3. A brute force approach using the definition (5) requires $2p$ real multiplications for each complex number. Such an approach is inefficient and does not exploit the special form of the set D. For $p = 4$, it is clear that comparison tests alone suffice to solve this subproblem. For $p \geqq 8$, comparison tests and at most $2 \log_2 p - 5$ real multiplications are sufficient. To see this, first determine the quadrant of the complex plane in which the given number lies, and determine whether it lies above or below the 45° line bisecting the quadrant. This can be done using comparison tests only. Now that the "half-quadrant" in which the number lies is known, its projections onto the bounding rays of this half-quadrant can be computed in this special case using only one multiplication. If $p = 8$, a final comparison test ends the problem. If $p \geqq 16$, then the larger of the two projections reveals the "quarter-quadrant" in which the number must lie. The projection onto one of the bounding rays of this quarter-quadrant is already known; so it is only necessary to compute the projection onto the other bounding ray. This requires 2 real multiplications. If $p = 16$, a final comparison test ends the problem. If $p \geqq 32$, we continue as before. Counting the total possible number of steps proves the claim. This bisection method works because of the special form of the set D.

In principle the discretized absolute value and corresponding MCI can be found with computational effort independent of p. The argument (phase) of the given complex number can be computed, essentially as an inverse tangent, and from it the MCI can be found using comparison tests. Whenever the inverse tangent computation requires fewer than $2 \log_2 p - 5$ multiplications, it is more efficient than the bisection method described above. For $p \leqq 1024$ the bisection method is more efficient, and it is used in [11].

The number of real multiplications required to complete Step 3 using these methods is significantly less than that required in the usual approach. The straightforward method requires the computation of $(m+n+r)p - (2n+1)$ real inner products of length $2n$. Taking account of the simple form of the W columns gives a total of approximately

$$(2p-4)[n(m+r)+1]+4n(m+r-n-1/2) \tag{29}$$

real multiplications. The special methods discussed above require $m+n+r$ complex inner products of length n followed by the computation of the discretized absolute value and corresponding MCI for each inner product. Counting one complex multiplication as four real multiplications and considering the special form of the W columns gives a total of

$$4n(m+r)+(m+n+r)N_D \tag{30}$$

real multiplications, where N_D is the number of multiplications needed to compute one discretized absolute value and corresponding MCI. If the inverse tangent method is used, N_D is a constant independent of p. If the bisection method is used $N_D = 2 \log_2 p - 5$ for $p \geqq 8$, $N_D = 0$ for $p = 4$. The special methods are clearly better when $p \geqq 4$ and $m > n$. In the derivation of both (29) and (30) it was assumed that the last row of (21) in the dual problem was specially treated to avoid multiplications by 1 and 0.

The simplex multipliers $\lambda^{(0)} \in R^{2n+1}$ corresponding to the initial basic feasible solution (22) are now derived. Multiplying the initial basis inverse on the left by the

row vector containing the cost coefficients of the initial basic variables gives the row vector $\lambda^{(0)}$. The initial basis inverse is the identity matrix, the cost coefficients of the basic W variables (22) are given in Table 1, and the cost coefficient of the slack variable Q is 0. Consequently,

$$\lambda^{(0)} = [d+(a)^R \quad d+(-ia)^R \quad 0] = [d+a^R \quad d+a^I \quad 0] \in R^{2n+1}.$$

The definition (23) thus gives

$$z^{(0)} = a + d\, e^{i\pi/4}\sqrt{2}, \qquad \varepsilon^{(0)} = 0. \tag{31}$$

From the proof of Theorem 2 it can be seen that $\varepsilon = 0$ for as long as the slack variable remains in the basis and is positive.

The matrices S, W, and T are sparse because basic feasible solutions of the dual consist of only $2n+1$ nonnegative variables. Furthermore, no row of S, W, and T can contain more than two basic variables as the next theorem shows.

THEOREM 3. *No basic feasible solution of the dual problem* (20)-(21) *can have more than two basic variables in any one row of W or T. If a basic feasible solution of the dual problem has corresponding simplex multipliers with* $\varepsilon > 0$, *then S cannot have more than two basic variables in any one row.*

Proof. The first statement is proved for the matrix T; the proof for W is a special case. Consider the jth row of T. Suppose a basic feasible solution has three basic variables $T_{j\alpha}$, $T_{j\beta}$, and $T_{j\gamma}$ with α, β, and γ being distinct. Then the reduced costs for all three variables must be zero. A result analogous to (24) was used to prove (27); using it here gives

$$C_T^{jq} = 0 = c_j - [(zB_j - g_j)\, e^{-i\theta_q}]^R, \qquad q = \alpha, \beta, \gamma. \tag{32}$$

Thus the single complex number $zB_j - g_j$ has the same projection, namely c_j, in three distinct directions. This is impossible unless $zB_j - g_j = c_j = 0$, in contradiction to the assumption that $c_j > 0$. This establishes the first statement. The second statement is proved in the same way, by using (24) itself.

The following theorem relates knowledge of an optimal basis of the dual to "observable" quantities in the primal problem. The results of the theorem depend on the *names*, but not the actual numerical values, of the optimal dual basis. In addition it seems to indicate that the upper bound (12) in Theorem 1 will often be attained in practice.

THEOREM 4. *Let* $\varepsilon^{**} \in R$ *and* $z^{**} \in C^n$ *denote the simplex multipliers in the form* (23) *of an optimal basis for the dual problem* (20)-(21), *and suppose that* $\varepsilon^{**} > 0$. *If the jth row of one of the matrices S, W, or T contains two optimal basic variables in columns* α *and* β *with* $p \geqq \alpha > \beta \geqq 1$, *then either* $\alpha - \beta = 1$ *or* $\alpha - \beta = p - 1$. *If* $\alpha - \beta = 1$, *then*

$$z^{**}A_j - f_j = \varepsilon^{**} \sec(\pi/p) \exp[i(2\beta - 1)\pi/p], \tag{33}$$

or

$$z^{**}B_j - g_j = c_j \sec(\pi/p) \exp[i(2\beta - 1)\pi/p], \tag{34}$$

or

$$z_j^{**} - a_j = d_j \sec(\pi/p) \exp[i(2\beta - 1)\pi/p], \tag{35}$$

according to whether the jth row is a row of S, T, or W, respectively. Replacing β *with p in* (33)-(35) *gives the equations corresponding to the alternative case* $\alpha - \beta = p - 1$.

Proof. Only the S matrix case is treated since the other two cases are similar. The two basic variables involved are $S_{j\alpha}$ and $S_{j\beta}$. Assume that $p \geqq \alpha > \beta \geqq 1$. The reduced

costs $C_S^{j\alpha}$ and $C_S^{j\beta}$ must be 0, so (24) gives the two equations

$$\varepsilon^{**} = [(z^{**}A_j - f_j)\, e^{-i\theta_\alpha}]^R, \qquad \varepsilon^{**} = [(z^{**}A_j - f_j)\, e^{-i\theta_\beta}]^R. \tag{36}$$

Any complex number having identical projections in two directions is uniquely defined in both magnitude and phase. If θ_α differs from θ_β by π radians, the system (36) implies that $\varepsilon^{**} = 0$, contrary to assumption. Thus (36) implies that $z^{**}A_j - f_j = \varepsilon^{**} \sec(\phi/2) > 0$, where $\phi = \min\{\theta_\alpha - \theta_\beta, 2\pi - \theta_\alpha + \theta_\beta\}$. By Theorem 1, $\phi = \pi/p$, so that either $\theta_\alpha - \theta_\beta = \pi/p$ or $\theta_\alpha - \theta_\beta = \pi(2p-1)/p$. From (6), either $\alpha - \beta = 1$ or $\alpha - \beta = p - 1$. For $\alpha - \beta = 1$, solving the system (36) for the phase of $z^{**}A_j - f_j$ gives (33). The case $\alpha - \beta = p - 1$ is handled in the same way. This completes the proof.

Theorem 4 is useful in practice. Computed optimal dual solutions can be inspected to verify that optimal basic variables occurring in the same row are in fact "paired" in the manner described. If they are not, then numerical round-off errors have adversely affected the computed solution.

THEOREM 5. *Let ε^{**} and z^{**} be as in Theorem 4. If the jth row of one of the matrices S, W, or T contains an optimal basic variable in column α, $1 \leqq \alpha \leqq p$, then*

$$\varepsilon^{**} \leqq |z^{**}A_j - f_j| \leqq \varepsilon^{**} \sec \pi/p, \qquad \theta_\alpha - \pi/p \leqq \arg(z^{**}A_j - f_j) \leqq \theta_\alpha + \pi/p,$$

or

$$c_j \leqq |z^{**}B_j - g_j| \leqq c_j \sec \pi/p, \qquad \theta_\alpha - \pi/p \leqq \arg(z^{**}B_j - g_j) \leqq \theta_\alpha + \pi/p,$$

or

$$d_j \leqq |z_j^{**} - a_j| \leqq d_j \sec \pi/p, \qquad \theta_\alpha - \pi/p \leqq \arg(z_j^{**} - a_j) \leqq \theta_\alpha + \pi/p,$$

according to whether the jth row is a row of S, T, or W, respectively.

Proof. The proof is closely related to the method of proof of Theorem 4 and is not presented.

This section is concluded with a concise statement of the dual problem using complex arithmetic notation.

Dual problem: complex format.

$$\min_{\substack{S,T \\ W,Q}} [(fS + gT + aW)\, e^{-iD}]^R + \sum_{j=1}^{p} (CT_j + DW_j)$$

subject to: $S \geqq 0$, $T \geqq 0$, $W \geqq 0$, $Q \geqq 0$, and

$$(AS + BT + W)\, e^{-iD} = 0 \in C^n, \qquad Q + \sum_{j=1}^{m} \sum_{k=1}^{p} S_{jk} = 1.$$

We have used e^{-iD} to denote a complex column vector of length n whose kth component is $\exp(-i\theta_k)$; other notation is unchanged from (20)-(21).

3. Solution of the discretized problem for large p. One reason to solve large p discretized problems is that applications requiring 5 or more significant digits of relative accuracy in the optimal value of the objective function and/or in constraint satisfaction need to take $p \geqq 1024$; see Theorem 1. Another reason to solve large p problems is that their solutions furnish starting points for other methods which potentially provide greater accuracy. For instance, the problem (1)-(3) can be rewritten as a semiinfinite program, or SIP, and an interesting algorithm [5], [6] for solving a class of SIP's can be utilized. This method sets up an appropriate nonlinear system of algebraic equations which are solved using the Newton-Raphson method (or other iterative method); a

feasible solution of the nonlinear system is a solution of the SIP. The starting point of the Newton-Raphson iteration is taken to be the solution of a discretized problem. Large p discretized problems will have to be solved whenever very good starting points are needed to ensure convergence of the Newton-Raphson iteration.

There is, however, a practical limit to how large p may be taken in many problems. A discretized problem is numerically unstable for sufficiently large p if its optimal solution has, for every p, two basic dual variables in at least one row or S, W, or T. The columns of two such basic dual variables are less distinguishable numerically as p increases (see Table 1). Consequently, the basis matrix is more ill-conditioned for large p. Only those problems which never, for any p, have more than one optimal basic variable per row of S, W, or T can escape numerical ill-conditioning from this cause. Such problems seem to be uncommon.

The algorithm we suggest for solving the discretized problem for large p begins by solving the smallest dual problem, that is, the dual problem with $p=4$. Next, the $p=8$ dual problem is solved using the optimal basis for the $p=4$ dual to start the simplex algorithm. The $p=16$ dual is then solved starting at the optimal basis for the $p=8$ dual, and so forth. The algorithm is always well-defined because basic feasible dual solutions for a given p are also basic feasible dual solutions for all larger values of p because the sets D are nested for $p=4, 8, 16, 32, \cdots$. By doubling p at each stage beginning with $p=4$, this algorithm avoids bases associated with numerical instability from the discretization process until p becomes very large. Difficulties caused by ill-conditioning in the complex equations themselves cannot, of course, be avoided.

One advantage of this algorithm is that the optimal basis for each intermediate value of p can be easily inspected using Theorems 4 and 5 to determine if numerical round-off errors are significant. If sufficient error is present, the algorithm can be terminated early, or alternatively, the basis can be reinverted before continuing to the next value of p.

The primary drawback of the algorithm is that more simplex iterations are usually required to reach the final optimal dual basis by proceeding via smaller values of p than by solving the full dual problem all at once. This difficulty does not seem to be significant in practice and, in any event, can be partially overcome by skipping more rapidly through the available values of p. It is also possible to begin the algorithm with a larger initial value of p; that is, $p>4$.

Optimal solutions of the primal discretized problem converge only linearly with increasing p, while the optimal values ε^{**} converge quadratically. It would be useful to be able to extrapolate the primal solutions to obtain a better solution of the original problem (1)-(3). Richardson extrapolation (see, e.g., [7], [10]) worked very well for Examples 1-3 in § 5 for sufficiently large p, but failed in other problems. It is apparently successful only when (a) the row numbers of the optimal dual basic variables of the discretized problems identify the optimal active constraints of the original problem, and (b) the optimal values of the discretized problems equal the optimum value of the original problem. The first requirement can be met by taking p sufficiently large. The second requirement imposes more severe limitations on the practical utility of Richardson extrapolation.

4. Details of the revised simplex algorithm. Computer codes which treat complex matrices and vectors by separating them into their real and imaginary parts cause thrashing on virtual memory systems. Therefore the solution vector z of the primal problem is best stored as a complex vector and the simplex multipliers reordered to reflect the storage of z. The rows of the dual problem should also be reordered. The

computer code therefore visualizes the dual problem rows in the following order: $\{1, n+1, 2, n+2, \cdots, n-1, 2n-1, n, 2n, 2n+1\}$. These numbers denote the row numbers in the original system (21). The reordered system is much easier to work with in FORTRAN than the original system. With the rows of the dual problem in this order the reduced cost calculations can be coded in FORTRAN just as they are written in (26)-(28), provided the initial data of the problem are typed COMPLEX.

The name of a dual variable is a triplet $i/j/k$ of positive integers, where:

$i = 1, 2$, or 3 according to whether it is an S, W, or T variable,
$j =$ constraint number, from (9)-(11),
$k =$ projection number of the angle in the set D, $1 \leqq k \leqq p$.

The middle name j has different ranges depending on the value of the first name i. These triplets are ordered lexicographically.

The most negative reduced cost determines the entering basic variable in the simplex algorithm. Ties for the most negative reduced cost are broken by choosing the variable with the least lexicographically ordered name. Because the highly degenerate initial starting point (22) can cause cycling in the simplex algorithm, there is one exception to the least name rule in case of ties for the entering variable. As long as the slack variable Q remains in the basis, the only entering variables permitted are S variables with negative reduced costs. If S variables with negative reduced costs do not exist, then the entering variable is permitted to be a W or a T variable and ties are resolved by the least name rule. Thus, S variables are given priority for entering the basis only for as long as the slack Q is in the basis. Once Q is removed from the basis it never enters again, and exceptions to the tie breaking rule cease.

The outgoing basic variable is determined by the usual ratio test. If the least ratio is attained by more than one variable, the variable having the largest magnitude pivot leaves the basis. If more than one variable has the same magnitude pivot, then the variable with least index is selected. Because of degeneracy and cycling, there is one exception to this tie-breaking rule for the exiting variable. So long as the slack Q remains in the basis, only W variables are permitted to exit. This rule makes sense only when a W variable is involved in the tie; if no such W variable exists, the exception is not invoked. If more than one W variable is involved in the tie, then the one having the largest magnitude pivot with the least index is selected to exit. Just as for the entering variable, this exception ceases once the slack Q leaves the basis.

Cycling in the simplex algorithm has not been observed with these modifications to the usual tie breaking rules for entering and exiting variables. However, if these modifications are not used, cycling may well occur. Example 3 of § 5 below cycled (with a cycle of length 19) without these modifications. It is possible that cycling in this particular example is an artifact of finite precision arithmetic.

A nonzero tolerance is necessary when testing for the most negative reduced cost and for possible divisors in the ratio test. This number must not be too small and it must somehow be dependent on the scale of the problem data. The number used in [11] is the product of the unit round-off error of the host computer with the sum of the absolute values of the incoming column (i.e., its l_1 norm). This number is used for both reduced cost and pivot tolerance tests.

Besides the usual termination criteria in the simplex algorithm, the pricing method implicit in (26)-(28) yields a novel way to terminate the algorithm. The pricing method computes the most negative reduced cost by indirectly examining *all* reduced costs, not just the reduced costs of the nonbasic variables. Hence it can happen that the entering and the exiting variables are identical because of numerical round-off errors.

This event seems to signal that no further improvement in the solution is numerically possible. Solutions returned by terminating the algorithm whenever this "self-cycling" occurs appear to be satisfactory.

The Fortran code [11] was developed to test the methods described for solving the dual problem. It holds an explicit basis inverse and performs pivoting to update the inverse in each simplex iteration. Pivoting is known to be numerically unstable, but easily programmed. To forestall numerical difficulties the inverse is held in double precision, although a double precision inverse is not a satisfactory substitute for a numerically stable technique. Updating the QR factorization of the basis is preferable. Nonetheless the explicit inverse code gives good performance in many problems.

5. Examples. Example 1 is taken directly from [14, p. 249]. Let $n=2$, $m=5$, $r=2$, and define the matrices

$$A=\begin{bmatrix} 1 & 1 & 1 & .5 & 2 \\ -2 & 0 & 3 & -1 & 1 \end{bmatrix}, \qquad B=\begin{bmatrix} 2 & 2 \\ 2 & -4 \end{bmatrix}, \tag{37}$$

$$a=g=[0,0], \quad c=[\sqrt{2},\sqrt{2}], \quad d=[10,10],$$

$$f=[-1+i,\ -1+i,\ .5i,\ 0,\ -1+i].$$

Only the vector f is complex. The exact solution is $z_1=(-1+i)/2$, $z_2=0$, and $\varepsilon=\sqrt{2}/2$. The constraints of type (2) are not part of the original problem given in [14]. They have been added because their discretizations provide the initial dual basis.

Table 2 gives the solutions of the discretized primal problem for selected values of p. The optimal value of ε for $p=8$ is the optimal value of ε for all $p\geqq 8$. For $p\geqq 8$, the accuracy of the primal solutions depends solely upon the discretization errors since the optimal ε does not change. Table 3 gives the optimal basic solutions of the dual problems for the same values of p. The active constraints do not change for $p\geqq 8$, except for their θ names. Hence the active constraints at the optimum of the original nonlinear problem (1)-(3) have been identified. The fourth and fifth basic variables are "paired" in an obvious way; this behavior is explained by Theorem 4.

All optimal dual solutions are degenerate, or very nearly so. It turns out that the "degenerate parts" of the optimal dual solutions approximately doubled as p is doubled, especially when $p\geqq 64$. Assuming the trend continues indefinitely, the optimal dual solution will eventually look nondegenerate. This trend is probably an artifact of the numerical ill-conditioning inherent in the discretization process.

The conditions mentioned at the end of § 3 for success using Richardson extrapolation seem to be met for $p\geqq 8$. Since convergence of the z vectors is linear, multiply the $p=32$ vector by two and subtract the $p=16$ vector to get

$$[z_1^R,\ z_1^I,\ z_2^R,\ z_2^I]=[-.500964,\ .499036,\ .40\times 10^{-10},\ .36\times 10^{-10}].$$

One step of this extrapolation gives values nearly as accurate as the values corresponding to $p=2048$.

Numerical computations for this and the next two examples were performed on a DEC 10. It has a double precision unit round-off error of approximately 2×10^{-19}.

Example 1 can be made infeasible by adjoining one constraint of type (3). Replace B, g, and c in (37) with

$$\tilde{B}=\begin{bmatrix} 2 & 2 & 1 \\ 2 & -4 & 1 \end{bmatrix}, \qquad \tilde{g}=[0,0,7-4i], \qquad \tilde{c}=[\sqrt{2},\sqrt{2},\ 29/4].$$

The discretized primal problem is feasible for $p=4$ and 8; for $p\geqq 16$, it is infeasible.

TABLE 2
Solutions of the primal problem, Example 1.

p	4	8	16	32	2048
z_1^R	$-.588760$	$-.292893$	$-.400544$	$-.450754$	$-.499233$
z_1^I	.588760	.707107	.599456	.549246	.500767
z_2^R	$.59\times10^{-1}$	$-.82\times10^{-9}$	$-.28\times10^{-9}$	$-.12\times10^{-9}$	$-.15\times10^{-11}$
z_2^I	$-.59\times10^{-1}$	$-.82\times10^{-9}$	$-.12\times10^{-9}$	$-.78\times10^{-10}$	$-.15\times10^{-11}$
ε	.4112399	.7071068	.7071068	.7071068	.7071068
total iterations	10	14	17	20	38

TABLE 3
Solutions of the dual problem, Example 1.

p	4	8	16	32	2048
basis names	1/2/1	1/1/8	1/1/15	1/1/29	1/1/1793
	1/2/4	1/2/1	1/2/16	1/2/30	1/2/1794
	1/3/3	3/1/4	3/1/6	3/1/12	3/1/768
	3/2/2	3/2/3	3/2/6	3/2/12	3/2/768
	3/2/3	3/2/4	3/2/7	3/2/13	3/2/769
basis values	.714286	1.000000	1.000000	1.000000	1.000000
	.000000	$.50\times10^{-19}$	$.66\times10^{-19}$	$.11\times10^{-18}$	$.58\times10^{-17}$
	.285714	$-.50\times10^{-37}$	$.91\times10^{-19}$	$.24\times10^{-19}$	$.19\times10^{-17}$
	.000000	$.11\times10^{-18}$	$.22\times10^{-18}$	$.43\times10^{-18}$	$.27\times10^{-16}$
	.214286	.500000	.500000	.500000	.500000

This illustrates the remark made in § 1 that some discretized problems have feasible solutions when the original problem is actually infeasible.

Example 2 is the same as Example 1, except that constraints of type (2) are tightened so that they are active at the solution. Replace the vector d in (37) with $\tilde{d}=[.4, .4]$. The exact solution of this problem is $\varepsilon=\sqrt{2}-.4$, $z_1=(-1+i)\sqrt{2}/5$, and

$$z_2^I=\frac{317(\sqrt{2}-1)-(431902-190320\sqrt{2})^{1/2}}{300-1200\sqrt{2}}\cong -.208846903,$$

$$z_2^R=-\frac{\sqrt{2}}{10}+\left[\frac{1}{8}-\left(z_2^I-\frac{\sqrt{2}}{10}\right)^2\right]^{1/2}\cong -.093336568.$$

Tables 4 and 5 give, respectively, the solutions of the primal and dual discretized problem for selected values of p. The obvious "pair" of basic variables in Table 5 is explained by Theorem 4. The conditions for success using Richardson extrapolation seem to be met for $p\geqq 32$. Extrapolation of the $p=32$ and $p=64$ vectors in Table 4 performed as in Example 1 gives

$$[z_1^R, z_1^I, z_2^R, z_2^I]=[-.282776, .282911, -.092665, -.209334],$$

which is comparable to the values corresponding to $p=2048$.

Example 3 is taken from [12] and is an unconstrained complex function approximation problem; that is, constraints of type (2)-(3) are absent. The 101 columns of the

TABLE 4
Solutions of the primal problem, Example 2.

p	4	8	16	32	64	2048
z_1^R	−.400000	−.345442	−.293794	−.310700	−.296738	−.283277
z_1^I	.400000	.220244	.271891	.254985	.268948	.282409
z_2^R	.153553	−.026274	−.076571	−.061857	−.077261	−.092805
z_2^I	−.153553	−.243431	−.217608	−.214390	−.211862	−.208960
ε	.600000	1.014214	1.014214	1.014214	1.014214	1.014214
total iterations	7	11	13	16	18	33

TABLE 5
Solutions of the dual problem, Example 2.

p	4	8	16	32	64	2048
	1/2/1	1/2/8	1/2/15	1/2/29	1/2/57	1/2/1793
	1/2/4	1/3/6	1/3/12	1/3/22	1/3/43	1/3/1346
basis names	2/1/3	2/1/4	2/1/7	2/1/13	2/1/25	2/1/769
	3/2/2	3/2/3	3/2/5	2/1/14	2/1/26	2/1/770
	3/2/3	3/2/4	3/2/6	3/2/10	3/2/19	3/2/558
	1.	1.	1.	1.	1.	1.
	0.	0.	0.	0.	0.	0.
basis values	1.	1.	1.	1.	1.	1.
	0.	0.	0.	0.	0.	0.
	0.	0.	0.	0.	0.	0.

matrix $A \in C^{3\times 101}$ are

$$A_j = [1 \quad \exp(i(j-1)\pi/400) \quad \exp(i2(j-1)\pi/400)]^T, \qquad j = 1, 2, \cdots, 101,$$

while the components of $f \in C^{101}$ are $f_j = \exp(i3(j-1)\pi/400)$, $j = 1, 2, \cdots, 101$. In other words, the complex valued function e^{i3x} is approximated by complex linear combinations of the three functions 1, e^{ix}, and e^{i2x} over 101 equispaced points on the x-interval $[0, \pi/4]$. Bounds of type (2) must be specified, so we take $a = [0, 0, 0]$, $d = [10, 10, 10]$. These constraints are not active at the optimal solution.

It can be verified that the exact solution of Example 3 is $z_1 = \alpha_1 \exp(i3\pi/8)$, $z_2 = \alpha_2 \exp(i5\pi/4)$, $z_3 = \alpha_3 \exp(i\pi/8)$, where

$$\alpha_1 = a \cong .96157056080646,$$

$$\alpha_2 = b - 2(b - a^2)/(1 - a^2) \cong 2.8122548927058,$$

$$\alpha_3 = a(1 - 2b + a^2)/(1 - a^2) \cong 2.8477590650226,$$

$$a = \lambda \cos(\pi/16) + (1 - \lambda)\cos(\pi/8),$$

$$b = \lambda \cos(\pi/8) + (1 - \lambda)\cos(\pi/4),$$

$$c = \lambda \cos(3\pi/16) + (1 - \lambda)\cos(3\pi/8),$$

$$\lambda = \sin(\pi/8)/(\sin(\pi/16) + \sin(\pi/8)),$$

$$\varepsilon = (1 - c\alpha_1 + b\alpha_2 - a\alpha_3)^{1/2} \cong .014706309694449.$$

Tables 6 and 7 give, respectively, the solutions of the primal and dual discretized problems for selected values of p. The obvious "pairing" of the basic variables in Table 7 is explained by Theorem 4. Note also that the row numbers of the optimal dual basic variables are different for $p = 1024$ and $p = 64$ (probably because the dual does not have a unique solution). Nonetheless, Richardson extrapolation works when applied to the cases $p = 32$ and $p = 64$. As in the previous two examples, one extrapolation step gives

$$[z_1^R, z_1^I, z_2^R, z_2^I, z_3^R, z_3^I] = [.367954, .888319, -1.988481, -1.988481, 2.630930, 1.089767]$$

which, in turn, gives the values $\alpha_1 = .96151$, $\alpha_2 = 2.81214$, $\alpha_3 = 2.84770$. The case $p = 1024$ used directly gives the values $\alpha_1 = .96236$, $\alpha_2 = 2.81376$, $\alpha_3 = 2.84855$, which are clearly inferior to the extrapolated values.

TABLE 6
Solutions of the primal problem, Example 3.

p	8	16	32	64	1024
z_1^R	.378265	.377950	.377718	.372836	.368281
z_1^I	.913212	.912452	.911891	.900105	.889108
z_2^R	−2.026895	−2.024346	−2.022845	−2.005663	−1.989632
z_2^I	−2.026895	−2.024346	−2.022845	−2.005663	−1.989632
z_3^R	2.654494	2.654624	2.654502	2.642716	2.631719
z_3^I	1.099528	1.099581	1.099531	1.094649	1.090094
ε	.0141560	.0145244	.0147063	.0147063	.0147063
total iterations	20	25	33	36	48

TABLE 7
Solutions of the dual problem, Example 3.

p	8	16	32	64	1024
basis names	1/1/8	1/1/14	1/1/28	1/1/56	1/1/896
	1/25/4	1/1/15	1/1/29	1/1/57	1/1/897
	1/28/5	1/26/7	1/26/13	1/26/26	1/26/417
	1/74/8	1/27/8	1/26/14	1/26/27	1/76/993
	1/77/1	1/75/16	1/76/32	1/76/63	1/76/994
	1/101/5	1/76/1	1/101/17	1/101/33	1/101/513
	1/101/6	1/101/9	1/101/18	1/101/34	1/101/514
basis values	.163234	.004365	.000000	.000000	.000000
	.244029	.160548	.168829	.168829	.168829
	.091325	.170912	.000000	.000000	.331171
	.070677	.164573	.331171	.331171	.331170
	.263126	.173184	.331171	.331171	.000000
	.157491	.162060	.168829	.168829	.168829
	.010118	.164358	.000000	.000000	.000000

Another unconstrained complex function approximation problem in [12] is moderately large and completely dense. The motivating background and engineering application of this problem are fully discussed in [12]. The 501 columns of the matrix $A \in C^{44\times 501}$ are

$$A_j = [\exp(ik_1x_j) \quad \exp(ik_2x_j) \quad \cdots \quad \exp(ik_{44}x_j)]^T$$
$$-\exp(ik_{45}x_j)[1 \quad 1 \quad \cdots \quad 1]^T, \qquad j = 1, 2, \cdots, 501$$

where $1 = k_1 < k_2 < \cdots < k_{44} < k_{45} = 49$ are the distinct integers between 1 and 49, excluding the integers 7, 17, 21, and 29, and where $x_j = u_0 + (j-1)(1-u_0)/250$, $j = 1, 2, \cdots, 501$ with $u_0 = .0538117$. The components of $f \in C^{501}$ are $f_j = \exp(ik_{45}x_j)$, $j = 1, \cdots, 501$. This example lacks constraints of type (2)-(3). The discretized problem for $p = 16$ was solved on a DEC VAX 11/780 in 1350 simplex iterations. Total CPU time was 25 minutes and .7 million page faults were incurred. Only 80,000 words of storage were needed. In contrast, the algorithm proposed in [12] (which utilizes the algorithm [1] as a subroutine) solved this problem on the same VAX in 1270 simplex iterations, requiring 179 minutes of CPU time and incurring 11 million page faults. Over 360,000 words of storage were needed. The difference in the number of simplex iterations is explained as follows. The algorithm [12] solves the full problem for $p = 16$, while the algorithm developed in this paper solves the $p = 4$ problem and the $p = 8$ problem before solving the $p = 16$ problem. This indirect route to the full problem solution is less efficient in this example than solving the $p = 16$ problem immediately.

6. Concluding remarks. A solution of the discretized problem for sufficiently large p identifies the constraints active at a solution of the original problem (1)-(3). Deleting inactive constraints from the original problem yields an equality constrained nonlinear optimization problem. Lagrange's method gives rise to a nonlinear system of algebraic equations in the optimum value ε, the solution vector z, and the multipliers λ. Iterative methods for the solution of this system can be started from an initial point $(\varepsilon, z, \lambda)$ provided by a discretized problem solution. Safeguarded Newton-Raphson iteration may be highly effective for solving this system, especially if advantage is taken of the system's special form (i.e., for λ given, the vector z can be found by solving a system of linear equations). A possible limitation of this approach is that very large values of p might be necessary in order to identify the right active set. The examples of the previous section, however, indicate that the optimal active set is found for relatively small values of p. Specifically, in Examples 1, 2, and 3, the correct active sets (determined from the optimal dual basis names in Tables 3, 5, and 7) first appear when p is 8, 8, and 32, respectively.

Certain kinds of domain and range constraints can be adjoined to the discretized problem (8)-(11) with only minor extension of the algorithm proposed here. Let the matrix $H \in C^{n \times q}$, and the row vectors $e \in C^q$, $\phi \in R^q$, and $h \in R^q$ be given. Then the constraints

$$((zH_j - e_j)\exp(-i\phi_j))^R \leqq h_j, \qquad j = 1, \cdots, q \tag{38}$$

are linear in z^R and z^I, and so can be added to the discretized problem. The constraints (10) and (11) are instances of (38); however, (38) can impose constraints not possible with (10) and (11). For instance, if $q = 1$, the constraint that the complex number $zH_1 - e_1$ must lie in the right half complex plane is equivalent to $((zH_1 - e_1)\exp(-i\pi))^R \leqq 0$. Furthermore, if $q \geqq 1$ and the columns H and e are identical to their first columns, then the number $zH_1 - e_1$ can be confined to any closed convex polygonal region (bounded or unbounded) in the complex plane by appropriate choices of ϕ, h, and q.

When complex function approximation on an arc or domain boundary in the complex plane gives rise to the problem (1)-(3), then an implicit natural ordering of the columns of the matrix A exists. The ordering is inherited from the ordering of the discrete points along the arc, and it makes possible clever strategies of both multiple and partial pricing which may significantly reduce overall computation time when m and n are large. Effective partial pricing schemes require far fewer evaluations of the

vector-matrix products zA_j in (26) without significantly increasing the total number of iterations. Effective multiple pricing schemes decrease the number of iterations by increasing the change in ε in each iteration. Both multiple and partial pricing can be implemented simultaneously.

One particularly interesting problem is complex function approximation on the mth roots of unity. When $m \geqq n$ and when m is a power of 2, the fast Fourier transform (FFT) algorithm can be used to compute the m products zA_j in $2m \log_2 m$ operations. The straightforward products zA_j require mn operations. Therefore, the FFT method is more efficient whenever $2 \log_2 m \leqq n \leqq m$.

It has been assumed throughout this paper that the unknown vector z must lie in C^n. In some applications it is necessary to restrict z to R^n, while still retaining complex matrices A and B in original problem (1)-(3). Setting $z^I = 0$ in the discretized problem is equivalent to eliminating n of the $2n+1$ rows of the dual problem constraints (21). The techniques developed for the dual problem simplify when applied to this modified problem. Consequently the modified dual problem is smaller and easier to solve. Examples and a Fortran program for this problem are given in [11].

REFERENCES

[1] I. BARRODALE AND C. PHILLIPS, *Solution of an overdetermined system of linear equations in the Chebyshev norm*, ACM Algorithm 495, ACM Trans. Math. Software, 1 (1975), pp. 264-270.

[2] S. I. GASS, *Comments on the possibility of cycling with the simplex method*, Letter to The Editor, Oper. Res., 27 (1979), pp. 848-852.

[3] K. GLASHOFF AND S.-A. GUSTAFSON, *Linear Optimization and Approximation*, Springer-Verlag, New York, 1983.

[4] K. GLASHOFF AND K. ROLEFF, *A new method for Chebyshev approximation of complex-valued functions*, Math. Comp., 36 (1981), pp. 233-239.

[5] S.-A. GUSTAFSON, *Nonlinear systems in semi-infinite programming*, in Numerical Solution of Nonlinear Algebraic Systems, G. B. Byrnes and C. A. Hall, eds., Academic Press, New York, 1973, pp. 63-99.

[6] S.-A. GUSTAFSON AND K. KORTANEK, *Numerical treatment of a class of semiinfinite programming problems*, Naval Research Log. Quart., 20 (1973), pp. 477-504.

[7] D. C. JOYCE, *Survey of extrapolation processes in numerical analysis*, SIAM Rev., 13 (1971), pp. 435-488.

[8] D. G. LUENBERGER, *Introduction to Linear and Nonlinear Programming*, Addison-Wesley, Reading, MA, 1973.

[9] G. OPFER, *Solving complex approximation problems by semiinfinite-finite optimization techniques: a study on convergence*, Numer. Math., 39 (1982), pp. 411-420.

[10] A. RALSTON, *A First Course in Numerical Analysis*, McGraw-Hill, New York, 1965.

[11] R. L. STREIT, *An algorithm for the solution of systems of complex linear equations in the l_∞ norm with constraints on the unknowns*, ACM Trans. Math. Software, 11, 3 (1985).

[12] R. L. STREIT AND A. H. NUTTALL, *Linear Chebyshev complex function approximation and an application to beamforming*, J. Acoust. Soc. Amer., 72(1) (1982), pp. 181-190. (Also in Naval Underwater Systems Center Report 6403, 26 February 1981.)

[13] R. L. STREIT AND A. H. NUTTALL, *A note on the semi-infinite programming approach to complex approximation*, Math. Comp., 40(1983), pp. 599-605.

[14] S. I. ZUKHOVITSKIY AND L. I. AVDEYEVA, *Linear and Convex Programming*, W. B. Saunders, Philadelphia, 1966. (Original Russian edition, Moscow, 1964.)

SIAM J. SCI. STAT. COMPUT.
Vol. 7, No. 1, January 1986

011

EFFICIENT ALGORITHMS FOR COMPUTING THE CONDITION NUMBER OF A TRIDIAGONAL MATRIX*

NICHOLAS J. HIGHAM†

Abstract. Let A be a tridiagonal matrix of order n. We show that it is possible to compute $\|A^{-1}\|_\infty$, and hence $\text{cond}_\infty(A)$, in $O(n)$ operations. Several algorithms which perform this task are given and their numerical properties are investigated.

If A is also positive definite then $\|A^{-1}\|_\infty$ can be computed as the norm of the solution to a positive definite tridiagonal linear system whose coefficient matrix is closely related to A. We show how this computation can be carried out in parallel with the solution of a linear system $Ax = b$. In particular we describe some simple modifications to the LINPACK routine SPTSL which enable this routine to compute $\text{cond}_1(A)$, efficiently, in addition to solving $Ax = b$.

Key words. matrix condition number, tridiagonal matrix, positive definite matrix, LINPACK

1. Introduction.

Tridiagonal matrices

$$A = \begin{bmatrix} a_1 & c_1 & & & 0 \\ b_2 & a_2 & c_2 & & \\ & b_3 & a_3 & \ddots & \\ & & \ddots & \ddots & c_{n-1} \\ 0 & & & b_n & a_n \end{bmatrix} \in \mathbb{R}^{n\times n} \tag{1.1}$$

arise in many areas of numerical analysis, such as spline analysis [5, p. 133], difference methods for boundary value problems [9] and the numerical solution of linear second order recurrence relations [2, pp. 14 ff.]. Since the nonzero elements of A occur only within a band of width three, the cost of solving a tridiagonal system $Ax = b$ using Gaussian elimination with partial pivoting is $O(n)$ flops, as opposed to the $O(n^3)$ flops required when A is a full matrix [5, p. 166]. (See [15] for the definition of "flop".)

Let A be a square nonsingular matrix and $\|\cdot\|$ a matrix norm. When computing solutions of linear systems $Ax = b$ one would usually like to know the condition number of A,

$$\text{cond}(A) = \|A\|\,\|A^{-1}\|,$$

or at least an estimate, since this quantity measures the sensitivity of the true solution to perturbations in the data A and b, and features in various bounds for the norm of the error in the computed solution [17, pp. 192 ff.]. For the l_∞ matrix norm, given for $A = (a_{ij})$ by

$$\|A\|_\infty = \max_i \sum_j |a_{ij}|, \tag{1.2}$$

or the l_1 matrix norm, $\|A\|_1 = \|A^T\|_\infty$, $\|A\|$ is readily obtained whereas computation of $\|A^{-1}\|$ is more difficult.

Let A be a tridiagonal matrix of order n. One way of computing $\|A^{-1}\|_\infty$ is first to compute A^{-1}—using, say, Gaussian elimination with partial pivoting—and then to calculate the norm. However, this computation requires $O(n^2)$ flops, which, for large n, dominates the cost of solving $Ax = b$. We show that $\|A^{-1}\|_\infty$ may be computed in $O(n)$ flops; the methods used necessarily avoid explicit computation of the inverse.

* Received by the editors March 6, 1984. This work was carried out with the support of a SERC Research Studentship.

† Department of Mathematics, University of Manchester, Manchester M13 9PL, England.

The following definition is required.

DEFINITION. The tridiagonal matrix A in (1.1) is *irreducible* if $b_2, b_3, \cdots, b_n$ and $c_1, c_2, \cdots, c_{n-1}$ are all nonzero; otherwise it is *reducible.*

We remark that this definition is consistent with the more usual definition of irreducibility which applies to a general square matrix [16, pp. 102, 104].

The algorithms to be derived in § 4 apply to irreducible tridiagonal matrices. In § 7 we suggest a simple way of dealing with a tridiagonal matrix which is reducible.

Sections 5 and 6 are concerned with the numerical properties of our algorithms when implemented in floating-point arithmetic. The numerical stability of one of the algorithms is demonstrated with the aid of a backward error analysis.

For the case where A is positive definite we derive, in § 8, an alternative and more efficient way of computing $\|A^{-1}\|_\infty$. This method only requires the solution of one positive definite tridiagonal linear system. We show how the LINPACK routine SPTSL, which solves $Ax = b$ for positive definite tridiagonal matrices A, can be modified so that it also computes $\text{cond}_1(A)$, the latter computation proceeding in parallel with the solution of $Ax = b$.

The methods to be derived apply exclusively to the l_1 and l_∞ norms; we comment briefly on the l_2 norm condition number. The quantity

$$\phi = (\text{cond}_\infty(A)\,\text{cond}_1(A))^{1/2}$$

provides an order of magnitude estimate of $\text{cond}_2(A)$ since [19, p. 82]

$$\text{cond}_2(A) \leqq \phi \leqq n\,\text{cond}_2(A),$$

and ϕ may be computed in $O(n)$ flops when A is tridiagonal. A variety of alternative techniques are available for the estimation of $\text{cond}_2(A)$ when A is symmetric tridiagonal; among these are the well-known methods of inverse iteration, Sturm sequences, and bisection [20].

2. The inverse of a bidiagonal matrix. We begin by developing some properties of the inverse of a bidiagonal matrix B. These lead to an efficient algorithm for the computation of $\|B^{-1}\|_\infty$, and are also of use in § 8 since the LU factors of a tridiagonal matrix are bidiagonal, when they exist.

Consider the nonsingular upper bidiagonal matrix

$$U = \begin{bmatrix} u_1 & c_1 & & & 0 \\ & u_2 & c_2 & & \\ & & u_3 & \ddots & \\ & & & \ddots & c_{n-1} \\ 0 & & & & u_n \end{bmatrix} \in \mathbb{R}^{n\times n}, \quad u_i \neq 0, \quad 1 \leqq i \leqq n. \tag{2.1}$$

We find an explicit formula for the elements β_{ij} of U^{-1}. The jth column, $x_j = (\beta_{1j}, \beta_{2j}, \cdots, \beta_{nj})^T$, of U^{-1} satisfies $Ux_j = e_j$, where e_j is the jth unit vector. It follows that

$$\beta_{ij} = 0, \qquad i > j,$$

$$\beta_{jj} = \frac{1}{u_j},$$

$$\beta_{ij} = -\frac{c_i \beta_{i+1,j}}{u_i}, \qquad i < j.$$

Hence

$$\beta_{ij} = \begin{cases} 0, & i > j, \\ \dfrac{1}{u_j} \displaystyle\prod_{r=i}^{j-1} \left(\frac{-c_r}{u_r} \right), & i \leqq j, \end{cases} \tag{2.2}$$

where the empty product is defined to be 1. We observe from (2.2) that for all i and j $|\beta_{ij}|$ depends only on the moduli of the elements of U. In other words, the modulus of each element of U^{-1} is independent of the signs of the elements of U. It follows, using (1.2), that

$$|T| = |U| \text{ implies } \|T^{-1}\|_\infty = \|U^{-1}\|_\infty, \tag{2.3}$$

where, for $A = (a_{ij})$, $|A|$ denotes the matrix $(|a_{ij}|)$.

There is one particular distribution of the signs of the elements which yields a matrix for which the l_∞ norm of the inverse is easily calculated. To show this, we define for $A = (a_{ij})$ A's *comparison matrix* $M(A) = (m_{ij})$ by

$$m_{ij} = \begin{cases} |a_{ij}|, & i = j, \\ -|a_{ij}|, & i \neq j. \end{cases} \tag{2.4}$$

From (2.2) it is clear that $M(U)^{-1} \geqq 0$, that is $M(U)^{-1}$ has nonnegative elements (cf. [12]). We now make use of the observation that if $A^{-1} \geqq 0$ then $\|A^{-1}\|_\infty = \|A^{-1}e\|_\infty$, where $e = (1, 1, \cdots, 1)^T$. Together with (2.3) and the fact that $|M(A)| = |A|$ this yields

$$\|U^{-1}\|_\infty = \|M(U)^{-1}\|_\infty = \|M(U)^{-1}e\|_\infty. \tag{2.5}$$

These relations are also valid if U is lower bidiagonal.

Hence, in order to calculate $\|B^{-1}\|_\infty$ for a bidiagonal matrix B it is only necessary to solve one bidiagonal linear system and then to evaluate the l_∞ norm of the solution vector. We have the following algorithm, an analogue of which holds for a lower bidiagonal matrix.

ALGORITHM 2.1. Given the nonsingular upper bidiagonal matrix U in (2.1) this algorithm computes $\gamma = \|U^{-1}\|_\infty$.

$z_n := 1/|u_n|$; $\gamma := z_n$

For $i := n-1$ to 1 step -1

$z_i := (1 + |c_i| * z_{i+1})/|u_i|$

$\gamma := \max(\gamma, z_i)$.

Cost. $3n$ flops. (We count $\max(\cdot)$ as a flop.)

Thus Algorithm 2.1 requires $O(n)$ flops, a significant reduction on the $O(n^2)$ flops which would be required to compute U^{-1} and then calculate the norm.

3. The inverse of a tridiagonal matrix. Let A be a nonsingular tridiagonal matrix of order n. A can be represented in terms of $O(n)$ quantities: its nonzero elements. The next theorem shows that, as might be expected, A^{-1} is also representable in terms of $O(n)$ quantities—even though A^{-1} has in general $O(n^2)$ nonzero elements.

THEOREM 1. *Let the tridiagonal matrix A in* (1.1) *be nonsingular.*

(1) *If A is irreducible and $A^{-1} = (\alpha_{ij})$ then there are vectors x, y, p and q such that*

$$\alpha_{ij} = \begin{cases} x_i y_j, & i \leqq j, \\ p_i q_j, & i \geqq j. \end{cases} \tag{3.1}$$

(2) *If A is reducible then*

(a) *if $c_i = 0$, so that*

$$A = \begin{bmatrix} A_1 & 0 \\ B_1 & A_2 \end{bmatrix}, \tag{3.2}$$

where $A_1 \in \mathbb{R}^{i \times i}$ and $A_2 \in \mathbb{R}^{(n-i)\times(n-i)}$ are tridiagonal, then

$$A^{-1} = \begin{bmatrix} A_1^{-1} & 0 \\ X & A_2^{-1} \end{bmatrix}, \tag{3.3}$$

where $X \in \mathbb{R}^{(n-i)\times i}$ is a rank-one matrix if $b_{i+1} \neq 0$, or a zero matrix if $b_{i+1} = 0$, and the theorem applies recursively to A_1 and A_2;

(b) *if $b_{i+1} = 0$, part* (a) *applies to A^T.*

Proof. (1). See [14, Thm. 2].

(2). If $c_i = 0$ then A is clearly of the form (3.2) and $0 \neq \det(A) = \det(A_1)\det(A_2)$, so A_1 and A_2 are nonsingular. It is easily verified that $X = -A_2^{-1} B_1 A_1^{-1}$. B_1 has at most one nonzero element, b_{i+1}, in its $(1, i)$ position, so if $b_{i+1} = 0$, $X = 0$; otherwise B_1 is of rank one and hence X is of rank one. □

We remark that for the case $A = A^T$ part (1) of Theorem 1 is proved by Bukhberger and Emel'yanenko [1] and stated without proof by Graybill [10].

The vectors x and y (and similarly p and q) in (3.1) are easily seen to be unique up to a nonzero scale factor; any nonzero element of x or y can be assigned an arbitrary nonzero value (clearly $x_1 \neq 0$ and $y_n \neq 0$).

There is, in fact, some redundancy in the representation (3.1) of A^{-1}. For the four vectors x, y, p and q contain $4n-2$ "free" values, while A depends on only $3n-2$ numbers. This redundancy arises from the way in which part (1) of Theorem 1 is proved, namely by considering the upper triangular and lower triangular parts of A^{-1} separately. The following theorem provides a more concise representation of A^{-1}, in terms of $3n-2$ numbers.

THEOREM 2. *Let the tridiagonal matrix A in* (1.1) *be nonsingular and irreducible. Then there exist vectors x and y such that $A^{-1} = (\alpha_{ij})$ is given by*

$$\alpha_{ij} = \begin{cases} x_i y_j d_j, & i \leqq j, \\ y_i x_j d_j, & i \geqq j, \end{cases} \tag{3.4}$$

where

$$d_j = \prod_{r=1}^{j-1} \left(\frac{c_r}{b_{r+1}} \right), \qquad 1 \leqq j \leqq n.$$

Proof. Let $D = \operatorname{diag}(d_i)$. D exists and is nonsingular since A is irreducible. It is easily verified that $T = DA$ is tridiagonal, symmetric and irreducible. By applying Theorem 1 to T, and using the symmetry of T, we find that there are vectors x and y such that $T^{-1} = (\beta_{ij})$ is given by

$$\beta_{ij} = \begin{cases} x_i y_j, & i \leqq j, \\ y_i x_j, & i \geqq j. \end{cases}$$

From the relation $A^{-1} = T^{-1} D$ the result follows. □

Remark. Theorem 2, when combined with the method for computing x and y described in the next section, is essentially the same as [21, Thm. 2].

4. Algorithms for the irreducible case. Let the tridiagonal matrix A in (1.1) be nonsingular and irreducible. We now show, using the results of the previous section, that $\|A^{-1}\|_\infty$ can be computed in $O(n)$ flops.

Theorem 1 asserts the existence of vectors x, y, p and q such that the elements α_{ij} of A^{-1} are given by (3.1). Formation of each element α_{ij} of A^{-1} explicitly, using (3.1), requires $O(n^2)$ multiplications. However, in order to evaluate $\|A^{-1}\|_\infty$ we need only the row sums of A^{-1}. The ith row sum of A^{-1} is, from (3.1),

$$|p_iq_1|+|p_iq_2|+\cdots+|p_iq_{i-1}|+|x_iy_i|+|x_iy_{i+1}|+\cdots+|x_iy_n|,$$

which may be expediently rewritten as

$$|p_i|(|q_1|+|q_2|+\cdots+|q_{i-1}|)+|x_i|(|y_i|+|y_{i+1}|+\cdots+|y_n|).$$

Clearly, by accumulating the sums $t_i=|q_1|+\cdots+|q_i|$ and $s_i=|y_i|+\cdots+|y_n|$ the row sums of A^{-1} can be evaluated in $O(n)$ flops, given the vectors x, y, p and q. We now show how these vectors can be computed.

Following Ikebe [14] we equate the last columns in $AA^{-1}=I$ and the first rows in $A^{-1}A=I$, to obtain, using (3.1), $A(y_nx)=e_n$ and $(x_1y^T)A=e_1^T$, that is,

$$Ax=y_n^{-1}e_n, \tag{4.1}$$

$$A^Ty=x_1^{-1}e_1. \tag{4.2}$$

The one degree of freedom in the vectors x and y may be expended by setting $x_1=1$; then equations (4.1) and (4.2) may be solved for $x_2,\cdots,x_n,y_n,\cdots,y_1$ using the method of [14].

The vectors p and q are obtained in a similar way, using A^T in place of A. Thus we obtain the following algorithm.

ALGORITHM 4.1. Given the nonsingular $n\times n$ irreducible tridiagonal matrix A in (1.1) this algorithm computes $\gamma=\|A^{-1}\|_\infty$.

(1) $x_1:=1$; $x_2:=-a_1/c_1$

For $i:=3$ to n

$x_i:=-(a_{i-1}*x_{i-1}+b_{i-1}*x_{i-2})/c_{i-1}$

$y_n:=1/(b_n*x_{n-1}+a_n*x_n)$

$y_{n-1}:=-a_n*y_n/c_{n-1}$

For $i:=n-2$ to 1 step -1

$y_i:=-(a_{i+1}*y_{i+1}+b_{i+2}*y_{i+2})/c_i$.

(2) Repeat step (1) with x_i, y_i, b_i and c_i replaced by q_i, p_i, c_{i-1} and b_{i+1} respectively.

(3) $s_n:=|y_n|$

For $i:=n-1$ to 1 step -1

$s_i:=s_{i+1}+|y_i|$

$t_1:=1$

For $i := 2$ to n

$t_i := t_{i-1} + |q_i|$

$\gamma := \max(s_1, |p_n| * t_n)$

For $i := 2$ to $n-1$

$\gamma := \max(\gamma, |p_i| * t_{i-1} + |x_i| * s_i)$.

Cost. $17n$ flops.

An algorithm for the computation of $\|A^{-1}\|_\infty$ which is in general more efficient than Algorithm 4.1 can be derived from Theorem 2. Equating the last columns, and the first columns, in $AA^{-1} = I$ we find, using (3.4), that $A(y_n d_n)x = e_n$ and $A(x_1 d_1)y = e_1$. These equations may be rewritten, using $d_1 = 1$, as

$$Ax = (y_n d_n)^{-1} e_n, \tag{4.3}$$

$$Ay = x_1^{-1} e_1. \tag{4.4}$$

Choosing $x_1 = 1$, we can solve for x as in the previous algorithm and then determine y from (4.4), making use of the knowledge that $y_n \neq 0$.

Because the factor d_j is common to each element in the jth column of A^{-1} (see (3.4)) it is more convenient to evaluate the l_1 norm of A^{-1} than to evaluate the l_∞ norm.

ALGORITHM 4.2. Given the nonsingular $n \times n$ irreducible tridiagonal matrix A in (1.1) this algorithm computes $\gamma = \|A^{-1}\|_1$.

(1) $x_1 := 1$; $x_2 := -a_1/c_1$

For $i := 3$ to n

$x_i := -(a_{i-1} * x_{i-1} + b_{i-1} * x_{i-2})/c_{i-1}$.

(2) $z_n := 1$; $z_{n-1} := -a_n/b_n$

For $i := n-2$ to 1 step -1

$z_i := -(a_{i+1} * z_{i+1} + c_{i+1} * z_{i+2})/b_{i+1}$

$\theta := a_1 * z_1 + c_1 * z_2$.

(3) $s_n := |z_n|$

For $i := n-1$ to 1 step -1

$s_i := s_{i+1} + |z_i|$

$t_1 := 1$

For $i := 2$ to $n-1$

$t_i := t_{i-1} + |x_i|$

$d_1 := 1$; $\gamma := s_1$

For $j := 2$ to n

$d_j := d_{j-1} * c_{j-1}/b_j$

$\gamma := \max(\gamma, (|z_j| * t_{j-1} + |x_j| * s_j) * |d_j|)$

$\gamma := \gamma/|\theta|$.

Cost. $14n$ flops.

We now derive an algorithm for computing $\|A^{-1}\|_\infty$ which makes use of the LU factorisation of A, assuming this exists. The algorithm is based on the representation (3.1) and is obtained by choosing $x_1 = 1$ and rewriting (4.2) and (4.1) as

$$A^T y = e_1, \tag{4.5}$$

$$Az = e_n \qquad (z = y_n x). \tag{4.6}$$

ALGORITHM 4.3. Given a nonsingular $n \times n$ irreducible tridiagonal matrix A possessing a LU factorisation $A = LU$, this algorithm computes $\|A^{-1}\|_\infty$.

(1) Compute the LU factorisation of A.
(2) Use the LU factorisation to solve for the vectors y and z in (4.5) and (4.6). Similarly, solve for p and r, where $Ap = e_1$, $A^T r = e_n$ $(r = p_n q)$.
(3) Execute step (3) of Algorithm 4.1 with p, q and x replaced by $p_n^{-1}p$, r and $y_n^{-1}z$ respectively.

Cost. $18n$ flops.

We note that if Algorithms 4.1, 4.2 and 4.3 are adapted to take advantage of symmetry, then in each case the operation count is reduced to about $11n$ flops.

5. Computational considerations. Let the tridiagonal matrix A in (1.1) be nonsingular and irreducible, and consider the representation (3.1) of $A^{-1} = (\alpha_{ij})$ (the following applies to p and q in place of y and x if A is replaced by A^T).

Let $x_1 = 1$. Using the standard determinantal formula for the elements of the inverse [17, p. 402] one finds that

$$y_n = \alpha_{1n} = \frac{c_1 c_2 \cdots c_{n-1}}{\det(A)}.$$

From (3.1) the transpose of y is the first row of A^{-1} and x is the last column of A^{-1} scaled by y_n^{-1}. Clearly, the vectors x, y, p and q in (3.1) can be very badly scaled, in the sense that the nonzero elements of any particular vector can vary widely in order of magnitude. This is true whatever the choice of x_1, and is not related to the conditioning of A.

Examples. (1) For the $n \times n$ tridiagonal matrix A defined by $a_i = 4$, $b_i = c_{i-1} = 1$, we have $(x_1 = 1)|x_n| \approx \theta^{n-1}$, $|y_1| \approx \theta^{-1}$ and $|y_n| \approx \theta^{-n}$, where $\theta = 2 + \sqrt{3}$; $\operatorname{cond}(A) \leqq 3$.

(2)

$$A = \begin{bmatrix} 1 & \varepsilon \\ 1 & 1 \end{bmatrix}, \quad 0 < \varepsilon \ll 1, \qquad \|A^{-1}\|_\infty \approx 2.$$

Here $(x_1 = 1)$ $x_2 = -1/\varepsilon$, $y_1 = 1/(1-\varepsilon)$ and $y_2 = -\varepsilon/(1-\varepsilon)$.

We make two observations. First, there is a strong possibility of overflow and harmful underflow when Algorithm 4.1 is implemented on a computer. Second, it is not clear that in the presence of rounding errors the norm computed by Algorithm 4.1 will be of the correct order of magnitude (one does not usually want the condition number to many significant digits). For as Examples (1) and (2) illustrate we may have $\alpha_{ij} = x_i y_j = O(1)$ with $|x_i| \gg 1$ and $|y_j| \ll 1$; this is an ill-conditioned representation of α_{ij} in the sense that the product is very sensitive to absolute perturbations in y_j.

Similar comments apply to Algorithm 4.2 and the representation (3.4).

Note that the elements of the vectors y, z, p and r in Algorithm 4.3 are elements of A^{-1}, so they will not overflow on a computer as long as $\|A^{-1}\|_\infty$ is not too large. Step (3) of Algorithm 4.3 results in the divisions by the potentially very small quantities p_n and y_n being carried out at the last possible stage.

6. Error analysis. The observations of the last section lead us to investigate the accuracy of the norms computed by Algorithms 4.1, 4.2 and 4.3 in floating-point arithmetic and to consider how overflows and underflows can be avoided when the algorithms are implemented.

Consider Algorithm 4.3 implemented on a computer with t-digit base β floating-point arithmetic, and assume that the algorithm runs to completion. Make the usual assumption [20, p. 113] that

$$fl(x \text{ op } y) = (x \text{ op } y)(1+\delta), \qquad \text{op} = *, /, +, -,$$

for some $|\delta| \leqq \varepsilon$, where $\varepsilon = \frac{1}{2}\beta^{1-t}$ is the relative machine precision.

A backward error analysis in the style of de Boor and Pinkus [6] reveals that the computed LU factors from step (1) of Algorithm 4.3 satisfy (using a bar denote a computed quantity)

$$\bar{L}\bar{U} = A + E, \tag{6.1}$$

where E satisfies

$$|E| \leqq \mu_1 \varepsilon |\bar{L}|\,|\bar{U}|; \tag{6.2}$$

μ_i, $i = 1, 2, \cdots$, denotes a constant of order one. (The absence of a term involving n in this and subsequent bounds is due to the fact that A is tridiagonal.)

The standard backward error analysis for solution of a triangular system [17, p. 408] can be used to show that the computed vector $\bar{y}$ from step (2) of Algorithm 4.3 satisfies

$$(\bar{U} + \delta\bar{U})^T(\bar{L} + \delta\bar{L})^T\bar{y} = e_1,$$

where

$$|\delta\bar{U}| \leqq \mu_2\varepsilon|\bar{U}| \quad \text{and} \quad |\delta\bar{L}| \leqq \mu_3\varepsilon|\bar{L}|. \tag{6.3}$$

It follows that $\bar{y}$ is the true solution of

$$W^T\hat{y} = e_1, \tag{6.4}$$

where

$$W = A + F = (\bar{L} + \delta\bar{L})(\bar{U} + \delta\bar{U}).$$

Combining (6.1), (6.2) and (6.3) we have, writing $\mu = \max(\mu_1, \mu_2, \mu_3)$,

$$|F| \leqq (3\mu\varepsilon + \mu^2\varepsilon^2)|\bar{L}|\,|\bar{U}|,$$

from which, using (6.1) and (6.2), it can be deduced that W is tridiagonal and irreducible.

Let $\hat{z}$ satisfy

$$W\hat{z} = e_n. \tag{6.5}$$

By comparing the back substitutions which determine $\hat{z}$ and the computed vector $\bar{z}$ from step (2) of Algorithm 4.3, one finds that

$$\bar{z} = (1+\rho)\hat{z}, \qquad |\rho| \leqq \mu_4 n\varepsilon. \tag{6.6}$$

Thus we have shown that the vectors $\bar{y}$ and $\bar{z}$ computed by Algorithm 4.3 in floating-point arithmetic are, respectively, $\hat{y}$, and an approximation to $\hat{z}$ with small componentwise relative error, where $\hat{y}$ and $\hat{z}$ are the true vectors in (4.5) and (4.6) corresponding to using the matrix $W = A + F$ in place of A.

Now assume that

$$\|F\|_\infty \approx \varepsilon \|A\|_\infty \tag{6.7}$$

and

$$\operatorname{cond}_\infty(A)\varepsilon \ll 1. \tag{6.8}$$

Then, using the result [17, p. 189]

$$\frac{\|A^{-1}-(A+E)^{-1}\|}{\|A^{-1}\|} \leqq \frac{\phi}{1-\phi}, \qquad \phi = \operatorname{cond}(A)\frac{\|E\|}{\|A\|} < 1,$$

we have, approximately,

$$\|A^{-1} - W^{-1}\|_\infty \leqq \operatorname{cond}_\infty(A)\varepsilon \|A^{-1}\|_\infty. \tag{6.9}$$

From (6.4), (6.5) and (6.6) it follows that the "upper triangular parts" $\bar{R}_i$ of the row sums computed in step (3) of Algorithm 4.3 (that is, $\bar{R}_i$ approximates $|\alpha_{ii}|+\cdots+|\alpha_{in}|$, where $A^{-1}=(\alpha_{ij})$) satisfy

$$\bar{R}_i = \left(\sum_{j \geqq i} |\beta_{ij}|\right)(1+O(n\varepsilon)),$$

where $W^{-1}=(\beta_{ij})$. Hence

$$\left|\sum_{j\geqq i}|\alpha_{ij}| - \bar{R}_i\right| \approx \left|\sum_{j\geqq i}(|\alpha_{ij}|-|\beta_{ij}|)\right|$$

$$\leqq \sum_{j\geqq i}|\alpha_{ij}-\beta_{ij}| \leqq \operatorname{cond}_\infty(A)\varepsilon\|A^{-1}\|_\infty,$$

using (6.9). A similar result holds for the remaining parts of the computed row sums.

Thus we conclude that if (6.7) and (6.8) are satisfied, then the number $\bar{\gamma}$ computed by Algorithm 4.3 in floating-point arithmetic satisfies, approximately,

$$\frac{|\bar{\gamma} - \|A^{-1}\|_\infty|}{\|A^{-1}\|_\infty} \leqq 2\operatorname{cond}_\infty(A)\varepsilon. \tag{6.10}$$

This is just about the best that can be expected, since it can be shown that the condition number for the problem of computing $\|A^{-1}\|_\infty$ is $\operatorname{cond}_\infty(A)$.

Algorithm 4.3 breaks down in step (3) if the computed quantity $\bar{y}_n$ is zero. It is easily checked that this can happen only if $\bar{y}_n$, or some intermediate quantity, underflows to zero (cancellation cannot take place). Unfortunately, underflow can occur even for quite well-behaved matrices, as indicated by the first example in § 5. The possibility of underflow (and overflow) may be avoided by representing all the numbers which arise in Algorithm 4.3 by a pair $(d, e) \equiv d \times b^e$, where $1 \leqq d < b$ (say), e is an integer, and b is some base, preferably a power of the machine base β. (This idea can also be applied to Algorithms 4.1 and 4.2.) Then, for example, the computation $z = x * y$ becomes

$$z = (d_x * d_y) \times b^{e_x+e_y} = d_z \times b^{e_z},$$

where d_z is kept in the desired range through a suitable scaling by a power of b together with an adjustment of the exponent e_z. This technique, with $b = 10$, is used in LINPACK in all the code which evaluates determinants.

We have been unable to prove a result such as (6.10) for Algorithms 4.1 and 4.2. When Algorithm 4.2 for example is executed in floating-point arithmetic, steps (1) and

(2) can be interpreted as being exact for perturbed matrices $A+E_1$ and $A+E_2$ with $|E_i| \leqq \mu_i \varepsilon |A|$, $i=1,2$, but it is difficult to assess the effect of $E_1 \neq E_2$, which will be true in general, on the accuracy of the row sums computed in step (3). Note that a forward error analysis is not helpful because the coefficient matrices of the triangular systems which are solved in Algorithms 4.1 and 4.2 can be arbitrarily ill-conditioned—their diagonal elements are b_i's or c_i's.

However, our experience in using Algorithms 4.1 and 4.2 is that they both produce extremely accurate results, whatever the condition number of A. In fact, we have not come across an instance in which the norms computed by Algorithms 4.1, 4.2 and 4.3 differed from each other in more than the last two significant digits—even when the computed vectors had elements covering the whole spectrum of machine numbers: from underflow level to overflow level. The machine used here was a Commodore 4032 microcomputer with $\beta=2$, $t=32$ and exponent range $-128 \leqq e \leqq 127$.

We feel that this unexpectedly high accuracy of the computed norms is due to the phenomenon observed by Wilkinson [19, pp. 103 ff.], [20, pp. 249 ff.] (see also [17, p. 150]), whereby the accuracy of the computed solution to a triangular system is commonly independent of the condition number of the coefficient matrix.

Condition (6.7) relates to the stability of Gaussian elimination without pivoting and will certainly be satisfied if A is diagonally dominant; this is often the case in recurrence relation applications [3]. To avoid both large element growth and the possibility of Gaussian elimination breaking down one could use partial pivoting in Algorithm 4.3, obtaining $PA=LU$, with a guaranteed bound of 2 for the "growth factor" [17, p. 158]. Note, however, that pivoting changes the form of the triangular factors (L is now lower triangular with at most two nonzero elements per column and U is upper triangular with $u_{ij}=0$ for $j>i+2$), resulting in a possible extra n flops for both the factorisation stage and each substitution; and our proof of the result (6.10) does not apply in this case.

Note that everything we have said concerning the properties and computation of the vectors in (3.1) and (3.4) applies perforce to the methods for inverting irreducible tridiagonal matrices which have been proposed in [1], [14], [21].

7. Dealing with reducibility. For particular classes of matrix it is possible to verify irreducibility in advance. Difference methods for boundary value problems can lead to tridiagonal matrices with off-diagonal elements of the form $\alpha+O(h)$, where $\alpha>0$ and h is the mesh length [16, pp. 96 ff.]; here, irreducibility is assured for sufficiently small h. The tridiagonal matrices which arise in the numerical solution of linear second order recurrence relations can be assumed, without loss of generality, to be irreducible [3]. In some situations, however, one will be unable to guarantee irreducibility. In this section we describe how to deal with the case where A is reducible.

Let A be an $n \times n$ nonsingular reducible tridiagonal matrix. If A is symmetric then A has the block form diag $(A_1, A_2, \cdots, A_k)$, where each matrix A_i is nonsingular and either diagonal, or irreducible and tridiagonal. Hence $\|A^{-1}\|_\infty = \max_i \|A_i^{-1}\|_\infty$, which can be computed in $O(n)$ flops by applying Algorithm 4.3 (say) to each of the matrices A_i, as appropriate.

If A is nonsymmetric then we suggest the following approach. Let E be the tridiagonal matrix whose diagonal elements are zero, and whose off-diagonal elements are zero or $\delta/2$ according as the corresponding elements of A are nonzero or zero; here, δ is a small nonzero number. The matrix $G=A+E$ is irreducible and tridiagonal. If

$$\delta \|A^{-1}\|_\infty \leqq 0.1 \tag{7.1}$$

then [18, p. 293]

$$\tfrac{8}{9}\|A^{-1}\|_\infty \leqq \|G^{-1}\|_\infty \leqq \tfrac{10}{9}\|A^{-1}\|_\infty.$$

Thus, by choosing δ small enough a satisfactory approximation to $\|A^{-1}\|_\infty$ can be computed in $O(n)$ flops, by applying one of the algorithms for the irreducible case to G. A suitable choice for δ is a small multiple of the relative machine precision ε. For such a δ, (7.1) will be satisfied unless A is very nearly singular to working precision (assuming $\|A\|_\infty \approx 1$). We note that the poor scaling discussed in § 5 is quite likely to be associated with G if A has many zero elements on its subdiagonal and superdiagonal.

An alternative method for dealing with the case where A is nonsymmetric and reducible, leading to computation of $\|A^{-1}\|_\infty$ rather than an approximation, is described in [13]. The method is motivated by part (2) of Theorem 1 and regards A as a block tridiagonal matrix, where the diagonal blocks are tridiagonal and irreducible. The computational cost is again $O(n)$ flops but the method is rather more difficult to implement than the above technique.

8. Positive definiteness.

8.1. Let

$$A = \begin{bmatrix} a_1 & b_2 & & & 0 \\ b_2 & a_2 & b_3 & & \\ & b_3 & a_3 & \ddots & \\ & & \ddots & \ddots & b_n \\ 0 & & & b_n & a_n \end{bmatrix} \in \mathbb{R}^{n\times n} \tag{8.1}$$

be a positive definite tridiagonal matrix with, necessarily,

$$a_i > 0, \qquad 1 \leqq i \leqq n. \tag{8.2}$$

In this section we derive a method for computing $\|A^{-1}\|_\infty$ which is more efficient than the "symmetric" versions of Algorithms 4.1, 4.2 and 4.3 and which does not require A to be irreducible.

We begin by showing that there is a matrix $D = \operatorname{diag}(d_i)$, $d_i = \pm 1$, such that

$$M(A) = DAD, \tag{8.3}$$

where $M(A)$ is the comparison matrix defined in (2.4). Writing $W = DAD$, we have

$$w_{ii} = a_i, \qquad 1 \leqq i \leqq n$$

and

$$w_{i,i+1} = d_i b_{i+1} d_{i+1}, \qquad 1 \leqq i \leqq n-1.$$

Let $d_1 = 1$ and $d_{i+1} = -\operatorname{sgn}(d_i b_{i+1})$, $1 \leqq i \leqq n-1$, where

$$\operatorname{sgn}(x) = \begin{cases} 1, & x \geqq 0, \\ -1, & x < 0. \end{cases}$$

W is evidently tridiagonal and symmetric, $w_{ii} = a_i = |a_i|$ (using (8.2)), $|w_{i,i+1}| = |b_{i+1}|$ and $w_{i,i+1} \leqq 0$. Hence $W = M(A)$, which establishes (8.3).

A is positive definite and hence has a Choleski factorisation

$$A = LL^T, \tag{8.4}$$

where

$$L=\begin{bmatrix} l_1 & & & & 0 \\ m_2 & l_2 & & & \\ & m_3 & l_3 & & \\ & & \ddots & \ddots & \\ 0 & & & m_n & l_n \end{bmatrix}$$

with $l_i > 0$ for all i. We claim that

$$M(A) = M(L)M(L)^T. \tag{8.5}$$

This is proved by noting, first, that (8.4) implies $a_i = m_i^2 + l_i^2$ and $b_{i+1} = l_i m_{i+1}$. Then, writing $H = M(L)M(L)^T$,

$$h_{ii} = (-|m_i|)(-|m_i|) + l_i^2 = a_i = |a_i|$$

and

$$h_{i,i+1} = l_i(-|m_{i+1}|) = -|b_{i+1}|.$$

Hence $H = M(A)$ as required.

Note that $M(A)$ is positive definite, by (8.5). In fact, the positive definiteness of A is independent of the signs of the off-diagonal elements $\{b_i\}$.

From § 2, $M(L)^{-1} \geqq 0$, therefore

$$M(A)^{-1} = M(L)^{-T}M(L)^{-1} \geqq 0. \tag{8.6}$$

The final result we require is

$$M(A)^{-1} = |A^{-1}|, \tag{8.7}$$

which is a consequence of (8.3).

It follows from (8.6) and (8.7) that if A is a positive definite tridiagonal matrix,

$$\|A^{-1}\|_\infty = \| \, |A^{-1}| \, \|_\infty = \|M(A)^{-1}\|_\infty = \|M(A)^{-1}e\|_\infty.$$

As in the bidiagonal matrix case, computation of the l_∞ norm of the inverse reduces to solution of a linear system involving the comparison matrix. Thus we have

ALGORITHM 8.1. Given an $n \times n$ positive definite tridiagonal matrix A this algorithm computes $\gamma = \|A^{-1}\|_\infty$.

Solve $M(A)z = e$, evaluating $\gamma := \max_{1 \leqq i \leqq n} z_i = \|z\|_\infty$.

Cost. $6n$ flops.

The operation count given for Algorithm 8.1 assumes the use of the LDL^T factorisation of $M(A)$. The closely related Choleski factorisation is unattractive in this context because it requires n square roots, and an extra n divisions in the substitution stage. The LDL^T factors of the matrix A in (8.1),

$$L=\begin{bmatrix} 1 & & & & 0 \\ l_2 & 1 & & & \\ & l_3 & 1 & & \\ & & \ddots & \ddots & \\ 0 & & & l_n & 1 \end{bmatrix}, \qquad D = \operatorname{diag}(d_i),$$

are generated by

$$d_1 = a_1,$$

$$l_i = \frac{b_i}{d_{i-1}}, \qquad d_i = a_i - l_i b_i, \qquad 2 \leqq i \leqq n.$$

8.2. The nesting technique. LINPACK [7] has a routine SPTSL which solves $Ax = b$ for positive definite tridiagonal matrices A. We shall show that it is possible to "nest" Algorithm 8.1 inside the routine SPTSL in such a way that the composite routine computes $\operatorname{cond}_\infty(A)$ in addition to solving $Ax = b$, and is computationally more efficient than separate applications of SPTSL and Algorithm 8.1. Because SPTSL uses a nonstandard factorisation method which is more complicated than the LDL^T factorisation, we first derive the nesting technique for the LDL^T method.

An important feature which Algorithm 8.1 does not exploit is that the LDL^T factorisations of A and $M(A)$ are related: by comparison with (8.4) and (8.5), if $A = LDL^T$ then $M(A) = M(L)DM(L)^T$. Therefore, solution of $M(A)z = e$ can be accomplished using the LDL^T factorisation of A, which has to be computed anyway in the course of solving $Ax = b$; there is no need to explicitly factorise $M(A)$. The next algorithm makes use of this observation, thereby saving $2n$ flops.

ALGORITHM 8.2. Given the $n \times n$ positive definite tridiagonal matrix A in (8.1) and the vector f, this algorithm computes both the solution to the linear system $Ax = f$ and $\gamma = \|A^{-1}\|_\infty$. The solution overwrites f.

The statements marked with an asterisk are those which have been added to the basic routine for $Ax = f$ in order to compute $\|A^{-1}\|_\infty$.

```
(1)       d_1 := a_1
     (*)  z_1 := 1
          For i := 2 to n
               l_i := b_i / d_{i-1}
               d_i := a_i - l_i * b_i
               f_i := f_i - l_i * f_{i-1}
     (*)       z_i := 1 + |l_i| * z_{i-1}.
(2)       f_n := f_n / d_n
     (*)  z_n := z_n / d_n;  γ := z_n
          For i := n - 1 to 1 step -1
               f_i := f_i / d_i - l_{i+1} * f_{i+1}
     (*)       z_i := z_i / d_i + |l_{i+1}| * z_{i+1}
     (*)       γ := max(γ, z_i).
```

Cost. $9n$ flops.

In Algorithm 8.2 computation of $\|A^{-1}\|_\infty$ adds only $4n$ flops to the cost of solving $Ax = f$. Furthermore, computation of $\|A^{-1}\|_\infty$ does not introduce any loops over and above those required for solution of the linear system. This is an important feature, since typically the loop overheads may account for a significant portion of the machine execution time of Algorithm 8.2 [8]. We conclude that on most computers the execution

time for a routine based on Algorithm 8.2 should be less than 80% greater than that of an equivalent routine which only solves $Ax = f$.

We now show how Algorithm 8.1 can be nested within the LINPACK routine SPTSL. First, we describe briefly the nonstandard reduction technique which this routine uses; for full details see [7]. The reduction consists of a form of Gaussian elimination without pivoting in which subdiagonal elements are eliminated using row operations working from the top, and, simultaneously, superdiagonal elements are eliminated using row operations working from the bottom. Thus zeros are introduced to the elements in positions $(2,1)$, $(n-1,n)$, $(3,2)$, $(n-2,n-1)$, $\cdots$, in this order, until the two eliminations meet in the middle. The reduced system is such that the unknowns can be determined by a simple substitution process, which works from the middle to the top and bottom of the matrix simultaneously.

The algorithm used in SPTSL is known as the "burn at both ends" (BABE) algorithm. The motivation for the BABE algorithm is that each of its two loops (one for the reduction phase and one for the substitution phase) is executed only half as many times as the corresponding loop in a standard algorithm (since two eliminations or two substitutions are performed on each run through a loop), so that the loop overhead is reduced by a factor of two (cf. [8]). The LINPACK manual claims that the BABE algorithm can solve positive definite tridiagonal linear systems up to 25% faster than conventional algorithms.

It can be shown (see [13] for example) that the BABE algorithm corresponds to the factorisation

$$A = LUB, \tag{8.8}$$

where L is unit lower bidiagonal, U is unit upper bidiagonal, and the nonzero elements of B lie on the diagonal and in positions $(1,2)$, $(2,3)$, $\cdots$, $(k,k+1)$, $(k+2,k+1)$, $(k+3,k+2)$, $\cdots$, $(n,n-1)$, where $k=[n/2]$ ($[x]$ denotes the integer part of x).

The following lemma is the basis for the application of the nesting technique to the BABE algorithm.

LEMMA. *If the positive definite tridiagonal matrix A is factored according to* (8.8), *then*

$$M(A) = M(L)M(U)M(B).$$

Proof. The proof proceeds by comparing, elementwise, the reduction phases of the BABE algorithm applied to A and to $M(A)$. For further details see [13]. □

The lemma shows that the equation

$$M(A)z = e \tag{8.9}$$

can be solved using information gleaned during the solution of $Ax = b$ by the BABE algorithm, namely, the elements of L, U and B.

The modifications to the LINPACK routine SPTSL which are required in order for this routine to compute $\mathrm{cond}_1(A)$ (the same as $\mathrm{cond}_\infty(A)$ since A is symmetric) in addition to solving $Ax = b$ consist simply of extra statements, some for the evaluation of $\|A\|_1$ and some analogous to those marked with an asterisk in Algorithm 8.2. Two extra parameters are required by the modified routine: a work vector Z of length n and a REAL variable CONINV in which to return the reciprocal of the condition number. A vector w satisfying

$$\|Aw\|_1 = \mathrm{CONINV}\|A\|_1\|w\|_1$$

is easily obtained from z in (8.9) by using (8.3).

All the LINPACK code which performs condition estimation incorporates a scaling technique, designed to prevent overflow during computation of the condition estimates [4], [7]. Similar precautions are clearly desirable in the computation of the solution of (8.9). However, the LINPACK scaling technique requires $O(n^2)$ flops, which is prohibitively expensive in the context of SPTSL; for this special case we suggest a simple modification which brings the cost down to $O(n)$ flops. The modified scaling technique takes advantage of the bidiagonal nature of the matrix factors in (8.8).

ALGORITHM SCALE. Let $L=(l_{ij})$ be a nonsingular lower bidiagonal matrix. This algorithm computes a vector z, with $\|z\|_\infty \leqq 1$, and a number θ, such that $Lz = \theta b$.

```
(1)     (l_{1,0} := 0; z_0 := 0)
        θ := 1
        For i := 1 to n
(*)         α := θ * b_i − l_{i,i−1} * z_{i−1}
            If |α| > |l_ii| then
                β := l_ii / α
(*)             θ := θ * β
                w_i := β
                z_i := 1
            else
                w_i := 1
                z_i := α / l_ii.
(2)     μ := 1
        For i := n − 1 to 1 step −1
(*)         If w_{i+1} ≠ 1 then μ := μ * w_{i+1}
(*)         z_i := z_i * μ.
```

Cost. $2n$ flops, plus at most $4n$ flops for the scalings, in the statements marked with an asterisk.

The LINPACK scaling technique would, on introducing a scaling at the ith stage, immediately rescale all the previously computed values $z_1, \cdots, z_{i-1}$. Instead Algorithm Scale stores the scale factors in the vector w (thus an extra n storage locations are required), and is thereby able to rescale at the final stage in $O(n)$ flops.

A modification to the LINPACK scaling technique which attempts to reduce the frequency of the scalings is described in [11]; this modification could profitably be adopted in Algorithm Scale. In a small number of tests that we performed using a version of Algorithm Scale which incorporates the technique in [11], the cost of the scaling in Algorithm Scale was never greater than $2n$ flops.

Finally, we note that the basic modifications to SPTSL suggested above increase the computational cost of the routine from $5n$ flops to $11n$ flops ($2n$ of which arise from the evaluation of $\|A\|_1$). Incorporation of the scaling method used in Algorithm Scale could at worst add another $8n$ flops to the operation count.

9. Concluding remarks. We conclude by briefly discussing the choice of algorithm for computing $\|A^{-1}\|_\infty$ from among those presented. If the tridiagonal matrix A is positive definite or bidiagonal, then Algorithms 8.1 and 2.1 respectively are recommended. Both these algorithms are very satisfactory from the point of view of numerical stability. We note that the triangular systems which are solved are of the form discussed by Wilkinson [20, p. 250].

For general irreducible tridiagonal matrices A the choice is between Algorithms 4.1, 4.2 and 4.3, for which the differences in computational cost are relatively small. In the context of solving a linear system $Ax = b$, a factorisation $PA = LU$ is likely to be already available, in which case Algorithm 4.3 (minus the first step and using $PA = LU$ in the second step) is attractive. In general, Algorithm 4.2 is both more efficient and easier to "code-up" than Algorithms 4.1 and 4.3. As regards the very real dangers of overflow and underflow when these algorithms are implemented, and their numerical stability, see §§ 5 and 6.

Acknowledgments. I would like to thank Professor G. H. Golub and Dr. J. R. Cash for suggesting this work.

I also wish to thank Dr. I. Gladwell and Dr. G. Hall for their constructive criticism of the manuscript, and the referees and editor for their helpful suggestions.

REFERENCES

[1] B. BUKHBERGER AND G. A. EMEL'YANENKO, *Methods of inverting tridiagonal matrices*, USSR Computat. Math. and Math. Phys., 13 (1973), pp. 10-20.

[2] J. R. CASH, *Stable Recursions: With Applications to the Numerical Solution of Stiff Systems*, Academic Press, London, 1979.

[3] ———, private communication, 1983.

[4] A. K. CLINE, C. B. MOLER, G. W. STEWART AND J. H. WILKINSON, *An estimate for the condition number of a matrix*, SIAM J. Numer. Anal., 16 (1979), pp. 368-375.

[5] G. DAHLQUIST AND A. BJORCK, *Numerical Methods*, Prentice-Hall, Englewood Cliffs, NJ, 1974.

[6] C. DE BOOR AND A. PINKUS, *Backward error analysis for totally positive linear systems*, Numer. Math., 27 (1977), pp. 485-490.

[7] J. J. DONGARRA, J. R. BUNCH, C. B. MOLER AND G. W. STEWART, LINPACK *Users' Guide*, Society for Industrial and Applied Mathematics, Philadelphia, 1979.

[8] J. J. DONGARRA AND A. R. HINDS, *Unrolling loops in* FORTRAN, Software-Practice and Experience, 9 (1979), pp. 216-226.

[9] C. F. FISCHER AND R. A. USMANI, *Properties of some tridiagonal matrices and their application to boundary value problems*, SIAM J. Numer. Anal., 6 (1969), pp. 127-142.

[10] F. A. GRAYBILL, *Introduction to Matrices with Applications in Statistics*, Wadsworth, Belmont, CA, 1969.

[11] R. G. GRIMES AND J. G. LEWIS, *Condition number estimation for sparse matrices*, this Journal, 2 (1981), pp. 384-388.

[12] N. J. HIGHAM, *Upper bounds for the condition number of a triangular matrix*, Numerical Analysis Report No. 86, Univ. Manchester, England, 1983; submitted for publication.

[13] ———, *Matrix condition numbers*, M.Sc. Thesis, University of Manchester, England, 1983.

[14] Y. IKEBE, *On inverses of Hessenberg matrices*, Linear Algebra and Appl., 24 (1979), pp. 93-97.

[15] C. B. MOLER, MATLAB *User's Guide*, Technical Report CS81-1 (revised), Dept. Computer Science, Univ. New Mexico, Albuquerque, 1982.

[16] J. M. ORTEGA, *Numerical Analysis: A Second Course*, Academic Press, New York, 1972.

[17] G. W. STEWART, *Introduction to Matrix Computations*, Academic Press, New York, 1973.

[18] J. H. WILKINSON, *Error analysis of direct methods of matrix inversion*, J. Assoc. Comput. Mach., 8 (1961), pp. 281-330.

[19] ———, *Rounding Errors in Algebraic Processes*, Notes on Applied Science No. 32, Her Majesty's Stationery Office, London, 1963.

[20] ———, *The Algebraic Eigenvalue Problem*, Oxford Univ. Press, Oxford, 1965.

[21] T. YAMAMOTO AND Y. IKEBE, *Inversion of band matrices*, Linear Algebra and Appl., 24 (1979), pp. 105-111.

SIAM J. SCI. STAT. COMPUT.
Vol. 7, No. 1, January 1986

012

A NOTE ON MONITORING THE STABILITY OF TRIANGULAR DECOMPOSITION OF SPARSE MATRICES*

J. L. BARLOW†

Abstract. A method is proposed for bounding the numerical instability in Gaussian elimination. The computational cost of obtaining the bound contributes little to the cost of the elimination. The bound is a slight improvement on that given by Erisman and Reid.

Key words. Gaussian elimination, linear equations, backward error matrix, measure of stability

We consider the numerical solution by Gaussian elimination, using floating point arithmetic, of a system of linear equations

$$Ax = b \tag{1}$$

where A is a sparse nonsingular $n \times n$ matrix, b is an $n \times 1$ vector, and x is the $n \times 1$ solution vector. This process decomposes A into a lower triangular matrix $L = (l_{ij})$ and an upper triangular matrix $U = (u_{ij})$ which satisfy

$$A + E = LU \tag{2}$$

where $E = (e_{ij})$ is the backward error matrix. Reid [3] extended the error analysis of Wilkinson [4] to show that the entries of E satisfy

$$|e_{ij}| \leqq 3.01 \mu \rho m_{ij} \tag{3}$$

where μ is the machine unit, $\rho = \max_{(i,j,k)} |a_{ij}^{(k)}|$ is the largest element in any of the intermediate matrices $A^{(k)}$, $k = 0, 1, 2, \cdots, n-1$ encountered in the elimination, and m_{ij} is the number of multiplications required in the computation of l_{ij} if $i > j$ or u_{ij} if $i \leqq j$. If we neglect rounding errors, then the elements of $A^{(k)}$ satisfy

$$a_{ij}^{(k)} = a_{ij} - \sum_{m=1}^{k} l_{im} u_{mj}, \qquad 0 < k < i, j \leqq n \tag{4}$$

and $A^{(0)} = A$, $A^{(n-1)} = U$. Note that no assumption is made about the size of the elements of the lower triangular matrix L.

The size of the parameter ρ in equation (3) is an excellent measure of the stability of the LU decomposition. The applications of estimating ρ to sparse matrix factorization is discussed in [2]. The calculation of ρ takes as many comparisons as the number of times

$$a_{ij}^{(k)} = a_{ij}^{(k-1)} - m_{ik} a_{kj}^{(k-1)}$$

is done in the elimination process plus the number of nonzero elements in the matrix A. Thus it can contribute significantly to the cost of factoring A. The following proposition contains a method for reducing that cost.

PROPOSITION. *The parameter ρ in* (3) *satisfies the bound*

$$\rho \leqq \max_{(i)} \|l_{(i)}\|_p \max_{(j)} \|u^{(j)}\|_q \tag{5}$$

* Received by the editors February 2, 1984, and in revised form July 16, 1984. This work was supported by the National Science Foundation under contract MCS-8201065 and by the Office of Naval Research under contract N0014-80-0517.

† Department of Computer Science, Pennsylvania State University, University Park, Pennsylvania 16802.

where $l_{(i)}$ is the ith *row of the matrix L in* (2), *$u^{(j)}$ is the* jth *column of the matrix U in* (2), *$\|\cdot\|_p$, $\|\cdot\|_q$ are Hölder norms, and* $1/p+1/q=1$.

Proof. From the definition of Gaussian elimination, if we neglect rounding errors

$$u_{ij} = a_{ij} - \sum_{m=1}^{\min(i-1,j)} l_{im}u_{mj}, \qquad 0<i,j\leqq n \tag{6}$$

where $u_{ij}=0$ if $i\geqq j$. After subtracting (4) and (6), we get

$$a_{ij}^{(k)} - u_{ij} = \sum_{m=k+1}^{\min(i-1,j)} l_{im}u_{mj}, \qquad 0\leqq k\leqq i-1, j\leqq n.$$

Thus

$$a_{ij}^{(k)} = \sum_{m=k+1}^{\min(i,j)} l_{im}u_{mj}.$$

By the Hölder inequality

$$|a_{ij}^{(k)}| \leqq \begin{cases} \|(l_{i,k+1},\cdots,l_{ii})^T\|_p \|(u_{k+1,j},\cdots,u_{ij})^T\|_q, & 0\leqq k<i\leqq j\leqq n, \\ \|(l_{i,k+1},\cdots,l_{ij})^T\|_p \|(u_{k+1,j},\cdots,u_{jj})^T\|_q, & 0\leqq k<j<i\leqq n, \end{cases}$$

where $1/p+1/q=1$. The above is clearly bounded by (5). Q.E.D.

Erisman and Reid [2] give the bound

$$\rho \leqq \max_{(i,j)} |a_{ij}| + \max_{(i)} \|(l_{i1},\cdots,l_{i,i-1})T\|_p \max_{(j)} \|(u_{1j},\cdots,u_{j-1,j})^T\|_q \tag{7}$$

where $1/p+1/q=1$. This is generalization of a bound discovered by Businger [1].

The three most important special cases of (5) and (7) are when $p=1, 2, \infty$. Other values of p are unlikely to be used in practice.

The computation of the bounds (5) and (7) contribute little to the cost of factorization. Examples of matrices where (5) is sharper than (7) and vice versa are easy to construct. If

$$L=\begin{pmatrix} 1 & 0 \\ 2 & 1 \end{pmatrix} \quad \text{and} \quad U=\begin{pmatrix} 1 & 5 \\ 0 & 1 \end{pmatrix},$$

and we take $p=\infty$ and $q=1$, then the bound in (5) states that $\rho\leqq 11$ (which is exact) whereas the bound in (7) states that $\rho\leqq 21$. However, if

$$L=\begin{pmatrix} 1 & 0 \\ 2 & 1 \end{pmatrix} \quad \text{and} \quad U=\begin{pmatrix} 1 & -1 \\ 0 & 5 \end{pmatrix},$$

then the bound in (5) states that $\rho\leqq 12$, whereas the bound in (7) states that $\rho\leqq 5$ (which is exact). Thus one cannot say that either of the bounds (5) or (7) is sharper than the other.

The most convenient cases of the bounds (5) and (7) are when $p=\infty$ and $q=1$ or when $p=1$ and $q=\infty$. If A, L, and U each have k_A, k_L, and k_U nonzero elements, then the bound in (5) requires k_L+2n comparisons and k_U additions, whereas the bound in (7) requires k_A+k_L+n comparisons and k_U-n additions. Since $k_A>2n$ for many realistic sparse problems, the bound in (5) requires a little less computation.

Acknowledgments. The author would like to thank Richard J. Zaccone and the referees for their suggestions. He would also like to acknowledge Ramsey S. Barlow for proofreading a much longer earlier version of this paper.

REFERENCES

[1] P. A. BUSINGER, *Monitoring the numerical stability of Gaussian elimination*, Numer. Math., 16 (1971), pp. 360-361.

[2] A. M. ERISMAN AND J. K. REID, *Monitoring the stability of the triangular factorization of a sparse matrix*, Numer. Math., 22 (1974), pp. 183-186.

[3] J. K. REID, *A note on the stability of Gaussian elimination*, J. Inst. Math. Appl., 8 (1971), pp. 374-375.

[4] J. H. WILKINSON, *Error analysis of direct methods of matrix inversion*, J. Assoc. Comput. Mach., 8 (1961), pp. 281-330.

SIAM J. SCI. STAT. COMPUT.
Vol. 7, No. 1, January 1986

013

AN ALGORITHM FOR SIMULTANEOUS ORTHOGONAL TRANSFORMATION OF SEVERAL POSITIVE DEFINITE SYMMETRIC MATRICES TO NEARLY DIAGONAL FORM*

BERNHARD N. FLURY† AND WALTER GAUTSCHI‡

Abstract. For $k \geqq 1$ positive definite symmetric matrices $A_1, \cdots, A_k$ of dimension $p \times p$ we define the function $\Phi(A_1, \cdots, A_k; n_1, \cdots, n_k) = \prod_{i=1}^{k} [\det(\operatorname{diag} A_i)]^{n_i} / [\det(A_i)]^{n_i}$, where n_i are positive constants, as a measure of simultaneous deviation of $A_1, \cdots, A_k$ from diagonality. We give an iterative algorithm, called the FG-algorithm, to find an orthogonal $p \times p$-matrix B such that $\Phi(B^T A_1 B, \cdots, B^T A_k B; n_1, \cdots, n_k)$ is minimum. The matrix B is said to transform $A_1, \cdots, A_k$ simultaneously to nearly diagonal form. Conditions for the uniqueness of the solution are given.

The FG-algorithm can be used to find the maximum likelihood estimates of common principal components in k groups (Flury (1984)). For $k = 1$, the FG-algorithm computes the characteristic vectors of the single positive definite symmetric matrix A_1.

Key words. diagonalization, principal components, eigenvectors

1. The problem. It is well known (see, e.g., Basilevsky (1983, § 5.3) that if A is a positive definite symmetric (p.d.s.) matrix of dimension $p \times p$, then there exists a real orthogonal matrix B such that

$$B^T A B = \Lambda = \operatorname{diag}(\lambda_1, \cdots, \lambda_p), \tag{1.1}$$

where the λ_i are all positive. For $k > 1$ p.d.s. matrices $A_1, \cdots, A_k$ the associated orthogonal matrices are in general different. We call $A_1, \cdots, A_k$ simultaneously diagonalizable if there exists an orthogonal matrix B such that

$$B^T A_i B = \Lambda_i \text{ (diagonal)} \quad \text{for } i = 1, \cdots, k. \tag{1.2}$$

Conditions equivalent to (1.2) have been given by Flury (1983).

Now suppose that $A_1, \cdots, A_k$ are not simultaneously diagonalizable, but we wish to find an orthogonal matrix B which makes them simultaneously "as diagonal as possible" in a sense to be defined. As a simple measure of "deviation from diagonality" of a p.d.s. matrix F we can take

$$\varphi(F) = |\operatorname{diag} F| / |F|, \tag{1.3}$$

where $|\cdot|$ is the determinant and diag F is the diagonal matrix having the same diagonal elements as F. The fact that φ is a reasonable measure of deviation from diagonality can be seen from Hadamard's inequality (Noble and Daniel (1977, exercise 11.51)):

$$|F| \leqq |\operatorname{diag} F| \tag{1.4}$$

with equality exactly if F is diagonal. Therefore, $\varphi(F) \geqq 1$ holds, with equality exactly when F is diagonal. Actually, $\varphi(G)$ increases monotonically as G is continuously "inflated" from diag F to F. This can be seen from the following lemma.

LEMMA 1. *If $F = (f_{ij})$ is a p.d.s. matrix of dimension $p \times p$, then*

$$d(\alpha) := \det \begin{pmatrix} f_{11} & \alpha f_{12} & \cdots & \alpha f_{1p} \\ \alpha f_{21} & f_{22} & \cdots & \alpha f_{2p} \\ \vdots & \vdots & & \vdots \\ \alpha f_{p1} & \alpha f_{p2} & \cdots & f_{pp} \end{pmatrix} \tag{1.5}$$

* Received by the editors March 15, 1984. The work of the first author was supported by the Swiss National Science Foundation under contract #82.008.0.82 in the Department of Statistics at Purdue University.

† Department of Statistics, University of Berne, CH-3012 Berne, Switzerland.

‡ Department of Computer Sciences, Purdue University, West Lafayette, Indiana 47907. The work of this author was supported in part by the National Science Foundation under grant DCR 8320561.

is a decreasing function of α for $\alpha \in [0, 1]$. If F is not diagonal, $d(\alpha)$ is strictly decreasing.

Proof. The case $F = \operatorname{diag}(f_{11}, \cdots, f_{pp})$ is trivial; assume therefore that F is not diagonal. Write

$$\begin{aligned} d(\alpha) &= |\alpha F + (1-\alpha)\operatorname{diag} F| \\ &= |\operatorname{diag} F| \cdot |\alpha(\operatorname{diag} F)^{-1/2} F (\operatorname{diag} F)^{-1/2} + (1-\alpha) I_p| \end{aligned} \tag{1.6}$$

and note that $d(\alpha) > 0$ for all $\alpha \in [0, 1]$, since both F and $\operatorname{diag} F$ are p.d.s. Let $R = (\operatorname{diag} F)^{-1/2} F (\operatorname{diag} F)^{-1/2}$. R is p.d.s. with 1's on the main diagonal. Let $d_1(\alpha) = |\alpha R + (1-\alpha) I_p|$. Then $d_1(0) = 1$ and $d_1(1) < 1$ by Hadamard's inequality. It remains to show that $d_1(\alpha)$ is strictly decreasing in $(0, 1)$. Let $\rho_1 \geqq \rho_2 \geqq \cdots \geqq \rho_p > 0$ denote the eigenvalues of R. The eigenvalues of $\alpha R + (1-\alpha) I_p$ are

$$\gamma_j = \alpha \rho_j + 1 - \alpha \qquad (j = 1, \cdots, p), \tag{1.7}$$

and therefore

$$d_1(\alpha) = \prod_{j=1}^{p} \gamma_j = \prod_{j=1}^{p} (1 + \alpha(\rho_j - 1)). \tag{1.8}$$

Taking the first derivative gives

$$\frac{\partial d_1}{\partial \alpha} = \sum_{h=1}^{p} (\rho_h - 1) \prod_{\substack{j=1 \\ j \neq h}}^{p} [1 + \alpha(\rho_j - 1)] = d_1(\alpha) \sum_{j=1}^{p} \frac{\rho_j - 1}{1 + \alpha(\rho_j - 1)}, \tag{1.9}$$

where all denominators are positive because of $\rho_j > 0$ and $\alpha \leqq 1$. Letting

$$d_2(\alpha) = \sum_{j=1}^{p} \frac{\rho_j - 1}{1 + \alpha(\rho_j - 1)}, \tag{1.10}$$

we note that $d_2(0) = \sum_{j=1}^{p} (\rho_j - 1) = \operatorname{tr} R - p = 0$ and

$$\frac{\partial d_2}{\partial \alpha} = -\sum_{j=1}^{p} \frac{(\rho_j - 1)^2}{(1 + \alpha(\rho_j - 1))^2} < 0. \tag{1.11}$$

Therefore, $d_2(\alpha) < 0$ on $(0, 1]$, implying that $\partial d_1 / \partial \alpha < 0$ for $0 < \alpha \leqq 1$. This proves the lemma.

The reader may notice a similarity to ridge regression: Hoerl and Kennard (1970, Thms. 4.1 and 4.2) have given monotonicity properties of some functions related to the trace of the matrix $(F + \alpha I_p)^{-1}$ for $\alpha > 0$.

Let us now consider k p.d.s. matrices $F_1, \cdots, F_k$ and positive weights $n_1, \cdots, n_k$. Then we define the simultaneous deviation from diagonality of the matrices $F_1, \cdots, F_k$ with given weights $n_1, \cdots, n_k$ as

$$\Phi(F_1, \cdots, F_k; n_1, \cdots, n_k) = \prod_{i=1}^{k} [\varphi(F_i)]^{n_i}. \tag{1.12}$$

Let now $F_i = B^T A_i B$ $(i = 1, \cdots, k)$ for a given orthogonal matrix B. Then we can take

$$\Phi_0(A_1, \cdots, A_k; n_1, \cdots, n_k) = \min_{B \in O(p)} \Phi(B^T A_1 B, \cdots, B^T A_k B; n_1, \cdots, n_k), \tag{1.13}$$

where $O(p)$ is the group of orthogonal $p \times p$-matrices, as a measure of simultaneous diagonalizability of $A_1, \cdots, A_k$. Clearly, $\Phi_0 \geqq 1$ holds, with equality if and only if (1.2) is satisfied.

It can be shown (Flury (1984)) that if the minimum Φ_0 is attained for a matrix $B_0 = (b_1, \cdots, b_p) \in O(p)$, then the following system of equations holds:

$$b_l^T \left(\sum_{i=1}^{k} n_i \frac{\lambda_{il} - \lambda_{ij}}{\lambda_{il}\lambda_{ij}} A_i \right) b_j = 0 \qquad (l, j = 1, \cdots, p; l \neq j) \tag{1.14}$$

where

$$\lambda_{ih} = b_h^T A_i b_h \qquad (i = 1, \cdots, k; h = 1, \cdots, p). \tag{1.15}$$

In this paper we give an algorithm for finding B_0.

It may be noted that our measure of "deviation from diagonality" (formula (1.3)) is not the only natural one; one could, for instance, also consider the sum of squared off-diagonal elements. Our reason for considering (1.3) was that this criterion arises naturally from a statistical problem of maximizing a likelihood function in principal component analysis of several groups; see Flury (1984) for details.

2. The FG-algorithm. The FG-algorithm consists of two algorithms, called F and G respectively, which minimize Φ by iteration on two levels:

On the outer level (F-level), every pair (b_l, b_j) of column vectors of the current approximation B to the solution B_0 is rotated such that the corresponding equation in (1.14) is satisfied. One iteration step of the F-algorithm consists of rotations of all $p(p-1)/2$ pairs of vectors of B. The F-algorithm is similar to algorithms used in factor analysis to perform varimax and other rotations (see, e.g., Weber (1974)).

On the inner level (G-level), an orthogonal 2×2-matrix Q which solves a two dimensional analog of (1.14) is found by iteration. This matrix defines the rotation of a pair of vectors currently being used on the F-level.

THE F-ALGORITHM. Let

$$\Phi(B) = \Phi(B^T A_1 B, \cdots, B^T A_k B; n_1, \cdots, n_k) \tag{2.1}$$

denote the simultaneous deviation of $B^T A_1 B, \cdots, B^T A_k B$ from diagonality as a function of B, the A_i and n_i being considered as fixed. The F-algorithm yields a converging sequence of orthogonal matrices $B^{(0)}, B^{(1)}, \cdots$, such that $\Phi(B^{(f+1)}) \leqq \Phi(B^{(f)})$.

The algorithm proceeds as follows:

step F_0: Define $B = (b_1, \cdots, b_p) \in O(p)$ as an initial approximation to the orthogonal matrix minimizing Φ, e.g. $B \leftarrow I_p$. Put $f \leftarrow 0$.

step F_1: Put $B^{(f)} \leftarrow B$ and $f \leftarrow f+1$

step F_2: Repeat steps F_{21} to F_{24} for all pairs (l, j), $1 \leqq l < j \leqq p$:

step F_{21}: Put $H(p \times 2) \leftarrow (b_l, b_j)$ and

$$T_i(2 \times 2) \leftarrow \begin{pmatrix} b_l^T A_i b_i & b_l^T A_i b_j \\ b_j^T A_i b_l & b_j^T A_i b_j \end{pmatrix} \qquad (i = 1, \cdots, k).$$

The T_i are p.d.s.

step F_{22}: Perform the G-algorithm on $(T_1, \cdots, T_k)$ to get an orthogonal

$$2 \times 2 - \text{matrix } Q = \begin{pmatrix} \cos\alpha & -\sin\alpha \\ \sin\alpha & \cos\alpha \end{pmatrix}.$$

step F_{23}: Put $H^*(p \times 2) = (b_l^*, b_j^*) \leftarrow HQ$. (This corresponds to an orthogonal rotation of the two columns of H by an angle α).

step F_{24}: In the matrix B, replace columns b_l and b_j by b_l^* and b_j^*, respectively, and call the new matrix again B.

step F_3: If, for some small $\varepsilon_F > 0$, $\Phi(B^{(f-1)}) - \Phi(B) < \varepsilon_F$ holds, stop. Otherwise, start the next iteration step at F_1.

THE G-ALGORITHM. This algorithm solves the equation

$$q_1^T\left(n_1\frac{\delta_{11}-\delta_{12}}{\delta_{11}\delta_{12}}T_1+\cdots+n_k\frac{\delta_{k1}-\delta_{k2}}{\delta_{k1}\delta_{k2}}T_k\right)q_2=0, \tag{2.2}$$

where $T_1,\cdots,T_k$ are fixed p.d.s. 2×2-matrices, $n_i>0$ are fixed constants,

$$\delta_{ij}=q_j^T T_i q_j \qquad (i=1,\cdots,k;\, j=1,2), \tag{2.3}$$

and $Q=(q_1, q_2)$ is an orthogonal 2×2-matrix. The iteration of the algorithm yields a sequence of orthogonal matrices $Q^{(0)}, Q^{(1)},\cdots$, converging to a solution of (2.2).

The algorithm proceeds as follows:

step G_0: Define $Q(2\times 2)$ as an initial approximation to the solution of (2.2), e.g. $Q\leftarrow I_2$. Put $g\leftarrow 0$.

step G_1: Put $Q^{(g)}\leftarrow Q$ and $g\leftarrow g+1$.

step G_2: Compute the δ_{ij} (2.3), using the current Q. Put

$$T(2\times 2)\leftarrow n_1\frac{\delta_{11}-\delta_{12}}{\delta_{11}\delta_{12}}T_1+\cdots+n_k\frac{\delta_{k1}-\delta_{k2}}{\delta_{k1}\delta_{k2}}T_k.$$

step G_3: Compute the (normalized) eigenvectors of T. In $Q=(q_1, q_2)$, put $q_1\leftarrow$ first eigenvector of T, $q_2\leftarrow$ second eigenvector of T.

step G_4: If $\|Q^{(g-1)}-Q\|<\varepsilon_G$ (where $\|\cdot\|$ denotes a matrix norm and $\varepsilon_G>0$ is a small positive constant), stop. Otherwise, start the next iteration step at G_1. Note that, since the order of eigenvectors is arbitrary, as well as their signs, it may be necessary to exchange q_1 and q_2 and/or to multiply one or both columns of Q by -1 before comparing Q with $Q^{(g-1)}$.

The motivation for the two algorithms and their connection with the basic system of equations (1.14) is as follows. Suppose that the (l,j)th equation of (1.14) is to be solved. With $H=(b_l: b_j)$ denoting the current lth and jth columns of B, and λ_{ih} being defined as in (1.15), b_l and b_j are the desired solution if and only if the 2×2-matrix

$$\sum_{i=1}^{k} n_i\frac{\lambda_{il}-\lambda_{ij}}{\lambda_{il}\lambda_{ij}}T_i \tag{2.4}$$

is diagonal, where

$$T_i=H^T A_i H \qquad (i=1,\cdots,k). \tag{2.5}$$

Assume now that b_l and b_j do not solve the (l,j)th equation, but $b_l^*=Hq_1$ and $b_j^*=Hq_2$ do, where $Q=(q_1: q_2)$ is an orthogonal 2×2 matrix. Then

$$b_l^{*T}\left[\sum_{i=1}^{k} n_i\frac{\lambda_{il}^*-\lambda_{ij}^*}{\lambda_{il}^*\lambda_{ij}^*}A_i\right]b_j^*=0, \tag{2.6}$$

where

$$\lambda_{ih}^*=b_h^{*T}A_i B_h^* \qquad (i=1,\cdots,k,\, h=l,j). \tag{2.7}$$

Putting $H^*=(b_l^*: b_j^*)=HQ$, (2.6) holds precisely if

$$\sum_{i=1}^{k} n_i\frac{\lambda_{il}^*-\lambda_{ij}^*}{\lambda_{il}^*\lambda_{ij}^*}H^{*T}A_i H^* \tag{2.8}$$

is diagonal. Now we note that

$$H^{*T}A_i H^*=(HQ)^T A_i(HQ)=Q^T T_i Q, \tag{2.9}$$

$$\lambda_{il}^{*} = (Hq_1)^T A_i (Hq_1) = q_1^T T_i q_1 \qquad (i = 1, \cdots, k), \tag{2.10}$$

and

$$\lambda_{ij}^{*} = q_2^T T_i q_2 \qquad (i = 1, \cdots, k). \tag{2.11}$$

Thus the problem of rotating the lth and jth columns of B so as to satisfy (1.14) can be reduced completely to the problem of finding an orthogonal 2×2-matrix $Q = (q_1 : q_2)$ such that

$$q_1^T \left[\sum_{i=1}^{k} n_i \frac{\delta_{i1} - \delta_{i2}}{\delta_{i1}\delta_{i2}} T_i \right] q_2 = 0, \tag{2.12}$$

where δ_{i1} and δ_{i2} have been written in place of λ_{il}^{*} and λ_{ij}^{*}, respectively.

Since (2.12) is a 2-dimensional analogue of (1.14), and since the group of orthogonal 2×2-matrices is compact, it follows that a solution of (2.12) always exists.

The problem of solving (2.12) is itself nontrivial. Although (2.12) can be written in terms of a rotation angle α, solving for α would involve solving a polynomial equation of degree $4k$ in $\cos \alpha$ and $\sin \alpha$, which seems rather tedious. A more elegant solution is provided by the G-algorithm, which is based on the observation that the vectors q_1, q_2 satisfying (2.12) are eigenvectors of the matrix in brackets. Since the latter, however, depends also on q_1, q_2, an iterative procedure is required.

3. Convergence of the FG-algorithm.

3.1. Convergence of the F-algorithm. We show that the F-algorithm, in theory (i.e. if $\varepsilon_F = \varepsilon_G = 0$), does not stop unless the equations (1.14) are satisfied for the current B, and that, if B does not satisfy (1.14), an iteration step of the F-algorithm will decrease Φ.

Suppose that the current orthogonal matrix $B = (b_1, \cdots, b_p)$ does not satisfy the (l, j)th equation of (1.14). For notational simplicity, we can take $l = 1$ and $j = 2$ without loss of generality. Let us write $B = (B^{(1)} : B^{(2)})$, where $B^{(1)} = (b_1, b_2)$. In step F_{21}, the matrices $T_i = B^{(1)T} A_i B^{(1)}$ are passed to the G-algorithm. The G-algorithm gives back an orthogonal 2×2 matrix Q (step F_{22}). (Note that Q is not necessarily unique, depending upon the conventions used in the G-algorithm. We will consider every matrix $\bar{Q}$ obtained from Q by interchainging the columns of Q and/or multiplying one or both columns by -1 as *equivalent* to Q). Steps F_{23} and F_{24} correspond to the transformation

$$B^{*} = B \begin{pmatrix} Q & 0 \\ 0 & I_{p-2} \end{pmatrix} = (B^{(1)} Q : B^{(2)}). \tag{3.1}$$

B^{*} is orthogonal, since it is the product of two orthogonal matrices. Now we have

$$\begin{aligned} \Phi(B^{*}) &= \prod_{i=1}^{k} [|\operatorname{diag} B^{*T} A_i B^{*}| / |B^{*T} A_i B^{*}|]^{n_i} \\ &= \prod_{i=1}^{k} [|\operatorname{diag} Q^T B^{(1)T} A_i B^{(1)} Q| \cdot |\operatorname{diag} B^{(2)T} A_i B^{(2)}| / |A_i|]^{n_i}. \end{aligned} \tag{3.2}$$

It will be shown in § 3.2 that if, as assumed, (1.14) is not satisfied for $l = 1$ and $j = 2$, then

$$\prod_{i=1}^{k} |\operatorname{diag} Q^T B^{(1)T} A_i B^{(1)} Q|^{n_i} < \prod_{i=1}^{k} |\operatorname{diag} B^{(1)T} A_i B^{(1)}|^{n_i}. \tag{3.3}$$

If (1.14) is satisfied, Q will be equivalent to I_2, and hence (3.3) holds with equality.

Therefore, each iteration step of the F-algorithm decreases Φ, and the algorithm will stop only if (1.14) is satisfied.

3.2. Convergence of the G-algorithm. In analogy to (1.13), (1.14), the equation (2.2) is satisfied for the matrix $Q=(q_1, q_2)$ which minimizes

$$\left(\prod_{i=1}^{k}|T_i|^{n_i}\right)\Phi(Q)=\prod_{i=1}^{k}|\operatorname{diag}(Q^TT_iQ)|^{n_i}.$$

Let $Q^{(g)}$ denote the orthogonal 2×2 matrix after the gth iteration. We will show that

$$\prod_{i=1}^{k}|\operatorname{diag}Q^{(g+1)^T}T_iQ^{(g+1)}|^{n_i}\leqq\prod_{i=1}^{k}|\operatorname{diag}Q^{(g)^T}T_iQ^{(g)}|^{n_i}, \tag{3.4}$$

and that the sequence $Q^{(g)}$ converges to an orthogonal matrix which solves (2.2).

Suppose now that the $(g+1)$st iteration of the G-algorithm is being performed. It is somewhat simpler to prove the convergence if we introduce the following notation: Let $Q=(q_1, q_2)$ contain the current approximation to the solution of (2.2) and δ_{ij} be defined as in (2.3). Then we put

$$a_i=\frac{\delta_{i1}-\delta_{i2}}{\delta_{i1}\delta_{i2}}\qquad(i=1,\cdots,k), \tag{3.5}$$

$$T=\sum_{i=1}^{k}n_ia_iT_i, \tag{3.6}$$

and

$$U_i=Q^TT_iQ=\begin{pmatrix}u_{11}^{(i)} & u_{12}^{(i)}\\ u_{21}^{(i)} & u_{22}^{(i)}\end{pmatrix}. \tag{3.7}$$

The U_i are p.d.s., and clearly

$$T=\sum_{i=1}^{k}n_ia_iQU_iQ^T \tag{3.8}$$

and

$$\delta_{i1}=u_{11}^{(i)},\qquad\delta_{i2}=u_{22}^{(i)}. \tag{3.9}$$

In step G_3 the characteristic vectors of T are computed. Denote the solution by Q^*, so that

$$Q^{*T}TQ^*=\Lambda \tag{3.10}$$

is diagonal. From (3.8) it follows that

$$\sum_{i=1}^{k}n_ia_iQ^{*T}QU_iQ^TQ^*=\Lambda. \tag{3.11}$$

The characteristic vectors of the symmetric matrix

$$U=\sum_{i=1}^{k}n_ia_iU_i \tag{3.12}$$

are therefore given by the orthogonal matrix

$$P=Q^TQ^*, \tag{3.13}$$

and Q^* can therefore be obtained by

$$Q^* = QP. \tag{3.14}$$

Note that $U = Q^T T Q$ is diagonal if and only if $Q = (q_1, q_2)$ is a solution of (2.2). From (3.9) and (3.12) it follows that

$$U = \sum_{i=1}^{k} n_i \frac{u_{11}^{(i)} - u_{22}^{(i)}}{u_{11}^{(i)} u_{22}^{(i)}} \begin{pmatrix} u_{11}^{(i)} & u_{12}^{(i)} \\ u_{21}^{(i)} & u_{22}^{(i)} \end{pmatrix}. \tag{3.15}$$

Let

$$\vartheta_i = \begin{cases} 1 & \text{if } u_{11}^{(i)} > u_{22}^{(i)}, \\ -1 & \text{if } u_{11}^{(i)} < u_{22}^{(i)}, \\ 0 & \text{if } u_{11}^{(i)} = u_{22}^{(i)}, \end{cases} \tag{3.16}$$

$$a_i' = \vartheta_i a_i, \tag{3.17}$$

and

$$S_i = \begin{pmatrix} s_{11}^{(i)} & s_{12}^{(i)} \\ s_{21}^{(i)} & s_{22}^{(i)} \end{pmatrix} = a_i' U_i \qquad (i = 1, \cdots, k). \tag{3.18}$$

S_i is p.d.s., unless $\vartheta_i = 0$. With this notation, we have

$$U = \sum_{i=1}^{k} n_i \vartheta_i S_i. \tag{3.19}$$

Now let $k' \leqq k$ denote the number of ϑ_i's which are not zero. Without loss of generality assume that the $k - k'$ matrices S_i which are zero have the indices $k'+1, \cdots, k$. Therefore the sum (3.19) extends only up to k', and we are going to show that

$$\prod_{i=1}^{k'} |\operatorname{diag} P^T S_i P|^{n_i} \leqq \prod_{i=1}^{k'} (s_{11}^{(i)} s_{22}^{(i)})^{n_i}, \tag{3.20}$$

with equality if and only if U is diagonal. Assuming for the moment that (3.20) holds true, the proof of (3.4) can be completed by noting that (3.20) implies

$$\prod_{i=1}^{k'} (a_i^2 \vartheta_i^2 |\operatorname{diag} P^T U_i P|)^{n_i} \leqq \prod_{i=1}^{k'} (a_i^2 \vartheta_i^2 u_{11}^{(i)} u_{22}^{(i)})^{n_i} \tag{3.21}$$

and therefore

$$\prod_{i=1}^{k'} |\operatorname{diag} Q^{*T} T_i Q^*|^{n_i} \leqq \prod_{i=1}^{k'} |\operatorname{diag} Q^T T_i Q|^{n_i}. \tag{3.22}$$

For the remaining $k - k'$ matrices $U_i (i = k'+1, \cdots, k)$ we have $u_{11}^{(i)} = u_{22}^{(i)}$, and therefore, as is easily verified,

$$|\operatorname{diag} B^T U_i B| \leqq |\operatorname{diag} U_i| \tag{3.23}$$

for any $B \in 0(2)$, with equality exactly if B is equivalent to I_2 or $u_{12}^{(i)} = 0$. This holds, in particular, for $B = P$. Putting (3.22) and (3.23) together gives now the desired result (3.4). It remains to show (3.20).

Let $P = (p_1, p_2)$ denote the eigenvectors of $U = \sum_{i=1}^{k'} \vartheta_i n_i S_i$, with p_1 being associated with the algebraically larger root. Since U is symmetric, P is orthogonal (or can be so chosen if the two roots are identical), and both characteristic roots are real. Assume

that U is not diagonal, and let $\varepsilon_i (i=1,\cdots,k')$ be defined by

$$P^T S_i P = \begin{pmatrix} s_{11}^{(i)} + \vartheta_i \varepsilon_i & \cdot \\ \cdot & s_{22}^{(i)} - \vartheta_i \varepsilon_i \end{pmatrix}. \tag{3.24}$$

From (3.5), (3.9) and (3.16) to (3.18) we have

$$s_{11}^{(i)} s_{22}^{(i)} = \vartheta_i (s_{11}^{(i)} - s_{22}^{(i)}), \quad \text{or} \quad 1 = \vartheta_i \left(\frac{1}{s_{22}^{(i)}} - \frac{1}{s_{11}^{(i)}} \right) \qquad (i=1,\cdots,k'), \tag{3.25}$$

which implies that either $s_{11}^{(i)}$ or $s_{22}^{(i)}$ is smaller than 1. It then follows that $\varepsilon_i < 1$ $(i=1,\cdots,k')$. Indeed, if $\vartheta_i = 1$, then $s_{22}^{(i)} < 1$ by (3.25), and the positivity of $s_{22}^{(i)} - \varepsilon_i$ implies $\varepsilon_i < 1$. If $\vartheta_i = -1$, then $s_{11}^{(i)} < 1$, and $s_{11}^{(i)} - \varepsilon_i > 0$ implies again $\varepsilon_i < 1$.

The product of the diagonal elements of $P^T S_i P$ is

$$\begin{aligned} |\operatorname{diag} P^T S_i P| &= (s_{11}^{(i)} + \vartheta_i \varepsilon_i)(s_{22}^{(i)} - \vartheta_i \varepsilon_i) \\ &= s_{11}^{(i)} s_{22}^{(i)} - \varepsilon_i \vartheta_i (s_{11}^{(i)} - s_{22}^{(i)}) - \varepsilon_i^2 \\ &= (1-\varepsilon_i) s_{11}^{(i)} s_{22}^{(i)} - \varepsilon_i^2 \\ &\leqq (1-\varepsilon_i) s_{11}^{(i)} s_{22}^{(i)} \qquad (i=1,\cdots,k'). \end{aligned} \tag{3.26}$$

Thus,

$$\prod_{i=1}^{k'} |\operatorname{diag} P^T S_i P|^{n_i} \leqq \left(\prod_{i=1}^{k'} (1-\varepsilon_i)^{n_i} \right) \left(\prod_{i=1}^{k'} |\operatorname{diag} S_i|^{n_i} \right), \tag{3.27}$$

and (3.20) holds if we can prove that

$$\prod_{i=1}^{k'} (1-\varepsilon_i)^{n_i} < 1. \tag{3.28}$$

To demonstrate this, we note that, since U is assumed not diagonal,

$$p_1^T U p_1 > u_{11}, \tag{3.29}$$

or equivalently,

$$\begin{aligned} &\sum_{i=1}^{k'} \vartheta_i n_i p_1^T S_i p_1 > \sum_{i=1}^{k'} \vartheta_i n_i s_{11}^{(i)}, \\ &\sum_{i=1}^{k'} \vartheta_i n_i (p_1^T S_i p_1 - s_{11}^{(i)}) > 0. \end{aligned} \tag{3.30}$$

Since $p_1^T S_i p_1 - s_{11}^{(i)} = \vartheta_i \varepsilon_i (i=1,\cdots,k')$, this implies

$$\sum_{i=1}^{k'} n_i \varepsilon_i > 0, \tag{3.31}$$

so that not all ε_i can be zero. On the other hand, if U is diagonal, then P is equivalent to I_2, and all ε_i are zero. Therefore the ε_i vanish simultaneously if and only if U is diagonal. Now we need the following lemma.

LEMMA 2. *If* $x_i > 0$, $n_i > 0 (i=1,\cdots,k')$ *and* $\sum_{i=1}^{k'} n_i x_i \leqq \sum_{i=1}^{k'} n_i$, *then* $\prod_{i=1}^{k'} x_i^{n_i} \leqq 1$.

Proof. Maximize the function $\prod_{i=1}^{k'} x_i^{n_i}$ under the restriction $\sum_{i=1}^{k'} n_i x_i = g$ (>0), using a Lagrange multiplier. The maximum has the value $(g/n)^n$ and is attained for $x_1 = \cdots = x_{k'} = g/n$, where $n = \sum_{i=1}^{k'} n_i$. Noting that $g \leqq n$ completes the proof.

Since $\varepsilon_i < 1 (i=1,\cdots,k')$ and $\sum_{i=1}^{k'} n_i \varepsilon_i > 0$, we can use Lemma 2 with $x_i = 1 - \varepsilon_i$ and get (3.28). Note that equality in (3.28) holds exactly if all ε_i are zero. This completes the proof of convergence of the G-algorithm.

4. Conditions for uniqueness of the solution. In § 3 we have shown that the FG-algorithm converges to a minimum of (2.1), unless the initial approximation of the orthogonal matrix B is (badly) chosen as a stationary point of Φ. However, we do not know whether Φ has a unique minimum. We are now going to show that in some "extreme" cases there exist more than one local minimum, and we give approximate conditions when this will happen. Throughout this section (unless otherwise stated) we will only consider the case $k=2$ and $p=2$.

Let the p.d.s. matrix S_1 have the characteristic roots $l_1 > l_2$ (the case $l_1 = l_2$ being trivial), and assume, for simplicity, that

$$S_1 = \begin{pmatrix} l_1 & 0 \\ 0 & l_2 \end{pmatrix}. \tag{4.1}$$

From (3.5) it can be seen that the solutions of (2.2) are unaffected by proportionality, i.e., we can assume (see also (3.25))

$$l_1 - l_2 = l_1 l_2 \tag{4.2}$$

without loss of generality. Consider now an orthogonal matrix

$$B = B(\varphi) = \begin{pmatrix} \cos\varphi & -\sin\varphi \\ \sin\varphi & \cos\varphi \end{pmatrix}. \tag{4.3}$$

The product of the diagonal elements of $B^T S_1 B$ is

$$\begin{aligned} |\mathrm{diag}(B^T S_1 B)| &= [l_2 + (l_1 - l_2)\cos^2\varphi][l_1 - (l_1 - l_2)\cos^2\varphi] \\ &= l_1 l_2 + (l_1 - l_2)^2 \cos^2\varphi \sin^2\varphi \\ &= l_1 l_2 [1 + l_1 l_2 \cos^2\varphi \sin^2\varphi] \\ &= r_1 [1 + r_1 \cos^2\varphi \sin^2\varphi], \end{aligned} \tag{4.4}$$

where

$$r_1 = l_1 l_2 \tag{4.5}$$

denotes the product of the characteristic roots of S_1. Let the eccentricity d_1 of S_1 be defined as the ratio of the larger to the smaller root of S_1,

$$d_1 = l_1 / l_2, \tag{4.6}$$

which is also the Euclidean condition number of S_1. From (4.2) it follows that $l_2 = l_1/(l_1+1)$, and therefore $d_1 = l_1 + 1$. Similarly, $d_1 = 1/(1-l_2)$, and therefore

$$l_1 = d_1 - 1, \qquad l_2 = (d_1 - 1)/d_1. \tag{4.7}$$

Multiplying these two equations gives

$$r_1 = (d_1 - 1)^2 / d_1. \tag{4.8}$$

Note that d_1 does not depend on the absolute size of S_1 (every matrix proportional to S_1 has the same eccentricity), and so the same is true for r_1.

For a second p.d.s. matrix S_2, let d_2 denote its eccentricity, and

$$r_2 = (d_2 - 1)^2 / d_2. \tag{4.9}$$

Let

$$B_0 = \begin{pmatrix} \cos\varphi_0 & -\sin\varphi_0 \\ \sin\varphi_0 & \cos\varphi_0 \end{pmatrix}$$

denote the orthogonal matrix which diagonalizes S_2. Then, in analogy to (4.4), we get

$$|\text{diag}\,(B^T S_2 B)| = r_2[1 + r_2 \cos^2(\varphi - \varphi_0) \sin^2(\varphi - \varphi_0)]. \tag{4.10}$$

The function Φ to be minimized is

$$\Phi(\varphi) = [1 + r_1 \cos^2\varphi \sin^2\varphi]^{n_1}[1 + r_2 \cos^2(\varphi - \varphi_0) \sin^2(\varphi - \varphi_0)]^{n_2}. \tag{4.11}$$

Let us now assume that $n_1 = n_2$, so that it remains to minimize

$$G(\varphi) = [1 + \tfrac{1}{4} r_1 \sin^2(2\varphi)][1 + \tfrac{1}{4} r_2 \sin^2(2(\varphi - \varphi_0))]. \tag{4.12}$$

$G(\varphi)$ is $\pi/2$-periodic, and from (4.12) it becomes clear that for $\varphi_0 \neq 0$, $G(\varphi)$ may have more than one local minimum in one period, depending on r_1, r_2 and φ_0 (and, in the general situation, on n_1 and n_2). Note that φ_0 is the minimum angle between two characteristic vectors of S_1 and S_2.

Let us first look at the extreme situation $\varphi_0 = \pi/4$. From a Taylor expansion it can be seen that in a neighborhood of 0,

$$G(\varphi) = 1 + \tfrac{1}{4} r_2 + (r_1 - r_2 + \tfrac{1}{4} r_1 r_2)\varphi^2 + O(\varphi^4). \tag{4.13}$$

The function $G(\varphi)$ has therefore a stationary point at $\varphi = 0$, which is a

$$\begin{aligned} &\text{minimum,} \quad \text{if } r_1 - r_2 + \tfrac{1}{4} r_1 r_2 > 0, \\ &\text{maximum,} \quad \text{if } r_1 - r_2 + \tfrac{1}{4} r_1 r_2 < 0. \end{aligned} \tag{4.14}$$

Note that for $r_1 \geqq 4$ or $r_1 = r_2$ this is always a minimum.

Similarly, at $\varphi = \pi/4$, we get a

$$\begin{aligned} &\text{minimum,} \quad \text{if } r_2 - r_1 + \tfrac{1}{4} r_1 r_2 > 0, \\ &\text{maximum,} \quad \text{if } r_2 - r_1 + \tfrac{1}{4} r_1 r_2 < 0. \end{aligned} \tag{4.15}$$

For $r_2 \geqq 4$ or $r_1 = r_2$ this is always a minimum.

Since r_1 and r_2 are both positive, there cannot be a maximum at 0 and $\pi/4$ simultaneously. Local minima at both points, however, are obtained e.g. if both r_1 and r_2 are larger than 4, or if $r_1 = r_2$ (even if $r_1 = r_2$ is very close to zero!). Thus the case of equal eccentricity of both matrices seems most "dangerous" in terms of multiple local minima.

Using the relation $r_i = (d_i - 1)^2 / d_i$(4.8, 4.9), the conditions (4.14) and (4.15) can be transformed to conditions on the eccentricities d_i ($i = 1, 2$). Figure 1 shows a partition of $[1, \infty) \times [1, \infty)$ into three areas in which a minimum is attained at 0 only, at $\pi/4$ only, or at both points, depending on the values of d_1 and d_2. Note that for $d_1 > 5.828427$ ($d_2 > 5.828427$) there is always a minimum at $\varphi = 0$ ($\varphi = \pi/4$), and if $d_1 = d_2$, there are always two minima.

The case $\varphi_0 = \pi/4$ treated so far is of course the "worst possible" case, since the minimum angle between two characteristic vectors of S_1 and S_2 cannot exceed $\pi/4$. For the application in common principal component analysis (Flury (1984)), however, we expect φ_0 rather close to zero, if the null hypothesis of identical principal components in the populations holds. Therefore we look now at the situation where φ_0 is close to zero. Without loss of generality we can assume $\varphi_0 > 0$. Again, for simplicity, we take $n_1 = n_2 = 1$.

Approximating the trigonometric functions in the two factors of G by Taylor series at $\varphi = 0$ (first factor) and at $\varphi = \varphi_0$ (second factor), and taking the first derivative

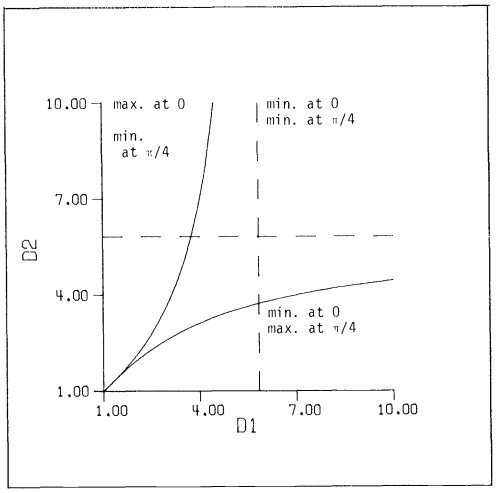


FIG. 1. *Conditions for minima and maxima if* $n_1 = n_2$, $\varphi_0 = \pi/4$.

of G with respect to φ yields

$$G'(\varphi) = 2r_1[\varphi + O(\varphi^3)][1 + r_2((\varphi - \varphi_0)^2 + O(\varphi - \varphi_0)^4)] + 2r_2[(\varphi - \varphi_0) + O(\varphi - \varphi_0)^3][1 + r_1(\varphi^2 + O(\varphi^4))]. \tag{4.16}$$

If φ_0 is close to zero, sufficient accuracy can be had for $0 \leqq \varphi \leqq \varphi_0$ if we ignore all terms of order higher than 2. An approximation to the solution(s) of $G'(\varphi) = 0$ within $[0, \varphi_0]$ is therefore given by the solution(s) of

$$r_1\varphi[1 + r_2(\varphi - \varphi_0)^2] + r_2(\varphi - \varphi_0)[1 + r_1\varphi^2] = 0. \tag{4.17}$$

This equation has either one or three real roots, depending on r_1, r_2 and φ_0. For $r_1 = r_2 = r$, (4.17) can be written as

$$\varphi(1 + r(\varphi - \varphi_0)^2) + (\varphi - \varphi_0)(1 + r\varphi^2) = 2\left(\varphi - \frac{\varphi_0}{2}\right)(r\varphi^2 - r\varphi_0\varphi + 1) = 0. \tag{4.18}$$

Thus $\varphi = \varphi_0/2$ is a solution of (4.18) (and also of (4.16) if $r_1 = r_2$). If, approximately,

$$\frac{4}{r} > \varphi_0^2, \tag{4.19}$$

$G(\varphi)$ takes a minimum at $\varphi_0/2$. Under the same condition (4.19), the polynomial

$$r\varphi^2 - r\varphi_0\varphi + 1 \tag{4.20}$$

has no real root, and the minimum at $\varphi_0/2$ is unique.

If, always approximately for small φ_0, $4/r < \varphi_0^2$, we get a maximum at $\varphi_0/2$, and two minima at

$$\tfrac{1}{2}(\varphi_0 \pm \sqrt{\varphi_0^2 - 4/r}). \tag{4.21}$$

In terms of the eccentricity parameters $d_1 = d_2 = d$, condition (4.19) becomes

$$\frac{4d}{(d-1)^2} > \varphi_0^2, \tag{4.22}$$

which shows that two minima are to be expected only if the eccentricity is high. For large d, (4.22) is approximately the same as

$$d < \left(\frac{2}{\varphi_0}\right)^2. \tag{4.23}$$

For example, if $\varphi_0 = .2 (\approx 11.5$ degrees), a single minimum can be expected approximately if $d < 100$.

Figure 2 shows the typical behavior of the function $G(\varphi)$ for $\varphi_0 = .2$ and $r = 160$ ($d = 161.99$). The two minima are approximately at .039 and .161. If different values are chosen for r_1 and r_2, the two minima are in general not identical, but the shape of the graph is similar, with one "valley" being less deep than the other.

Although these results are only approximate, they give a general idea about the conditions for uniqueness of the minimum. For $k > 2$ matrices, the relations are of course more complicated, but still we can expect a unique minimum unless some of the matrices are highly eccentric.

For dimension $p > 2$, the minimum is certainly unique if all the $p(p-1)/2$ equations (1.14) have a unique minimizing solution. (By a minimizing solution we mean a solution which corresponds to a local minimum of G, or, in the p-dimensional case, of the function Φ.) On the other hand, if some of the equations have more than one minimizing solution, this does not necessarily imply that the whole system (1.14) has more than one minimizing solution.

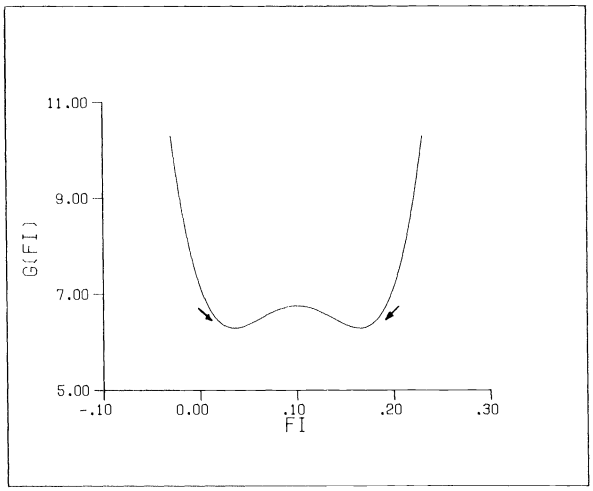


FIG. 2. *Graph of* $G(\varphi)$ *for* $\varphi_0 = .2$, $n_1 = n_2 = 1$ *and* $d_1 = d_2 = 162$.

A solution given by the FG-algorithm does of course not prove its uniqueness. However, Fig. 2 suggests the following: If we start the FG-algorithm with

$$B(0) = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

as an initial approximation, it will converge to the left minimum, while

$$B(\varphi_0) = \begin{pmatrix} \cos\varphi_0 & -\sin\varphi_0 \\ \sin\varphi_0 & \cos\varphi_0 \end{pmatrix}$$

as an initial approximation leads to convergence to the right minimum. (This is indicated

by the arrows in Fig. 2.) Since the function Φ is a product of k functions (see 1.12), minimizing solutions can always be expected to be somehow "close" to the characteristic vectors of one of the matrices. Therefore, if there is doubt about the uniqueness of the solution, it is recommended that one run the FG-algorithm k times, using the k sets of characteristic vectors of $A_1, \cdots, A_k$ as initial approximations. If all k solutions found are equal, it is reasonable to assume that there is a unique global minimum.

As a numerical example, let

$$S_1 = \begin{pmatrix} 100 & 0 \\ 0 & 1 \end{pmatrix} \quad \text{and} \quad S_2 = \begin{pmatrix} 96.0143 & 19.4603 \\ 19.4603 & 4.9857 \end{pmatrix},$$

so that $d_1 = d_2 = 100$ and $\varphi_0 = 202 (\approx 11.57$ degrees), which is a borderline case according to approximation (4.22). $G(\varphi)$ assumes two minima at .08 and .12, approximately. If we reduce the eccentricity to 90 (leaving φ_0 unchanged), we get the matrices

$$S_1 = \begin{pmatrix} 90 & 0 \\ 0 & 1 \end{pmatrix} \quad \text{and} \quad S_2 = \begin{pmatrix} 86.4168 & 17.4946 \\ 17.4946 & 4.5831 \end{pmatrix}.$$

For these two matrices, there is a unique minimum at $\varphi_0/2$. The bound (4.22) for d is in general too high, but the approximation becomes better when φ_0 gets smaller.

5. Remarks.

1. The proof of convergence of the G-algorithm makes strong use of the assumption that the matrices T_i are positive definite. If one or several of the matrices A_i are close to singularity, this could cause numerical problems, because the a_i (3.5) might become very large.

2. Since the stopping rule given in step F_3 depends on the absolute size of the matrices A_i, it may be better to replace it by a criterion similar to the one used in the G-algorithm:

F_3: If $\|B^{(f-1)} - B\| < \varepsilon_F$ for some small $\varepsilon_F > 0$, stop. Otherwise, start the next iteration step at F_1.

3. If the current version of B in the F-algorithm is a stationary point of Φ, and I_2 is taken as an initial approximation of Q in the G-algorithm, FG will not change B, since (1.14) is satisfied. This occurs, e.g., if the diagonal elements of the A_i-matrices are identical for each A_i, that is, diag $A_i = \text{diag}(c_i, \cdots, c_i)$ for some $c_i > 0 (i = 1, \cdots, k)$, and I_p is taken as an initial approximation of B. An important special case of this are correlation matrices, where the diagonal elements are all 1. If the first iteration of the F-algorithm does not change B, it might therefore be helpful to try FG with another initial approximation.

4. On the F-level, a better initial approximation than I_p might be to take the eigenvectors of one of the A_i (e.g. the one with the largest n_i), or the eigenvectors of $\sum_{i=1}^{k'} n_i A_i$. On the G-level, I_2 is a good initial approximation for Q, when the current B on the F-level is already close to the correct solution.

5. In step F_{24}, the lth and jth column of B are adjusted using the matrix Q given by the G-algorithm. Since these two columns will undergo changes in subsequent executions of steps F_{21} to F_{24}, it is not necessary to iterate on the G-level until full convergence is reached. In most cases the first iteration steps of the G-algorithm will decrease $\Phi(B)$ much more than the later iterations. If $k = 1$, only one iteration step is required in each execution of the G-algorithm.

6. In order to avoid permutations of the columns of B and multiplications by -1, it is convenient to order the columns of Q such that

$$Q = \begin{pmatrix} \cos\alpha & -\sin\alpha \\ \sin\alpha & \cos\alpha \end{pmatrix} \tag{5.1}$$

where $-\pi/2 < \alpha < \pi/2$.

7. If $k = 1$, the FG-algorithm reduces to a Jacobi-method (Parlett (1980, Chap. 9)) for diagonalizing the single p.d.s. matrix $A = A_1$.

8. The listing of a FORTRAN program performing the FG-algorithm (Flury (1985)) can be obtained from the first author upon request.

6. Example. In this section we illustrate the performance of the FG-algorithm by a numerical example of dimension $p = 6$ with $k = 2$ matrices and weights $n_1 = n_2 = 1$. The matrices are

$$A_1 = \begin{pmatrix} 45 & 10 & 0 & 5 & 0 & 0 \\ 10 & 45 & 5 & 0 & 0 & 0 \\ 0 & 5 & 45 & 10 & 0 & 0 \\ 5 & 0 & 10 & 45 & 0 & 0 \\ 0 & 0 & 0 & 0 & 16.4 & -4.8 \\ 0 & 0 & 0 & 0 & -4.8 & 13.6 \end{pmatrix},$$

$$A_2 = \begin{pmatrix} 27.5 & -12.5 & -.5 & -4.5 & -2.04 & 3.72 \\ -12.5 & 27.5 & -4.5 & -.5 & 2.04 & -3.72 \\ -.5 & -4.5 & 24.5 & -9.5 & -3.72 & -2.04 \\ -4.5 & -.5 & -9.5 & 24.5 & 3.72 & 2.04 \\ -2.04 & 2.04 & -3.72 & 3.72 & 54.76 & -4.68 \\ 3.72 & -3.72 & -2.04 & 2.04 & -4.68 & 51.24 \end{pmatrix}.$$

The characteristic vectors of A_1 are the columns of the matrix

$$B_1 = \begin{pmatrix} .5 & .5 & .5 & .5 & 0 & 0 \\ .5 & .5 & -.5 & -.5 & 0 & 0 \\ .5 & -.5 & -.5 & .5 & 0 & 0 \\ .5 & -.5 & .5 & -.5 & 0 & 0 \\ 0 & 0 & 0 & 0 & .8 & .6 \\ 0 & 0 & 0 & 0 & -.6 & .8 \end{pmatrix}.$$

The associated roots are 60, 50, 40, 30, 20 and 10. For matrix A_2, the characteristic vectors are

$$B_2 = \begin{pmatrix} .5 & .5 & .3 & -.6 & .1 & -.2 \\ .5 & .5 & -.3 & .6 & -.1 & .2 \\ .5 & -.5 & -.6 & -.3 & -.2 & -.1 \\ .5 & -.5 & .6 & .3 & .2 & .1 \\ 0 & 0 & -.18 & -.26 & .54 & .78 \\ 0 & 0 & -.26 & .18 & .78 & -.54 \end{pmatrix},$$

with roots 10, 20, 30, 40, 50 and 60. We used the FG-algorithm as programmed by Flury (1985) with $\varepsilon_F = \varepsilon_G = .0001$. The stopping rule for the F-algorithm was as described in Remark 2 above. As an initial approximation we used $B = I_6$, the identity matrix of dimension 6. The rotation pairs in the F-algorithm were chosen cyclically

(see Golub and Van Loan (1983, p. 299)). A sweep (or iteration step) of the F-algorithm consists therefore of $\binom{6}{2} = 15$ pairwise rotations. For each sweep of the F-algorithm, we give the current orthogonal matrix B, the value of the criterion $\Phi(B) = \prod_{i=1}^{2} |\mathrm{diag}(B^T A_i B)|/|A_i|$ and the average number of iterations of the G-algorithm per call. At the beginning, the value of the criterion is $\Phi(I_6) = 2.24718$.

after sweep 1

$$B^{(1)} = \begin{pmatrix} .8138 & -.0721 & .4584 & .3474 & -.0398 & .0124 \\ .0000 & .7646 & .4627 & -.4451 & .0553 & -.0114 \\ -.1321 & -.6346 & .5378 & -.5358 & -.0456 & -.0370 \\ -.5642 & 0384 & .5353 & .6256 & .0321 & .0358 \\ .0390 & -.0699 & .0002 & -.0357 & .7998 & .5938 \\ -.0224 & .0326 & .0003 & -.0379 & -.5937 & .8028 \end{pmatrix},$$

$\Phi(B^{(1)}) = 1.25461$

average number of iterations of G-algorithm: 2.73

after sweep 2

$$B^{(2)} = \begin{pmatrix} .4983 & -.5648 & .4993 & .4174 & -.0955 & .0072 \\ .5025 & .5613 & .4998 & -.4164 & .0955 & -.0072 \\ -.5003 & -.4124 & .5003 & -.5711 & -.0537 & -.0180 \\ -.4988 & .4150 & .5006 & .5703 & .0538 & .0180 \\ .0003 & -.1276 & -.0001 & -.0010 & .7921 & .5968 \\ .0003 & .0864 & .0000 & -.0323 & -.5903 & .8019 \end{pmatrix}$$

$\Phi(B^{(2)}) = 1.03574$

average number of iterations of G-algorithm: 2.4

after sweep 3

$$B^{(3)} = \begin{pmatrix} .5000 & -.5548 & .5000 & .4278 & -.0956 & .0083 \\ .5000 & .5548 & .5000 & -.4278 & .0956 & -.0083 \\ -.5000 & -.4247 & .5000 & -.5625 & -.0541 & -.0170 \\ -.5000 & .4247 & .5000 & .5625 & .0541 & .0170 \\ .0000 & -.1265 & .0000 & .0013 & .7919 & .5974 \\ .0000 & .0878 & .0000 & -.0336 & -.5905 & .8015 \end{pmatrix}$$

$\Phi(B^{(3)}) = 1.03568$

average number of iterations of G-algorithm: 1.8

after sweep 4

$$B^{(4)} = \begin{pmatrix} .5000 & -.5545 & .5000 & .4281 & -.0956 & .0083 \\ .5000 & .5545 & .5000 & -.4281 & .0956 & -.0083 \\ -.5000 & -.4250 & .5000 & -.5623 & -.0541 & -.0169 \\ -.5000 & .4250 & .5000 & .5623 & .0541 & .0169 \\ .0000 & -.1265 & .0000 & .0014 & .7919 & .5974 \\ .0000 & .0878 & .0000 & -.0337 & -.5906 & .8015 \end{pmatrix}$$

$\Phi(B^{(4)}) = 1.03568$

average number of iterations of G-algorithm: 1.07.

Sweep 5 did not produce any changes in the first four digits of the elements of B and the algorithm stopped. The "nearly diagonal" matrices B^TA_1B and B^TA_2B were given by the program as

$$B^TA_1B = \begin{pmatrix} 50.0000 & .0000 & .0000 & .0000 & .0000 & .0000 \\ .0000 & 29.9305 & .0000 & -1.2531 & 1.5738 & .0753 \\ .0000 & .0000 & 60.0000 & .0000 & .0000 & .0000 \\ .0000 & -1.2531 & .0000 & 39.7904 & -.6200 & .7728 \\ .0000 & 1.5738 & .0000 & -.6200 & 20.2584 & -.0351 \\ .0000 & .0753 & .0000 & .7728 & -.0351 & 10.0207 \end{pmatrix},$$

$$B^TA_2B = \begin{pmatrix} 20.0000 & .0000 & .0000 & .0000 & .0000 & .0000 \\ .0000 & 40.2336 & .0000 & -1.6232 & 3.1472 & 1.1790 \\ .0000 & .0000 & 10.0000 & .0000 & .0000 & .0000 \\ .0000 & -1.6232 & .0000 & 32.0055 & -1.0458 & 5.4272 \\ .0000 & 3.1472 & .0000 & -1.0458 & 59.2485 & .4738 \\ .0000 & 1.1790 & .0000 & 5.4272 & .4738 & 48.5123 \end{pmatrix}.$$

The FG-algorithm has clearly recovered the two common eigenvectors of A_1 and A_2. The four other columns of $B = B^{(4)}$ can be considered as "compromises" between eigenvectors of A_1 and A_2. Of course the order of the columns of B is not relevant; it is simply determined by the initial approximation used in the F-algorithm.

It is worth noting that the convergence is rather fast: after only two sweeps, the coefficients of B are already correct to two digits. This was typically also the case in statistical examples (see Flury (1984)), where the weights n_i are not necessarily equal. In none of these examples, more than five sweeps were needed to reach convergence.

The computation of the above example required .07 seconds of CPU time (not including input/output operations) on the CDC 170/855 computer of Indiana University.

REFERENCES

A. BASILEVSKY, (1983), *Applied Matrix Algebra in the Statistical Sciences*, North-Holland, New York.

B. N. FLURY, (1983), *Some relations between the comparison of covariance matrices and principal component analysis*, Computational Statistics and Data Analysis, 1, pp. 97–109.

——— (1984), *Common principal components in k groups*, J. Amer. Statist. Assoc, 79, pp. 892–898.

——— (1985), *The FG-algorithm*, accepted for publication in the algorithms' section of Applied Statistics.

G. H. GOLUB, AND C. F. VAN LOAN, (1983), *Matrix Computations*, Johns Hopkins Univ. Press, Baltimore.

A. E. HOERL, AND R. W. KENNARD, (1970), *Ridge regression: Biased estimation for nonorthogonal problems*, Technometrics, 12, pp. 55–67.

B. NOBLE, AND J. W. DANIEL, (1977), *Applied Linear Algebra*, Prentice-Hall. Englewood Cliffs, N.J.

B. N. PARLETT, (1980), *The Symmetric Eigenvalue Problem*, Prentice-Hall, Englewood Cliffs, NJ.

E. WEBER, (1974), *Einführung in die Faktorenanalyse*, Verlag Gustav Fischer, Stuttgart & New York.

SIAM J. SCI. STAT. COMPUT.
Vol. 7, No. 1, January 1986

014

RGSVD—AN ALGORITHM FOR COMPUTING THE KRONECKER STRUCTURE AND REDUCING SUBSPACES OF SINGULAR $A-\lambda B$ PENCILS*

BO KÅGSTRÖM†

Abstract. An algorithm (RGSVD) for computing the structure elements associated with the *Kronecker canonical form* (KCF) of a matrix pencil $A-\lambda B$, where A and B are complex m by n matrices, is presented. RGSVD is based on repeated generalized singular value decompositions (or more precisely cosine-sine decompositions of partitioned orthonormal matrices). It extracts the structures of the zero and/or the infinite eigenvalues together with the left (row) or right (column) minimal indices of $A-\lambda B$. By accumulating equivalence transformations, RGSVD also produces pairs of reducing subspaces associated with e.g. the zero structure and the right Kronecker indices.

Key words. matrix pencils (regular, singular), Kronecker structure, minimal indices, eigenvalues (finite, infinite), deflating subspace, reducing subspace, canonical form, linear system theory

1. Introduction. During the last few years there has been an increasing interest in the numerical treatment of general matrix pencils $A-\lambda B$ and the computation of the *Kronecker Canonical Form* (KCF). See [16] for a state of the art survey. The main reason for this interest is that in many applications, e.g. linear system theory [32], descriptor systems [23] and singular systems of differential equations [38], problems are modelled in terms of linear matrix pencils. Solvability issues around these problems like the existence and the unicity of a solution, the state of a system or an explicit solution of a problem can then easily be answered or obtained, respectively, by knowing the KCF of the underlying pencil.

In this paper we consider the problem of computing the structure elements associated with the KCF of a pencil $A-\lambda B$, where A and B are complex m by n matrices. These are the Jordan structures of finite and infinite eigenvalues and possibly minimal indices if $A-\lambda B$ is singular. For a complete definition of the Kronecker structure of $A-\lambda B$ see § 3. By definition (see e.g. [4]) a pencil $A-\lambda B$ is called regular if and only if A and B are square, $\det(A-\lambda B)\not\equiv 0$. In all other cases i.e., $m\neq n$, or $m=n$ but $\det(A-\lambda B)\equiv 0$, $A-\lambda B$ is called singular. Since the problem of computing Jordan structures is a subset of our problem, it inherits all the numerical difficulties of that problem (see [2]-[3], [6], [7], [12]-[14]). Moreover, the existence of minimal indices can make the problem even more ill-conditioned. Anyhow it is always possible to give a meaning to computed results in the following restricted way. Given $A-\lambda B$, we compute a KCF with Kronecker structure α, that exactly corresponds to a pencil $C-\lambda D$. We cannot guarantee that α is the true Kronecker structure of $A-\lambda B$, but we can give estimates of the distance from $A-\lambda B$ to $C-\lambda D$. The size of these estimates then validate the computed canonical reduction and α.

In this paper we present the *reiterating generalized singular value deflation* (RGSVD) algorithm for computing the Kronecker structure α of a singular pencil. The RGSVD algorithm is based on a reduction theorem that computes the Jordan structure α_0 of the zero eigenvalue and the right (column) minimal indices α_r. At the same time we obtain pairs of reducing subspaces [34], which is a generalization of deflating subspaces [24] to the singular case. The reduction is based on a finite sequence of column range-nullspace separations, in terms of generalized singular value decompositions [21], [26]-[27], [29]-[30], [36] of matrix pairs. We prove a relationship between

* Received by the editors January 10, 1984, and in revised form November 12, 1984.
† Institute of Information Processing, University of Umeå, S-901 87 Umeå, Sweden.

the problem of computing α_0 and α_* and the generalized singular value decomposition (GSVD). This relationship is similar to the one that exists between the Jordan structure problem of $A-\lambda I$ and the standard singular value decomposition (SVD) (see [6], [12]). As a consequence it is possible to express exactly the distance from $A-\lambda B$ to $C-\lambda D$, with the computed α_0 and α_*, in terms of deleted generalized singular values.

Before we go into details further, we outline the content of the rest of this paper. In § 2 we review the problem of finding the Jordan structure of an eigenvalue λ. In § 3 we collect the basic algebraic theory of singular pencils, for example the KCF, and we introduce notation. Section 4 illustrates the geometric properties of a singular pencil in canonical form, that form the basis of RGSVD.

In [18] we show how to extract geometric information concerning the row and column nullspaces of $A-\lambda B$ from the GSVD of the matrix pair (A, B), and in § 5 we summarize some facts and results from [18] that are useful in coming sections.

Section 6 is devoted to algorithms for computing the Kronecker structure. The reduction theorem is stated and proved in § 6.1. The RGSVD algorithm is presented in § 6.2 and the RGQZD algorithm in 6.3 which is based on a generalized QZ decomposition [18]. Finally in § 6.4 we discuss other approaches taken by Kublanovskaya [9]–[11] and Van Dooren [31]. In § 7 we discuss the concept of pairs of reducing subspaces and relate it to the reduction theorems in §§ 6.1 and 6.3. In § 8 we report results from a Matlab [19] program. We study one regular and one singular pencil. Finally in § 9 we make some conclusions and outline directions for future work.

2. The zero structure of $A-\lambda I$. Before discussing the general matrix pencil problem $A-\lambda B$ we review the problem of finding the *Jordan structure* of an eigenvalue λ corresponding to the pencil $A-\lambda I$. It is well known that the *Jordan normal form* (JNF) of A is not a continuous function of the matrix elements and therefore it can be very hard to deal with numerically (see [6], [7], [12]–[14], [3]). When we apply our JNF-algorithm [12]–[13] to A, we always compute a JNF such that

$$(A+E)S=SJ \tag{2.1}$$

where J is a direct sum of unnormalized *Jordan blocks* $J_k(\lambda)$ e.g.

$$J_3(\lambda)=\begin{bmatrix}\lambda & r_1 & 0\\ 0 & \lambda & r_2\\ 0 & 0 & \lambda\end{bmatrix}. \tag{2.2}$$

If $\|E\|$ in (2.1) is of order $\|A\|\cdot$ machep, where machep is the relative machine precision, then we say that we have computed a satisfactory JNF. In theory there might exist a matrix B closer to A that has a different JNF. However we cannot expect anything better when working in finite precision arithmetic.

If we only are interested in the Jordan structure of one eigenvalue λ, we can rely on unitary similarity transformations. The following theorem is originally due to Kublanovskaya [8].

THEOREM 2.1. *Let $A\in\mathbb{C}^{n\times n}$ and $\lambda=0$ be an eigenvalue of multiplicity t of A with Jordan structure*

$$\alpha_0=(\delta_1, \delta_2, \cdots, \delta_h) \tag{2.3}$$

where $\delta_k=n_k-n_{k+1}$ is the number of elementary divisors of grade k (= the number of $J_k(0)$-blocks), h is the maximal height of the Jordan chains and

$$n_k=\dim \mathcal{N}(A^k)-\dim \mathcal{N}(A^{k-1}). \tag{2.4}$$

Then there exists an unitary matrix $P \in \mathbb{C}^{n\times n}$ *such that*

$$P^H A P = \begin{bmatrix} A_0 & A_{12} \\ 0 & A_r \end{bmatrix}, \quad P^H I P = \begin{bmatrix} I_t & 0 \\ 0 & I_{n-t} \end{bmatrix} \tag{2.5}$$

where

$$A_0 = \begin{bmatrix} 0 & M_1 & & & & \\ & 0 & M_2 & & & \\ & & 0 & M_3 & & \\ & & & 0 & \ddots & \\ & & & & \ddots & M_{h-1} \\ & & & & & 0 \end{bmatrix} \tag{2.6}$$

$\left.\begin{matrix}\\ \\ \\ \\ \\ \\ \end{matrix}\right\} = \sum_{k=1}^{h} n_k$

$n_1 \geqq n_2 \geqq n_3 \cdots \qquad \geqq n_h$

and $\det A_r \neq 0$ *i.e. an empty intersection of the spectrums of* A_0 *and* A_r.

If we partition P as

$$P = [\underbrace{P_0}_{t} \mid \underbrace{P_r}_{n-t}], \tag{2.7a}$$

$$P_0 = [\underbrace{P_{01}}_{n_1} \quad \underbrace{P_{02}}_{n_2} \quad \cdots \quad \underbrace{P_{0h}}_{n_h}] \tag{2.7b}$$

then $\operatorname{span}\{P_0\} = \mathcal{N}(A^h)$ *is the maximal invariant subspace corresponding to the eigenvalue* $\lambda = 0$. *Further, the* n_k *columns of* P_{0k} *span the restricted nullspace* $\mathcal{N}(A^k)\backslash\mathcal{N}(A^{k-1})$, *i.e. the space spanned by the principal vectors of grade k.*

Proof. For a complete proof we refer the reader to [6] or [12]. A finite sequence of range-nullspace separations deflates the singularity of A and is done by utilizing the singular value decomposition (SVD). Here we give the first step of the deflation process since the technique will later be generalized to the matrix pencil $A - \lambda B$.

Let

$$A - \lambda I = U \Sigma V^H, \quad \Sigma = \operatorname{diag}\{\sigma_i\}, \quad 0 \leqq \sigma_1 \leqq \cdots \leqq \sigma_n \tag{2.8}$$

be the SVD of $A - \lambda I$ ($\lambda = 0$ in this case). Note that we order the singular values in increasing sequence.

Since $n_1 = \dim \mathcal{N}(A)$, the first n_1 singular values are zero which implies that

$$(A - \lambda I) V = U\Sigma = [\underbrace{0}_{n_1} \mid \text{▨}] \tag{2.9}$$

i.e. the first n_1 columns of V are eigenvectors and

$$V^H U \Sigma \equiv V^H (A - \lambda I) V = \begin{bmatrix} 0 & M_1 \\ 0 & A^{(1)} \end{bmatrix}. \tag{2.10}$$

(n_1 | $n - n_1$)

Apply the same operations on $A^{(1)}$. The process continues until step h where $A^{(h)} (= A_r)$ is nonsingular. The product of all transformation matrices gives the final P. □

In practice the Jordan structure α_0 (2.3) is not known in advance. At each step the numerical nullity of $A^{(k)}$ is determined from its singular values and a deflation tolerance. The perturbation matrix E in (2.1) accumulate deleted singular values from the deflation process. See Kågström [14] for an explicit expression of E and Kågström-Ruhe [12]-[13] for further interpretation and assessment of a computed JNF.

3. The Kronecker structure of $A-\lambda B$. If $A, B \in \mathbb{C}^{n\times n}$ and $\det(A-\lambda B) \equiv 0$ for all $\lambda \in \mathbb{C}$, i.e. every possible linear combination of A and B is singular, then $A-\lambda B$ is called a *singular pencil.* The simplest examples occur when A and B have a common row or column nullspace, but there are also more complicated singular pencils whose structure, described by minimal indices, is given by the *Kronecker canonical form* (KCF) of $A-\lambda B$. Rectangular pencils $A-\lambda B$, $A, B \in \mathbb{C}^{m\times n}$ are always singular. By a *strictly equivalent transformation* of $A-\lambda B$ we mean

$$P^{-1}(A-\lambda B)Q = \tilde{A}-\lambda\tilde{B} \tag{3.1}$$

where $P \in \mathbb{C}^{m\times m}$ and $Q \in \mathbb{C}^{n\times n}$ are constant and nonsingular. In the KCF (see e.g. Gantmacher [4]) P and Q are chosen so $\tilde{A}-\lambda\tilde{B}$ has the form

$$P^{-1}(A-\lambda B)Q = \operatorname{diag}\{J-\lambda I, I-\lambda N, L_{\varepsilon_1}, L_{\varepsilon_2}, \cdots, L_{\varepsilon_p}, L_{\eta_1}^T, L_{\eta_2}^T, \cdots, L_{\eta_q}^T\} \tag{3.2}$$

where

- J corresponds to the finite eigenvalues (including zero-eigenvalues) of $A-\lambda B$. N is nilpotent and corresponds to the infinite eigenvalue of $A-\lambda B$. Both N and J are direct sums of k by k Jordan blocks N_k and $J_k(\lambda_i)$ respectively.
- L_ε and L_η^T are $\varepsilon\times(\varepsilon+1)$ and $(\eta+1)\times\eta$ respectively, and bidiagonal. For example

$$L_3 = \begin{bmatrix} -\lambda & 1 & 0 & 0 \\ 0 & -\lambda & 1 & 0 \\ 0 & 0 & -\lambda & 1 \end{bmatrix}.$$

The ε_i and η_i are called the *minimal right* (*column*) and *left* (*row*) *indices* of $A-\lambda B$ respectively. Here we assume that $0 \leqq \varepsilon_1 \leqq \varepsilon_2 \leqq \cdots \leqq \varepsilon_p$ and $0 \leqq \eta_1 \leqq \eta_2 \leqq \cdots \leqq \eta_q$.

Notice that the blocks L_ε and/or L_η^T will only appear if $A-\lambda B$ is singular and they expose the singularity of $A-\lambda B$ in the following way. For L_ε there exists a polynomial column vector such that

$$L_\varepsilon \underbrace{[1\ \lambda\ \cdots\ \lambda^\varepsilon]}_{\varepsilon+1}^T = \underbrace{[0, \cdots, 0]}_{\varepsilon}^T$$

while for L_η^T there exists a polynomial row vector such that

$$\underbrace{[1\ \lambda\ \cdots\ \lambda^\eta]}_{\eta+1} L_\eta^T = \underbrace{[0, \cdots, 0]}_{\eta}.$$

For a *regular pencil* $A-\lambda B$ $\det(A-\lambda B)$ is not identically zero and the KCF simplifies to the *Weierstrass-Kronecker canonical form* (W-KCF):

$$\operatorname{diag}\{J-\lambda I, I-\lambda N\}. \tag{3.3}$$

This canonical form consists of finite and infinite eigenvalues but no minimal indices. If $A-\lambda B$ is of the form $A-\lambda I$, i.e., a standard eigenvalue problem the KCF reduces to the JNF, $\operatorname{diag}\{J-\lambda I\}$.

Assume that the singular pencil $A-\lambda B$ has eigenvalues $0\equiv\lambda_0, \lambda_1, \lambda_2, \cdots, \lambda_p$, and $\lambda_\infty=\infty$ where $\lambda_1, \cdots, \lambda_p$ are nonzero and distinct. Let $t_k \geqq 1$ denote the *algebraic multiplicity*, and let $\delta_i^{(k)}$ denote the number of *elementary divisors* of grade i at λ_k, h_k the maximal height of a Jordan chain at λ_k. Then

$$\alpha_k=(\delta_1^{(k)}, \delta_2^{(k)}, \cdots, \delta_{h_k}^{(k)}) \tag{3.4}$$

and denotes the *Jordan structure of the eigenvalue* λ_k for $k=0, 1, \cdots, p$ and ∞. Further, let ℓ_i and r_i denote *the number of left (row) and right (column) minimal indices of grade* i and let v and u be the maximal order of the blocks L_ε and L_η^T respectively. Collecting the left and right minimal indices in

$$\alpha_\ell=(\ell_0, \ell_1, \cdots, \ell_u), \tag{3.5a}$$

$$\alpha_r=(r_0, r_1, \cdots, r_v) \tag{3.5b}$$

and the structure indices α_k we form the multiindex

$$\alpha=(\alpha_0, \alpha_1, \cdots, \alpha_p, \alpha_\infty, \alpha_\ell, \alpha_r). \tag{3.6}$$

We refer to this as the *Kronecker structure* of $A-\lambda B$. We denote the set of pencils $A-\lambda B$ with structure α (3.6) by $\mathscr{E}_\alpha$. From (3.5a-b) the total number of L_k^T-blocks and L_k-blocks, respectively, can be expressed by

$$\tilde{\ell}=\sum_{i=0}^{u}\ell_i \tag{3.7a}$$

and

$$\tilde{r}=\sum_{i=0}^{v} r_i. \tag{3.7b}$$

4. A singular pencil in canonical form. Consider an 8 ($=m$) by 12 ($=n$) singular pencil $A-\lambda B$ with Kronecker structure $\alpha=(\alpha_0, \alpha_r)$ where $\alpha_0=(1, 0, 1)$ and $\alpha_r=(2, 1, 0, 1)$,

$$A-\lambda B=\left[\begin{array}{c|cccc|cccc|cccc} -\lambda & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \hline 0 & -\lambda & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -\lambda & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -\lambda & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & 0 & 0 & -\lambda & 1 & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -\lambda & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -\lambda & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -\lambda & 1 \end{array}\right] \tag{4.1}$$

i.e. its KCF consists of two Jordan blocks $J_1(0)$, $J_3(0)$ and four Kronecker blocks L_0, L_0, L_1 and L_3. Notice that the 0 by 1 blocks L_0 only contribute to the column dimension of $A-\lambda B$. In the same way L_0^T-blocks only contribute to the row dimension.

A finite structure-block $J_k(\lambda)-\lambda I$ can be recognized from either a deficient column or row rank in the A-part of the KCF. The L_k-blocks have a deficient column rank both in the A- and B-parts of the KCF. Correspondingly, an infinite structure-block is recognized from either a deficient column or row rank in the B-part and a L_k^T-block has a deficient row rank both in the A- and B-parts.

The deficiencies in the column ranks are further exposed by pivoting A and B so that all zero-columns of A are the leading ones:

(4.2a) $A' =$

$$\left[\begin{array}{cccccc|ccc|ccc}
0&0&0&0&0&0&1&0&0&0&0&0\\
0&0&0&0&0&0&0&1&0&0&0&0\\
0&0&0&0&0&0&0&0&1&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0\\
\hline
0&0&0&0&0&0&0&0&0&1&0&0\\
0&0&0&0&0&0&0&0&0&0&1&0\\
\hline
0&0&0&0&0&0&0&0&0&0&0&1\\
0&0&0&0&0&0&0&0&0&0&0&0
\end{array}\right]
\begin{array}{l}\left.\begin{array}{l}\\\\\\\end{array}\right\} r_1 = 4\\ \\ \\ \\ \\ \\ \\ \\ \end{array}$$

$$\underbrace{\qquad\qquad}_{n_1=6}$$

(4.2b) $B' =$

$$\left[\begin{array}{cccccc|ccc|ccc}
0&0&1&0&0&0&0&0&0&0&0&0\\
0&0&0&1&0&0&0&0&0&0&0&0\\
0&0&0&0&1&0&0&0&0&0&0&0\\
0&0&0&0&0&1&0&0&0&0&0&0\\
\hline
0&0&0&0&0&0&0&1&0&0&0&0\\
0&0&0&0&0&0&0&0&1&0&0&0\\
\hline
0&0&0&0&0&0&0&0&0&1&0&0\\
0&0&0&0&0&0&0&0&0&0&1&0
\end{array}\right]
\begin{array}{l}\left.\begin{array}{l}\\\\\\\end{array}\right\} r_1 = 4\\ \\ \\ \\ \\ \\ \\ \\ \end{array}$$

The transpositions made can be summarized in the following. We retrieve the L_3-block from columns 4, 8, 10, 12 and rows 2, 5, 7 of $A'-\lambda B'$ (4.2a-b) and the $J_3(0)$-block from columns 5, 9, 11 and rows 3, 6, 8. L_1 forms columns 3, 7 and row 1. $J_1(0)$ forms column 6 and row 4. Finally the two L_0-blocks form columns 1, 2.

From (4.2a-b) we deduce that the column nullity n_1 of A is 6 ($=\dim \mathcal{N}_c(A)$) and that A and B have a common nullspace of dimension 2. Here r_1 denotes the dimension of $\mathcal{N}_c(A)$ which is not in common to $\mathcal{N}_c(B)$ ($=\dim \mathcal{N}_c(A)\backslash \mathcal{N}_c(A)\cap \mathcal{N}_c(B)$). Generally

$$n_1 \geqq r_1 \geqq 0 \tag{4.3}$$

and the number of L_0-blocks, $\varepsilon_0 = n_1 - r_1$. Further $A-\lambda B$ has r_1 structure-blocks $J_k(0)$ or L_k of order $\geqq 1$. In our example $\varepsilon_0 = 2$ and $r_1 = 4$.

As in the case $A-\lambda I$ (see § 2) we can retrieve further information about the structure-blocks by repeatedly deflating the pencil. In our example we deflate the strict equivalent pencil $A'-\lambda B'$ (4.2a-b) at the r_1th row and the n_1th column:

(4.4a) $A^{(1)} := A'(5:8, 7:12) =$

$$\left[\begin{array}{ccc|ccc}
0&0&0&1&0&0\\
0&0&0&0&1&0\\
\hline
0&0&0&0&0&1\\
0&0&0&0&0&0
\end{array}\right]
\begin{array}{l}\left.\begin{array}{l}\\\\\end{array}\right\} r_2\\ \\ \\ \\ \end{array}$$

$$\underbrace{\qquad}_{n_2}$$

(4.4b) $B^{(1)} := B'(5:8, 7:12) =$

$$\left[\begin{array}{ccc|ccc}
0&1&0&0&0&0\\
0&0&1&0&0&0\\
\hline
0&0&0&1&0&0\\
0&0&0&0&1&0
\end{array}\right]
\begin{array}{l}\left.\begin{array}{l}\\\\\end{array}\right\} r_2\\ \\ \\ \\ \end{array}$$

Let $A^{(k-1)}-\lambda B^{(k-1)}$ denote the deflated pencil from step $k-1$ which will be studied next, and let $n_k=\dim \mathcal{N}_c(A^{(k-1)})$ and $n_k-r_k=\dim \mathcal{N}_c(A^{(k-1)})\cap \mathcal{N}_c(B^{(k-1)})$. By repeating the discussion above we see that $n_2=3$ and $r_2=2$. In general these column nullities expose the Kronecker structure $\alpha=(\alpha_0, \alpha_\infty)$ in the following way. For a proof see [17], [31] or [38].

THEOREM 4.1.

(4.5a) $$n_k-r_k = \textit{the number of } L_{k-1}\textit{-blocks of order } (k-1)\times k,$$

(4.5b) $$r_k-n_{k+1} = \textit{the number of } J_k(0)\textit{-blocks of order } k\times k.$$

Note that in the regular case $n_k=r_k$ for all k.

After the next deflation at the r_2th row and the n_2th column of $A^{(1)}-\lambda B^{(1)}$ we are left with:

(4.6a) $$A^{(2)}:=A^{(1)}(3:4,4:6)=\left[\begin{array}{cc|c} 0 & 0 & 1 \\ \underbrace{0 \quad 0}_{n_3} & & 0 \end{array}\right]\Big\}r_3,$$

(4.6b) $$B^{(2)}:=B^{(1)}(3:4,4:6)=\left[\begin{array}{cc|c} \overbrace{1 \quad 0}^{n_3} & & 0 \\ 0 & 1 & 0 \end{array}\right]\Big\}r_3.$$

We have that $n_3=2$ and $\mathcal{N}_c(A^{(2)})\cap\mathcal{N}_c(B^{(2)})=0$ so $r_3=n_3$. After deflating at the r_3th row and n_3th column of $A^{(2)}-\lambda B^{(2)}$ we are left with a 0 by 1 pencil $A^{(3)}-\lambda B^{(3)}$, i.e. a pencil that has no row but one column. The deflations so far give that $m=r_1+r_2+r_3=8$ and $n_1+n_2+n_3=11$. Since $n=12$ we define $n_4=1$ and $r_4=0$ so

(4.7) $$\begin{aligned} &n_1(=6)\geqq r_1(=4)\geqq n_2(=3)\geqq r_2(=2)\geqq n_3(=2)\geqq r_3(=2)\\ &n_4(=1)\geqq r_4(=0)\geqq 0, \end{aligned}$$

which also holds generally (see § 6).

In the coming sections we will show how to compute the n_k's and r_k's and the strictly equivalent pencils $A^{(k)}-\lambda B^{(k)}$ giving a general algorithm for computing the Kronecker structure α of $A-\lambda B$ (see (3.4)-(3.6)) and reducing subspaces (see § 7).

5. The generalized singular value decomposition (GSVD). From the introductory example in § 4 we observed that the column and row nullities of A and B, and the possible common nullspaces give information about the Kronecker structure of $A-\lambda B$. In [18] we show how to extract the significant information concerning these nullspaces from the GSVD of the matrix pair (A, B). Here we summarize some facts and results from [18] that are useful in coming sections.

5.1. Algebraic formulation. In the following theorem we formulate the GSVD of the m by n matrices A and B when $m\geqq n$. The proof (see [18]) is very similar to the one in Paige and Saunders [21]. We use a SVD and a QR-decomposition instead of a QR-decomposition and a LQ-decomposition respectively. This alternative derivation of the GSVD gives us an algorithm for computing the transformation matrices (V, X), used in an equivalence transformation $V^H(A-\lambda B)X$. The case $m<n$ is discussed in [18].

THEOREM 5.1. *Let $A, B\in\mathbb{C}^{m\times n}$ where $m\geqq n$ and*

(5.1) $$r=\operatorname{rank}\left[\begin{array}{c} A \\ \hline B \end{array}\right]\leqq n.$$

Then there exist unitary matrices $U, V \in \mathbb{C}^{m\times m}$ *and a nonsingular matrix* $X \in \mathbb{C}^{n\times n}$ *such that*

$$\left[\begin{array}{c|c} U^H & 0 \\ \hline 0 & V^H \end{array}\right]\begin{bmatrix} A \\ B \end{bmatrix} X = \begin{bmatrix} D_A \\ \hline \underbrace{D_B}_{n} \end{bmatrix}\begin{matrix}\}m \\ \}m\end{matrix} \tag{5.2}$$

where

$$D_A = [0 \mid \Sigma_A], \qquad \Sigma_A = \begin{bmatrix} C \\ \hline 0 \end{bmatrix}\begin{matrix}\}r \\ \}m-r\end{matrix} \tag{5.3a}$$

$$D_B = [\underbrace{0}_{n-r} \mid \underbrace{\Sigma_B}_{r}], \qquad \Sigma_B = \begin{bmatrix} S \\ \hline 0 \end{bmatrix}\begin{matrix}\}r \\ \}m-r\end{matrix} \tag{5.3b}$$

and

$$C = \operatorname{diag}\{c_1, c_2, \cdots, c_r\}, \qquad 0 \leqq c_1 \leqq c_2 \leqq \cdots \leqq c_r \leqq 1, \tag{5.4a}$$

$$S = \operatorname{diag}\{s_1, s_2, \cdots, s_r\}, \qquad 1 \geqq s_1 \geqq s_2 \geqq \cdots \geqq s_r \geqq 0 \tag{5.4b}$$

satisfying

$$C^2 + S^2 = I. \tag{5.5}$$

Since we only need (V, X) or (U, X) in an equivalence transformation of $A - \lambda B$, we do not come into the numerical difficulties one encounters, when trying to compute the complete GSVD ([21], [26]-[27], [37]). Sun [29]-[30] has made a perturbation analysis of the GSVD-problem, and Paige [22] derives an improved version of one of his results using straightforward matrix ideas.

5.2. A geometric interpretation of the GSVD. In § 4 we introduced the notations

$$n_1 = \dim \mathcal{N}_c(A), \tag{5.6a}$$

$$n_1 - r_1 = \dim \mathcal{N}_c(A) \cap \mathcal{N}_c(B) \tag{5.6b}$$

where r_1 is the dimension of $\mathcal{N}_c(A)$ that is not in common to $\mathcal{N}_c(B)$ i.e.

$$\begin{aligned} r_1 &= \dim \mathcal{N}_c(A) \backslash \mathcal{N}_c(A) \cap \mathcal{N}_c(B) \\ &= \dim \mathcal{N}_c(A) - \dim \mathcal{N}_c(A) \cap \mathcal{N}_c(B). \end{aligned} \tag{5.6c}$$

By knowing the GSVD of A and B [see (5.2)-(5.5)] we can write

$$A = U \cdot \left[\underbrace{0}_{n-r} \mid \underbrace{\begin{bmatrix} C \\ \hline 0 \end{bmatrix}}_{r}\right]\begin{matrix}\}m-r\end{matrix} \cdot X^{-1} \tag{5.7a}$$

and

$$B = V \cdot \overbrace{\left[0 \mid \begin{bmatrix} S \\ \hline 0 \end{bmatrix}\right]}^{n-r \quad r}\begin{matrix}\}m-r\end{matrix} \cdot X^{-1}. \tag{5.7b}$$

From our assumptions $c_1 = c_2 = \cdots = c_{r_1} = 0$ and therefore the corresponding s_i will be ones. By multiplying A and B in (5.7a-b) with X from the right we see that the columns x_i of X span the different column nullspaces:

$$\mathcal{N}_c(A) = \operatorname{span}\{x_1, \cdots, x_{n_1}\}, \tag{5.8a}$$

$$\mathcal{N}_c(A) \cap \mathcal{N}_c(B) = \operatorname{span}\{x_1, \cdots, x_{n_1 - r_1}\}, \tag{5.8b}$$

(5.8c) $$\mathcal{N}_c(A)\backslash\mathcal{N}_c(A)\cap\mathcal{N}_c(B)=\text{span}\{x_{n_1-r_1+1},\cdots,x_{n_1}\}.$$

Analogously the part of $\mathcal{N}_c(B)$ that is not in common to $\mathcal{N}_c(A)$ corresponds to the number of s_i equal to zero (say p_1) and

(5.8d) $$\mathcal{N}_c(B)=\mathcal{N}_c(A)\cap\mathcal{N}_c(B)+\text{span}\{x_{n-p_1+1},\cdots,x_n\}.$$

If we sort C and S in the opposite order, we instead get a basis of $\mathcal{N}_c(B)$ from the $(n-r)+p_1$ first consecutive columns of X. Row nullities and bases for the row nullspaces of A and B, $\mathcal{N}_r(A)$ and $\mathcal{N}_r(B)$ respectively, can also be obtained from the GSVD. Since A and $B\in\mathbb{C}^{m\times n}$ we have that

(5.9a) $$\dim\mathcal{N}_r(A)=(m-r)+r_1,$$

(5.9b) $$\dim\mathcal{N}_r(B)=(m-r)+p_1,$$

(5.9c) $$\dim\mathcal{N}_r(A)\cap\mathcal{N}_r(B)=m-r$$

where as before r_1 is the number of c_i equal to zero and p_1 is the number of s_i equal to zero. By premultiplying A (5.7a) with U^H and B (5.7b) with V^H we get orthonormal bases for $\mathcal{N}_r(A)$ and $\mathcal{N}_r(B)$ from the columns of U and V:

(5.10a) $$\mathcal{N}_r(A)=\text{span}\{u_1,\cdots,u_{r_1},u_{m-r+1},\cdots,u_m\},$$

(5.10b) $$\mathcal{N}_r(B)=\text{span}\{v_{m-(r_1+p_1)+1},\cdots,v_m\}.$$

Evidently we cannot get a basis for $\mathcal{N}_r(A)\cap\mathcal{N}_r(B)$ from the GSVD of (A, B). Since $\mathcal{N}_r(A)=\mathcal{N}_c(A^H)$ and $\mathcal{N}_r(B)=\mathcal{N}_c(B^H)$ we can compute the GSVD of (A^H, B^H) and obtain bases for the row nullspaces from (5.8a–d). The values of r_1 and p_1 will still be the same but $n_1=(m-r)+r_1$. In the following sections r_1 and n_1 denote column nullities and are defined by (5.6a–c).

By considering $A\in\mathbb{C}^{m\times n}$ as a mapping from $\mathbb{C}^n$ to $\mathbb{C}^m$ we can express $\mathbb{C}^n$ and $\mathbb{C}^m$ as direct sums of the range and nullspaces of A and A^H:

$$\mathbb{C}^m=\mathcal{R}(A)\oplus\mathcal{N}(A^H);\qquad\mathbb{C}^n=\mathcal{R}(A^H)\oplus\mathcal{N}(A).$$

In our notation $\mathcal{N}_c(A)=\mathcal{N}(A)$ and $\mathcal{N}_r(A)=\mathcal{N}(A^H)$. We find the notation of column and row nullspaces natural for our problem, since these nullities of A and B give us information about the column and row minimal indices of $A-\lambda B$.

6. Algorithms for computing the Kronecker structure. The RGSVD algorithm is based on a reduction theorem of a singular pencil, that is stated and proved in § 6.1. The theorem is a natural generalization of Theorem 2.1 to singular pencils. Apart from the Jordan structure α_0 (2.3), the reduced pencil displays α_r (3.5), the right (column) minimal indices of the Kronecker structure. For the $A-\lambda I$ probem a finite sequence of range-nullspace separations, in terms of SVD's, deflates the singularity. Here a finite sequence of column range-nullspace separations of matrix pairs, in terms of GSVD's, reduces the singularity. Several arguments used in the coming proof are in the style of Wilkinson [39]-[40], where he elegantly derives the KCF starting from a singular system of differential equations

(6.1) $$B\dot{x}(t)=Ax(t)+f(t).$$

In § 6.2 we present the RGSVD-algorithm. We also make use of the reduction theorem to compute α_∞, the Jordan structure of the infinite eigenvalue, and α_ℓ (3.5a), the left (row) minimal indices. The rest of the pencil is the regular part corresponding to the nonzero and finite eigenvalues of $A-\lambda B$. In fact this regular part is almost in standard

form and once we know the eigenvalues we can make use of Theorem 2.1 to compute its Jordan structure.

In § 6.3 we outline the RGQZD algorithm which, instead of GSVD, make use of a generalized QZ decomposition [18], giving a unitary equivalence transformation of $A-\lambda B$. In § 6.4 we make some comparisons with other approaches and algorithms, presented by Kublanovskaya and Van Dooren.

6.1. The reduction theorem.

THEOREM 6.1. *Let A, $B \in \mathbb{C}^{m\times n}$, m and n arbitrary, and $A-\lambda B$ be a singular pencil. Let $\lambda = 0$ be an eigenvalue of multiplicity t of $A-\lambda B$ with Jordan structure α_0 and let the pencil have the right minimal indices α_i,* (see (2.3)-(2.4) and (3.5b)).

Then there exist a unitary matrix $V \in \mathbb{C}^{m\times m}$ and a nonsingular matrix $X \in \mathbb{C}^{n\times n}$ such that

$$V^H A X = \left[\begin{array}{c|c} A_{0_z} & A_{12} \\ \hline 0 & A_r \end{array}\right] \tag{6.2a}$$

and

$$V^H B X = \left[\begin{array}{c|c} B_{0_z} & 0 \\ \hline 0 & B_r \end{array}\right] \tag{6.2b}$$

where

$$A_{0_z} = \begin{bmatrix} 0 & M_1 & & & \\ & 0 & M_2 & & \\ & & 0 & M_3 & \\ & & & 0 & M_4 \\ & & & & \ddots \end{bmatrix} \begin{matrix} r_1 \\ r_2 \\ r_3 \\ \\ \end{matrix}, \tag{6.3}$$

$$n_1 \geqq n_2 \geqq n_3 \geqq \cdots$$

$$B_{0_z} = \begin{bmatrix} 0 & I_{r_1} & & & \\ & 0 & I_{r_2} & & \\ & & 0 & I_{r_3} & \\ & & & & \ddots \end{bmatrix} \begin{matrix} r_1 \\ r_2 \\ r_3 \\ \\ \end{matrix}. \tag{6.4}$$

I_{r_k} denotes the unit matrix of order r_k. The block indices n_k and r_k of (6.3)-(6.4) *satisfy*

the inequalities

$$n_1 \geqq r_1 \geqq n_2 \geqq r_2 \geqq n_3 \geqq \cdots \geqq n_s \geqq r_s = n_{s+1} = 0 \tag{6.5}$$

and display the structure indices α_0 *and* α_* *as*

$$\alpha_0 = (\delta_1, \delta_2, \cdots, \delta_h) \quad \textit{with } \delta_k = r_k - n_{k+1} \tag{6.6}$$

and

$$\alpha_* = (\imath_0, \imath_1, \cdots, \imath_v) \quad \textit{with } \imath_{k-1} = n_k - r_k \tag{6.7}$$

where δ_k *is the number of* $J_k(0)$*-blocks and* $\imath_{k-1}$ *is the number of* L_{k-1}*-blocks.*

A_r *and* A_{12} *are full matrix blocks and* A_r *is of full column rank*; B_r *has only one diagonal with positive elements starting at the top left corner.*

The pencil $A_r - \lambda B_r$ might be singular and, using the notation in § 3, contains the finite eigenvalues $\lambda_1, \cdots, \lambda_p$ and λ_∞ together with α_ℓ, the left (row) minimal indices of $A - \lambda B$.

On purpose the lower rightmost blocks of A_{0*} and B_{0*} are omitted. There are two cases that will be illustrated in the proof.

Proof. The proof is constructive and based on repeated use of the GSVD on matrix pairs $(A^{(k)}, B^{(k)})$, each pair originating from a deflation of the previous pencil $A^{(k-1)} - \lambda B^{(k-1)}$. The first GSVD is made on $(A^{(0)}, B^{(0)}) = (A, B)$. In step k $(\geqq 1)$ the indices n_k and r_k are determined from the GSVD of $(A^{(k-1)}, B^{(k-1)})$, where

$$n_1 = \dim \mathcal{N}_c(A); \qquad n_1 - r_1 = \dim \mathcal{N}_c(A) \cap \mathcal{N}_c(B) \tag{6.8}$$

and

$$n_k = \dim \mathcal{N}_c(A^{(k-1)}), \qquad k = 2, 3, \cdots, \tag{6.9a}$$

$$n_k - r_k = \dim \mathcal{N}_c(A^{(k-1)}) \cap \mathcal{N}_c(B^{(k-1)}). \tag{6.9b}$$

Assume that $m \geqq n$. The reduction works as well for $m < n$, but the GSVD looks different (see [18]). The geometric interpretation of GSVD was introduced in § 5.2. See also § 4, where the case $m < n$ and the geometric indices n_k and r_k are discussed and illustrated.

We use the $\hat{V}_k$ and $\hat{X}_k$ from the GSVD of $(A^{(k-1)}, B^{(k-1)})$ as transformation matrices and get (see (5.7a-b))

$$A^{(k-1)} - \lambda B^{(k-1)} = \hat{V}_k \left(\left[\begin{array}{c|c} \underbrace{\begin{matrix} 0 \\ 0 \end{matrix}}_{n_k} & \begin{matrix} M_k \\ A^{(k)} \end{matrix} \end{array} \right]\Big\}^{r_k} - \lambda \left[\begin{array}{c|c} \underbrace{\begin{matrix} 0 & I_{r_k} \\ 0 & \end{matrix}}_{n_k} & \begin{matrix} 0 \\ B^{(k)} \end{matrix} \end{array} \right] \right) \hat{X}_k^{-1}. \tag{6.10}$$

Note that $A^{(k-1)}$ is left multiplied by $\hat{V}_k^H$ which makes $B^{(k)}$ diagonal,

$$B^{(k)} = \begin{bmatrix} S^{(k)} \\ 0 \end{bmatrix}, \qquad S^{(k)} = \operatorname{diag}\{s_{r_k+1}^{(k)}, \cdots, s_r^{(k)}\} \tag{6.11}$$

where $1 > s_{r_k+1}^{(k)} \geqq \cdots \geqq s_r^{(k)} \geqq 0$; not by $\hat{U}_k^H$ which would have diagonalized $A^{(k-1)}$ and that is why M_k and $A^{(k)}$ are full matrices.

The reduction continues until a step s where either $n_s = 0$ or $n_s \neq 0$ but $r_s = 0$. Without loss of generality we treat the case $s = 4$ and work on both cases. Introduce

$$\tilde{r}_k = \sum_{i=1}^{k} r_i, \qquad \tilde{n}_k = \sum_{i=1}^{k} n_i \tag{6.12}$$

and the augmented matrices $V_k \in \mathbb{C}^{m\times m}$, $X_k \in \mathbb{C}^{n\times n}$ where

$$V_k = \begin{bmatrix} I_{\tilde{r}_{k-1}} & 0 \\ 0 & \hat{V}_k \end{bmatrix}, \qquad X_k = \begin{bmatrix} I_{\tilde{n}_{k-1}} & 0 \\ 0 & \hat{X}_k \end{bmatrix}. \tag{6.13}$$

After 3 steps we have

$$A - \lambda B = V_1 V_2 V_3 \left(\begin{bmatrix} 0 & \hat{A}_{12} & \hat{A}_{13} & A_{14} \\ & 0 & \hat{A}_{23} & A_{24} \\ & & 0 & A_{34} \\ & & & A^{(3)} \end{bmatrix} \begin{matrix} r_1 \\ r_2 \\ r_3 \\ m-\tilde{r}_3 \end{matrix} \right. \tag{6.14}$$

$$\begin{matrix} n_1 & n_2 & n_3 & n-\tilde{n}_3 \end{matrix}$$

$$\left. -\lambda \begin{bmatrix} 0 & I_{r_1} & & \\ & 0 & I_{r_2} & \\ & & 0 & I_{r_3} \\ & & & B^{(3)} \end{bmatrix} \right) X_3^{-1} X_2^{-1} X_1^{-1}.$$

We notice that $\hat{A}_{12}$ must have full column rank, or else n_1 should have been greater. Similarly $\hat{A}_{23}$ is of full column rank, otherwise n_2 should have been greater, and generally the r_k by n_{k+1} block $\hat{A}_{kk+1}$ is of full column rank. This implies that $r_k \geqq n_{k+1}$. From (6.9b) we trivially have $n_k \geqq r_k$. So at this time we know that

$$n_1 \geqq r_1 \geqq n_2 \geqq r_2 \geqq n_3 \geqq r_3 > 0. \tag{6.15}$$

Compute the GSVD of $(A^{(3)}, B^{(3)})$. We have two cases. First, if $n_4 = 0$ then consequently $r_4 = 0$ and the reduction (6.14) is already complete i.e.

$$A_{0_*} = \begin{bmatrix} 0 & \hat{A}_{12} & \hat{A}_{13} \\ & 0 & \hat{A}_{23} \\ & 0 & 0 \end{bmatrix}, \qquad B_{0_*} = \begin{bmatrix} 0 I_{r_1} & & \\ & 0 I_{r_2} & 0 \\ & 0 & 0 I_{r_3} \end{bmatrix}. \tag{6.16}$$

$$\begin{matrix} n_1 & n_2 & n_3 \end{matrix}$$

From (6.14) we get $M_1 = [\hat{A}_{12}, \hat{A}_{13}]$, $M_2 = [\hat{A}_{23}]$ and A_{12} in (6.1) is $[A_{14}\ A_{24}\ A_{34}]^T$, $A_r = A^{(3)}$, $B_r = B^{(3)}$ and since $n_4 = 0$, A_r is of full column rank.

Secondly, if $n_4 \neq 0$ but $r_4 = 0$ we still have new blocks associated with A_{0_*} and B_{0_*}. Make use of $\hat{V}_4$ and $\hat{X}_4$ from the GSVD in an equivalence transformation of the right-

hand side of (6.14). Since $r_4 = 0$ we only add new columns to A_{0*} and B_{0*}:

$$(6.17)\qquad A_{0*} = \left[\begin{array}{c|c|c|c} 0 & \hat{A}_{12} & \hat{A}_{13} & \hat{A}_{14} \\ \hline & 0 & \hat{A}_{23} & \hat{A}_{24} \\ \hline & 0 & 0 & \hat{A}_{34} \end{array}\right], \qquad B_{0*} = \left[\begin{array}{c|c|c|c} 0I_{r_1} & & & \\ \hline & 0I_{r_2} & & \\ \hline & 0 & 0I_{r_3} & 0 \end{array}\right]$$

where also $\hat{A}_{34}$ is of full column rank.

Also in this case we get A_r (6.2a) of full column rank, and therefore $A_r - \lambda B_r$ cannot have any zero eigenvalues or right minimal indices i.e. $A_{0*} - \lambda B_{0*}$ and $A_r - \lambda B_r$ have nonintersecting spectrums.

Introduce the unitary $V \in \mathbb{C}^{m\times m}$ and nonsingular $X \in \mathbb{C}^{n\times n}$ where

$$(6.18a)\qquad V = V_1 \quad V_2 \quad \cdots \quad V_s,$$

$$(6.18b)\qquad X = X_1 \quad X_2 \quad \cdots \quad X_s.$$

If $n_s = 0$ ($r_s = 0$) then V_s and X_s are identity matrices. This completes the reduction.

It remains to show that the block indices n_k and r_k display the structure indices α_0 and α_* accordingly. This can be done by reducing A_{0*} (see (6.16-17)) to a form similar to A' in (4.2a), without changing B_{0*}. Then we obtain the structure blocks $J_k(0)$ and L_k from a reordering of the rows and columns. For details the reader is referred to the technical report [17]. We could also directly apply Theorem 4.1. □

6.2. The RGSVD algorithm. The reduction Theorem 6.1 is the basis for an algorithm to compute the complete Kronecker structure α (3.6). It is possible to prove a dual reduction of $A - \lambda B$ from a finite sequence of row range-nullspace separations of matrix pairs. At each step we then compute the GSVD of a matrix pair $(A^{(k)^H}, B^{(k)^H})$ and make a strictly equivalent transformation of $A^{(k)} - \lambda B^{(k)}$ that compress rows instead of columns (see (6.10) and § 5.2). Then, apart from α_0, we obtain the left row minimal indices α_ℓ (3.5a) of $A - \lambda B$. Since the row minimal indices of $A - \lambda B$ are the column minimal indices of the conjugate transposed pencil $A^H - \lambda B^H$, and vice versa, we also obtain α_ℓ by directly applying Theorem 6.1 to $A^H - \lambda B^H$. It is well known (see Gantmacher [4]) that $A - \lambda B$ and $B - \mu A$ have the same minimal indices and their eigenvalues are λ_k and λ_k^{-1}, respectively, with the same Jordan structures α_k. Therefore when we apply Theorem 6.1 to $B - \mu A$ we obtain α_∞, apart from α_*. A sensitivity analysis of $Ax = \lambda Bx$ (see Stewart [24]-[25] and Sun [29]-[30] and numerical experiment with the QZ-algorithm [39], [5], [20]), show that infinite or almost infinite eigenvalues and almost intersecting nullspaces of A and B are problems that give rise to ill-conditioned eigenvalues that can affect otherwise well-conditioned eigenvalues. Therefore we start by deflating these singularities.

RGSVD-ALGORITHM (*in exact arithmetic*)

1° Compute the infinity structure α_∞ and the left (row) minimal indices α_ℓ by applying the reduction theorem to $B^H - \mu A^H$, giving

$$(6.19)\qquad V_\infty^H (B^H - \mu A^H) X_\infty = \left[\begin{array}{c|c} B_{\infty\ell}^H & B_{12}^H \\ \hline 0 & B_r^{(1)^H} \end{array}\right] - \mu \left[\begin{array}{c|c} A_{\infty\ell}^H & 0 \\ \hline 0 & A_r^{(1)^H} \end{array}\right]$$

where A_{0*} and B_{0*} in (6.2a-b) are replaced by $B_{\infty\ell}^H$ and $A_{\infty\ell}^H$, respectively and now $A_r^{(1)^H}$ is diagonal.

2° Compute the zero structure α_0 and the right minimal indices α_* by applying the reduction theorem to $A_r^{(1)}-\lambda B_r^{(1)}$, giving

$$\hat{V}_0^H(A_r^{(1)}-\lambda B_r^{(1)})\hat{X}_0=\left[\begin{array}{c|c} A_{0*} & A_{12} \\ \hline 0 & A_r^{(2)} \end{array}\right]-\lambda\left[\begin{array}{c|c} B_{0*} & 0 \\ \hline 0 & B_r^{(2)} \end{array}\right] \tag{6.20}$$

where $A_r^{(2)}$ and $B_r^{(2)}$ are square and nonsingular. Further $B_r^{(2)}$ is diagonal with positive elements.

3° Compute the finite and nonzero eigenvalues λ_k of $A_r^{(2)}-\lambda B_r^{(2)}$ and their Jordan structures α_k, $k=1,2,\cdots,p$.

Apart from a row or column scaling of $A_r^{(2)}$, $A_r^{(2)}-\lambda B_r^{(2)}$ is on standard form $A-\lambda I$. Apply the JNF-algorithm (Kågström-Ruhe [12]-[13]) to $C_r=A_r^{(2)}(B_r^{(2)})^{-1}$ i.e.

$$C_rS_r=S_rJ_r \tag{6.21}$$

where J_r is a direct sum of Jordan blocks and the columns of S_r are the corresponding eigenvectors and principal vectors (see also § 2).

As usual (see [12]-[13]) we may obtain the Jordan structures α_k ($k\neq 0,\infty$) in Step 3 by computing an appropriately chosen Schur decomposition and make use of Theorem 2.1 for clusters of eigenvalues. When we know a good approximation β to a multiple eigenvalue λ_k of $A-\lambda B$, we can also compute its Jordan structure α_k by applying Theorem 6.1 to the shifted pencil $(A_r^{(2)}-\beta B_r^{(2)})-\lambda B_r^{(2)}$. This is of course a much more costly operation.

The behaviour of RGSVD in finite precision arithmetic is illustrated in § 8.

6.3. The RGQZD algorithm. In [18] we formulate a *generalized QZ-decomposition* (GQZD) of a matrix pair (A, B), which displays $\mathcal{N}_c(A)$ and $\mathcal{N}_c(A)\cap\mathcal{N}_c(B)$ similar to GSVD. So it is possible to let GQZD take the place of GSVD in the proof of Theorem 6.1. From the GQZ-decomposition of $(A^{(k-1)}, B^{(k-1)})$ we get unitary transformation matrices $\hat{V}_k$ and $\hat{Q}_k$, and (6.10) will be replaced by

$$A^{(k-1)}-\lambda B^{(k-1)}=\hat{V}_k\left(\underbrace{\left[\begin{array}{c|c} 0 & M_k \\ \hline 0 & A^{(k)} \end{array}\right]}_{n_k}\Big\}r_k-\lambda\left[\begin{array}{c|c} 0R_{kk} & N_k \\ \hline 0 & B^{(k)} \end{array}\right]\right)\hat{Q}_k^H. \tag{6.22}$$

Here R_{kk} is r_k by r_k and upper triangular and $B^{(k)}$ has only nonzero elements on and above the diagonal starting at the top left corner. M_k, $A^{(k)}$ and N_k are full matrices.

In the next step $A^{(k)}-\lambda B^{(k)}$ is treated similarly, and by following the proof of Theorem 6.1 we can formulate a reduction theorem in terms of unitary transformations.

THEOREM 6.2. *Assumptions from Theorem* 6.1. *Then there exist unitary matrices* $V\in\mathbb{C}^{m\times m}$ *and* $Q\in\mathbb{C}^{n\times n}$ *such that*

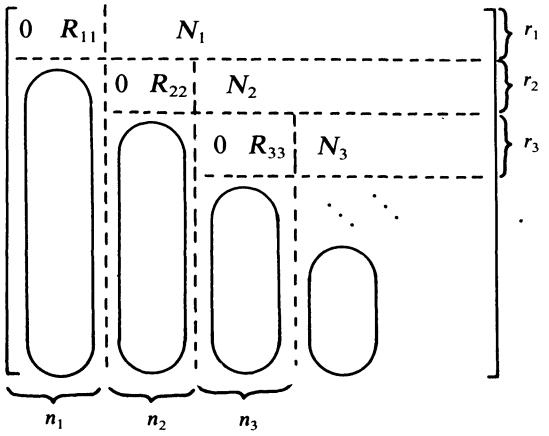
$$V^HAQ=\left[\begin{array}{c|c} A_{0*} & A_{12} \\ \hline 0 & A_r \end{array}\right] \tag{6.23a}$$

and

$$V^HBQ=\left[\begin{array}{c|c} B_{0*} & B_{12} \\ \hline 0 & B_r \end{array}\right] \tag{6.23b}$$

where A_0, is on the form of (6.3) *and*

(6.24) $B_{0*} =$



The block indices n_k and r_k display the structure indices α_0 and α_* according to (6.6)-(6.7).

A_r, A_{12} and B_{12} are full matrix blocks and A_r is of full column rank; B_r has only nonzero elements on and above the diagonal starting at the top left corner and might be singular. The r_k by r_k blocks R_{kk} are nonsingular and upper triangular.

It is now possible to replace the reduction Theorem 6.1 in RGSVD by Theorem 6.2 and we get the *reiterating* GQZ *deflation* (RGQZD) algorithm.

Going back to (6.22) we see that we obtain the same information about the Kronecker structure by using GQZD, but the B-part no longer has the nice and simple structure of $\hat{V}^H B^{(k-1)} \hat{X}_k^{-1}$ in (6.10). Of course there is a trade-off between the simplicity of the structure of the transformed pencil and the conditioning of $\hat{X}_k$. However, this choice can be controlled in terms of a tolerance parameter, say tol. Since $K(\hat{X}_k) = \|\hat{X}_k\|_2 \|\hat{X}_k^{-1}\|_2 = \tau_{\max}/\tau_{\min}$, where $\tau_{\max}$ and $\tau_{\min}$ are the largest and smallest nonzero singular values of $\begin{bmatrix} A^{(k-1)} \\ B^{(k-1)} \end{bmatrix}$, we know the conditioning of $\hat{X}_k$ as we proceed in the algorithm (see [18]), and if $K(\hat{X}_k) > \text{tol}$ we switch to GQZD. On the other hand if, except for the sizes of the blocks (n_k and r_k), we want the diagonal structure of the B-part, then RGSVD will give a better conditioned X (6.2a-b) than performing Gaussian eliminations without pivoting on $V^H(A - \lambda B)Q$ in (6.23a-b).

6.4. Other approaches. Recently V. N. Kublanovskaya [9]-[11] presented the AB-algorithm for handling spectral problems of linear matrix pencils. The AB-algorithm computes two sequences of matrices $\{A_k\}$ and $\{B_k\}$ satisfying

$$A_k B_{k+1} = B_k A_{k+1}, \qquad k = 0, 1, \cdots, \qquad A_0 = A, \quad B_0 = B \tag{6.25}$$

where A_{k+1} and B_{k+1} are blocks (one of them upper triangular) of the nullspace of the augmented matrix $C_k = [A_k \mid B_k]$ in the following way:

$$\mathcal{N}(C_k) = \begin{bmatrix} -B_{k+1} \\ \hline A_{k+1} \end{bmatrix}. \tag{6.26}$$

By applying the AB-algorithm to a regular pencil $A - \lambda B$ we get the Jordan structure α_0 (2.3) of the zero-eigenvalue. Different ways to compute $\mathcal{N}(C_k)$ (6.26) give rise to different algorithms. Kublanovskaya presents the AB-algorithm in terms of the QR and LR decompositions. Our experience computing Jordan structures (see [12]-[14]) demonstrates the necessity of using SVD in order to obtain a numerically stable range-nullspace separation. This motivated a formulation of a modified AB-algorithm

in terms of SVD (see Kågström [15]). When we apply this AB-SVD algorithm to a regular pencil, we get α_0 via a reduction of $A-\lambda B$ that is similar to (6.2a-b) in Theorem 6.1. Now the right-hand transformation is unitary and the left-hand is nonsingular. However, in the general case, where B may be singular, the right-hand transformation Q will have dim $\mathcal{N}(B)$ zero columns. This deficiency is cured in RGSVD (see Theorem 6.1). Although the underlying ideas behind the RGSVD and AB-SVD algorithms are different there are some similarities. In step k we perform a *CS* decomposition [27] of blocks of a partitioned matrix having orthonormal columns. In RGSVD these blocks originate from the range of $\left[\begin{smallmatrix}A\\B\end{smallmatrix}\right]$, while in AB-SVD they originate from the nullspace of $\left[\begin{smallmatrix}A\\B\end{smallmatrix}\right]$. While RGSVD solves the spectral problem for a singular pencil almost as easily as the regular case (see Theorem 6.1), that is not obvious for the *AB*-algorithm.

In [31], Van Dooren presents an algorithm for computing the Kronecker structure of a singular pencil, which is a straightforward generalization of Kublanovskaya's algorithm for determining the Jordan structure of $A-\lambda I$, as used in [12]-[13] (see Theorem 2.1). His reduction is obtained under unitary transformations and is similar in form to the one obtained from RGQZD. If we compare Theorem 6.2 to Van Dooren's corresponding reduction, we see that in his algorithm all diagonal blocks R_{kk} of B_{0_s} (6.24) are full instead of upper triangular, as well as B_r. Any further reduction of the B-part will then in his case include Gaussian-type eliminations without pivoting.

7. Reducing subspaces. In [34] Van Dooren introduces the concept of reducing subspaces, which is a straightforward generalization of deflating subspaces for regular pencils, as introduced by Stewart [24], to the singular case. For clarity we start to recapitulate the regular case.

Let $A-\lambda B$ be a *regular pencil* of order n, $\mathcal{X}$ be a subspace of $\mathbb{C}^n$ of dimension l and

$$\mathcal{Y} = A\mathcal{X} + B\mathcal{X}. \tag{7.1}$$

Then $(\mathcal{X}, \mathcal{Y})$ is a pair of *deflating subspaces* (DS) for $A-\lambda B$ if

$$\dim \mathcal{Y} = \dim \mathcal{X} = l. \tag{7.2}$$

Construct unitary matrices

$$Z = [\underbrace{Z_1}_{l}, Z_2], \qquad Q = [\underbrace{Q_1}_{l}, Q_2] \tag{7.3a}$$

such that

$$\mathcal{X} = \operatorname{span}\{Z_1\}, \qquad \mathcal{Y} = \operatorname{span}\{Q_1\}. \tag{7.3b}$$

From (7.1) we see that $Q_2^H A Z_1 = Q_2^H B Z_1 = 0$, thus

$$Q^H(A-\lambda B)Z = \left[\begin{array}{c|c} \underbrace{\begin{matrix} A_{11} \\ 0 \end{matrix}}_{l} & \begin{matrix} A_{12} \\ A_{22} \end{matrix} \end{array}\right] - \lambda \left[\begin{array}{c|c} \begin{matrix} B_{11} \\ 0 \end{matrix} & \begin{matrix} B_{12} \\ B_{22} \end{matrix} \end{array}\right] \tag{7.4}$$

where $A_{ij} = Q_i^H A Z_j$ and $B_{ij} = Q_i^H B Z_j$.

A *necessary and sufficient condition* that $(\mathcal{X}, \mathcal{Y})$ form a *pair of DS* is that

$$A_{21} = B_{21} = 0. \tag{7.5}$$

Since

$$\begin{bmatrix} Z_1^H \\ Z_2^H \end{bmatrix} (A^H - \lambda B^H)[Q_1 \;\; Q_2] = \left[\begin{array}{c|c} A_{11}^H & 0 \\ \hline A_{12}^H & A_{22}^H \end{array}\right] - \lambda \left[\begin{array}{c|c} B_{11}^H & 0 \\ \hline B_{12}^H & B_{22}^H \end{array}\right], \tag{7.6}$$

we see that $(\mathcal{U}, \mathcal{V})$ where

$$\mathcal{U} = \operatorname{span}\{Q_2\}, \qquad \mathcal{V} = \operatorname{span}\{Z_2\} \tag{7.7}$$

form a pair of DS for $A^H - \lambda B^H$.

As in the case with *invariant subspaces* (IS) for $A - \lambda I$ it is natural to define pairs of right and left DS. The subspaces $(\mathcal{X}, \mathcal{Y})$ satisfying (7.1)-(7.2) form a pair of *right-deflating* subspaces (RDS), and $(\mathcal{U}, \mathcal{V})$ in (7.7) form a pair of *left-deflating subspaces* (LDS) for the part of the spectrum $A - \lambda B$ that corresponds to $A_{22} - \lambda B_{22}$. In [33] Fortran subroutines for computing pairs of DS with specified spectrums are presented.

Now let $A - \lambda B$ be a *singular pencil* where A and $B \in \mathbb{C}^{m \times n}$. Let $\mathcal{X}$ and $\mathcal{Y}$ be subspaces of $\mathbb{C}^n$ and $\mathbb{C}^m$ respectively, l and k their respective dimensions and $\mathcal{Y} = A\mathcal{X} + B\mathcal{X}$ i.e. (7.1) holds. As in the regular case it is possible to construct unitary matrices Q and Z according to (7.3)-(7.7), but we have a different partitioning, namely

$$Z = [\underbrace{Z_1}_{l}, \underbrace{Z_2}_{n-l}]^n, \qquad Q = [\underbrace{Q_1}_{k}, \underbrace{Q_2}_{m-k}]^m \tag{7.8}$$

and consequently

$$\dim \mathcal{X} = l, \qquad \dim \mathcal{Y} = k, \tag{7.9a}$$

$$\dim \mathcal{V} = n - l, \qquad \dim \mathcal{U} = m - k. \tag{7.9b}$$

For any such pair $(\mathcal{X}, \mathcal{Y})$ the following inequality holds (see [34]):

$$\dim \mathcal{Y} \geqq \dim \mathcal{X} - \tilde{i} \tag{7.10}$$

where $\tilde{i}$ is the *number of L_k-blocks* in the KCF (see (3.7b)). When equality is satisfied in (7.10), $(\mathcal{X}, \mathcal{Y})$ is called a pair of *reducing subspaces* (RS) for $A - \lambda B$. Notice that when $A - \lambda B$ is regular then $\tilde{i} = 0$ and the concept of RS reduces to that of DS.

Some facts about a pair $(\mathcal{X}, \mathcal{Y})$ of RS (see [34] for proofs):

(i) The singularities of $A - \lambda B$ due to the minimal indices α_r (3.5b) and α_ℓ (3.5a) are separated to the diagonal pencils $A_{11} - \lambda B_{11}$ and $A_{22} - \lambda B_{22}$, respectively, i.e. $A_{11} - \lambda B_{11}$ has no left (row) minimal indices and $A_{22} - \lambda B_{22}$ has no row (column) minimal indices, or vice versa.

(ii) To every pair of RS there corresponds a pair of DS of the regular part of $A - \lambda B$.

(iii) To every disjoint subset β of the W-KCF-structure $(\alpha_0, \alpha_1, \cdots, \alpha_p, \alpha_\infty)$ (see (3.4) and (3.6)), for example $\beta \equiv (\alpha_0, \alpha_1)$, there exists an *unique* pair of RS that deflates $A - \lambda B$ such that $A_{11} - \lambda B_{11}$ has the Kronecker structure (β, α_r) and $A_{22} - \lambda B_{22}$ the remaining structure $\alpha \backslash (\beta, \alpha_r)$, or vice versa.

(iv) The minimal pair $(\mathcal{X}_{\min}, \mathcal{Y}_{\min})$ and the maximal pair $(\mathcal{X}_{\max}, \mathcal{Y}_{\max})$ of RS are those separating α_r (3.5b) and α_ℓ (3.5a) from the rest of the pencil, or vice versa.

(v) Any pair $(\mathcal{X}, \mathcal{Y})$ of RS satisfies:

$$\{0\} \subset \mathcal{X}_{\min} \subset \mathcal{X} \subset \mathcal{X}_{\max} \subset \mathbb{C}^n,$$

$$\{0\} \subset \mathcal{Y}_{\min} \subset \mathcal{Y} \subset \mathcal{Y}_{\max} \subset \mathbb{C}^m.$$

In [34] the definition of RS is stated in terms of the dimension of $N_r(A - \lambda B)$, the *right nullspace* of $A - \lambda B$, and it is shown that

$$\dim N_r(A - \lambda B) = \tilde{i}. \tag{7.11}$$

The *left nullspace* of $A-\lambda B$, $N_\ell(A-\lambda B)$ is also introduced, where

$$N_\ell(A-\lambda B)=N_s(A^H-\lambda B^H). \tag{7.12}$$

Since the left minimal indices of $A-\lambda B$ are the same as the right minimal indices of $A^H-\lambda B^H$, it follows that (see (3.7a))

$$\dim N_\ell(A-\lambda B)=\tilde{\ell}. \tag{7.13}$$

From the construction of Z and Q it follows that $(\mathcal{U}, \mathcal{V})=(\text{span}\{Q_2\}, \text{span}\{Z_2\})$ is a pair of reducing subspaces of $A^H-\lambda B^H$ and accordingly

$$\dim \mathcal{V}=\dim \mathcal{U}-\tilde{\ell}. \tag{7.14}$$

Following the regular case we denote $(\mathcal{X}, \mathcal{Y})$ a pair of *right-reducing subspace* (RRS) and $(\mathcal{U}, \mathcal{V})$ a pair of *left reducing subspaces* (LRS).

The definitions above are stated in terms of unitary deflations, but they can as well be expressed in terms of nonsingular strictly equivalent transformations. Let $P\in\mathbb{C}^{m\times m}$ and $X\in\mathbb{C}^{n\times n}$ be nonsingular

$$X=[\underbrace{X_1}_{l}|\underbrace{X_2}_{n-l}]; \qquad P^{-1}=\begin{bmatrix} Y_1^H \\ \hline Y_2^H \end{bmatrix}\begin{matrix}\}k \\ \}m-k\end{matrix} \tag{7.15}$$

where

$$\mathcal{X}=\text{span}\{X_1\}; \qquad \mathcal{Y}=\text{span}\{Y_1\} \tag{7.16}$$

and

$$P^{-1}(A-\lambda B)X=\left[\begin{array}{c|c} A_{11} & A_{12} \\ \hline 0 & A_{22} \end{array}\right]-\lambda\left[\begin{array}{c|c} B_{11} & B_{12} \\ \hline 0 & B_{22} \end{array}\right] \tag{7.17}$$

where $A_{ij}=Y_i^HAX_j$ and $B_{ij}=Y_i^HBX_j$. When equality in (7.10) is satisfied, then $(\mathcal{X}, \mathcal{Y})$ defined by (7.16) is a pair of RRS.

From Theorems 6.1 and 6.2 we get the following corollary.

THEOREM 7.1. *Assumptions from Theorem* 6.1.

Partition V and X in (6.2a-b) *accordingly, i.e.*

$$X=[\underbrace{X_1}_{\tilde{n}_s}|\underbrace{X_2}_{n-\tilde{n}_s}], \qquad V^H=\begin{bmatrix} V_1^H \\ \hline V_2^H \end{bmatrix}\begin{matrix}\}\tilde{r}_s \\ \}m-\tilde{r}_s\end{matrix} \tag{7.18}$$

where $\tilde{n}_s$ *and* $\tilde{r}_s$ *are defined in* (6.12), *and let*

$$\mathcal{X}=\text{span}\{X_1\}, \qquad \mathcal{Y}=\text{span}\{V_1\} \tag{7.19}$$

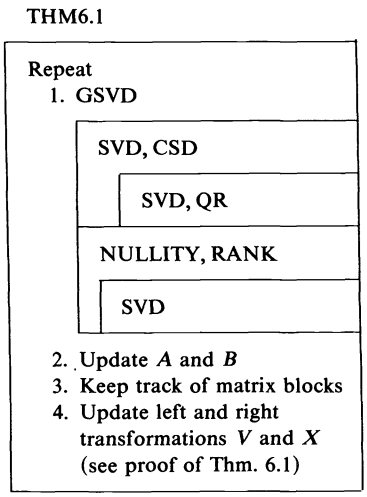
and

$$\mathcal{V}=\text{span}\{X_2\}, \qquad \mathcal{U}=\text{span}\{V_2\}. \tag{7.20}$$

Then $(\mathcal{X}, \mathcal{Y})$ *and* $(\mathcal{U}, \mathcal{V})$ *form pairs of* RRS *and* LRS *of* $A-\lambda B$, *respectively. If we substitute V and Q from* (6.23a-b) *for V and X above we get pairs of* RRS *and* LRS *with orthonormal bases.*

8. Two numerical examples. The reduction Theorem 6.1 has been implemented in Matlab [19]. The program, THM6.1, is of an experimental nature and its main building

blocks are displayed in the following figure:



For each step k, the algorithm works on a matrix pair $(A^{(k)}, B^{(k)})$ (see (6.10)). The most crucial step is in the GSVD algorithm [18] when we make a decision of $n_k = \dim \mathcal{N}(A^{(k)})$ and $n_k - r_k = \dim \mathcal{N}(A^{(k)}) \cap \mathcal{N}(B^{(k)})$. These nullities are determined from computed singular values interpreted as zeros. Singular values less than a deflation tolerance (e.g. machep times the norm of the current matrix) are deleted. These effects are explained by [18, Thm. 4.1], where we derive perturbation bounds to a nearby pencil with prescribed column nullities.

In the following two sections we report results from the Matlab program, implemented in double precision arithmetic (machep $= 1.4_{10} - 17$) on a VAX computer at Stanford Computer Science department. We study one regular pencil and one singular. Each pencil $A - \lambda B$ is constructed in the following way. Starting from a predetermined Kronecker structure α (3.6), the corresponding block structure of the KCF, $K_A - \lambda K_B$ is generated (see (3.2)). Then we compute nonsingular random matrices $P \in \mathbb{C}^{m \times m}$ and $Q^{-1} \in \mathbb{C}^{n \times n}$, corresponding to left and right transformations, respectively. Finally we form

$$A = PK_AQ^{-1}, \tag{8.1a}$$

$$B = PK_BQ^{-1}, \tag{8.1b}$$

i.e., A and $B \in \mathbb{C}^{m \times n}$.

8.1. A regular pencil; $\det A = \det B = 0$. Prescribed Kronecker structure $\alpha = (\alpha_\infty, \alpha_0, \alpha_1)$, where $\alpha_\infty = (0, 1, 1)$, $\alpha_0 = (1, 0, 1)$, $\alpha_1 = (0, 2)$ and $\lambda_1 = 3.0$. In other words, the KCF of $A - \lambda B$ consists of the following Jordan blocks, $J_k(\lambda)$: $J_2(\infty)$, $J_3(\infty)$, $J_1(0)$, $J_3(0)$ and two $J_2(3.0)$-blocks. Other characteristics of $A - \lambda B$:

$$K(P) = \|P\|_2 \cdot \|P^{-1}\|_2 = 329.6; \qquad K(Q) = 100.6,$$

$$\|A\|_F = 629.9; \quad \|B\|_F = 152.5; \quad m = n = 13.$$

Results from RGSVD:

Step 1. *Compute* α_∞. We apply THM6.1 to $B - \mu A$ giving

$$V_\infty^H (B - \mu A) X_\infty = \tilde{B} - \mu \tilde{A} = \left[\begin{array}{c|c} B_\infty & B_{12} \\ \hline 0 & B_r^{(1)} \end{array}\right] - \mu \left[\begin{array}{c|c} A_\infty & 0 \\ \hline 0 & A_r^{(1)} \end{array}\right] \tag{8.2}$$

where $K(V_\infty) = 1.0$ and $K(X_\infty) = 4.94_{10}+3$. The block structure of B_∞ displays the structure indices and we obtained $n_1 = r_1 = 2$, $n_2 = r_2 = 1$, $n_3 = r_3 = 1$ and $n_4 = 0$ showing that we computed $\alpha_\infty = (0, 1, 1)$. $\tilde{A}$ in (8.2) is now in diagonal form and the first five elements are ones to the rounding level. See [17] for a complete display of the computed $\tilde{A}$ and $\tilde{B}$.

Step 2. *Compute* α_0. We apply THM6.1 to $A_r^{(1)} - \lambda B_r^{(1)}$, from (8.2), giving

$$\hat{V}_0^H (A_r^{(1)} - \lambda B_r^{(1)}) \hat{X}_0 = \tilde{A}_1 - \lambda \tilde{B}_1 = \left[\begin{array}{c|c} A_0 & A_{12} \\ \hline 0 & A_r^{(2)} \end{array}\right] - \lambda \left[\begin{array}{c|c} B_0 & 0 \\ \hline 0 & B_r^{(2)} \end{array}\right] \tag{8.3}$$

where $K(\hat{V}_0) = 1.0$ and $K(\hat{X}_0) = 3.3$. Here we display the computed $\tilde{A}_1$ and $\tilde{B}_1$, where

$$\tilde{A}_1 = \begin{array}{c|cc|cc|cccc} r_1 & 0.0000 & -0.0000 & -0.7981 & -0.2135 & -1.1359 & -0.1861 & 1.2255 & -0.8995 \\ r_1 & -0.0000 & 0.0000 & -1.5048 & -0.8027 & -1.5851 & -0.4766 & 0.4333 & 1.4728 \\ \hline r_2 & -0.0000 & 0.0000 & -0.0000 & -0.8645 & -0.9025 & -0.0673 & 0.3577 & -0.3840 \\ r_2 & 0.0000 & 0.0000 & -0.0000 & -0.0000 & 0.0767 & 0.1204 & 0.0400 & 0.8470 \\ \hline & 0.0000 & 0.0000 & -0.0000 & -0.0000 & 0.9063 & 0.0348 & 0.1241 & -0.1311 \\ & -0.0000 & 0.0000 & 0.0000 & 0.0000 & -0.0330 & 0.9245 & -0.1017 & -0.1077 \\ & -0.0000 & 0.0000 & 0.0000 & 0.0000 & -0.1222 & 0.0898 & 0.9474 & -0.0143 \\ & 0.0000 & -0.0000 & 0.0000 & 0.0000 & 0.1001 & 0.0072 & -0.0309 & 0.4842 \\ & \underbrace{\qquad\qquad}_{n_1=2} & & \underbrace{\quad}_{n_2=1} & \underbrace{\quad}_{n_3=1} & n_4 = 0 & & & \end{array}$$

and

$$\tilde{B}_1 = \begin{array}{cc|cc|cccc} 1.0000 & 0. & 0. & 0. & 0. & 0. & 0. & 0. \\ 0. & 1.0000 & 0. & 0. & 0. & 0. & 0. & 0. \\ \hline 0. & 0. & 1.0000 & 0. & 0. & 0. & 0. & 0. \\ 0. & 0. & 0. & 1.0000 & 0. & 0. & 0. & 0. \\ \hline 0. & 0. & 0. & 0. & 0.3830 & 0. & 0. & 0. \\ 0. & 0. & 0. & 0. & 0. & 0.3485 & 0. & 0. \\ 0. & 0. & 0. & 0. & 0. & 0. & 0.2723 & 0. \\ 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0.1383 \end{array}$$

with $r_k = n_k$, showing that we computed $\alpha_0 = (1, 0, 1)$. The zeros and ones displayed above are all accurate to the rounding level.

Step 3. *Compute* $\alpha_k (k \neq 0, \infty)$. First we compute the remaining structure indices α_k by applying THM6.1 to $A_r^{(2)} - \lambda B_r^{(2)}$ from (8.3). The eigenvalues of $A_r^{(2)}(B_r^{(2)})^{-1}$ are

$$\begin{aligned} &2.999999999999977 + 0.000000257850190i \\ &3.000000000000050 + 0.000000044863244i \\ &3.000000000000050 - 0.000000044863244i \\ &2.999999999999977 - 0.000000257860190i. \end{aligned}$$

Apply THM6.1 to $(A_r^{(2)} - \beta B_r^{(2)}) - \lambda B_r^{(2)}$, where β is the mean value of the eigenvalues above. Then

$$\hat{V}_1^H (A_r^{(2)} - (\lambda + \beta) B_r^{(2)}) \hat{X}_1 = \tilde{A}_2 - \lambda \tilde{B}_2 \tag{8.4}$$

where $K(\hat{V}_1) = 1.0$ and $K(\hat{X}_1) = 2.8$. We display $\tilde{A}_2$ and $\tilde{B}_2$ in long precision (15

decimals), where

$$\tilde{A}_2 = \begin{array}{c} r_1 \\ r_2 \end{array} \left[\begin{array}{cc|cc} -0.000000000000000 & 0.000000000000000 & 0.398531413635066 & 1.364920415623216 \\ -0.000000000000000 & 0.000000000000000 & 0.820752952632748 & -0.662761749431175 \\ \hline 0.000000000000063 & -0.000000000000083 & 0.000000000000032 & 0.000000000000088 \\ -0.000000000000012 & 0.000000000000014 & 0.000000000000030 & 0.000000000000022 \end{array}\right]$$

$$\underbrace{\qquad\qquad\qquad\qquad}_{n_1=2} \qquad\qquad \underbrace{\qquad\qquad\qquad\qquad}_{n_2=2}$$

and

$$\tilde{B}_2 = \left[\begin{array}{cc|cc} 1.000000000000000 & 0. & 0. & 0. \\ 0. & 1.000000000000000 & 0. & 0. \\ \hline 0. & 0. & 1.000000000000000 & 0. \\ 0. & 0. & 0. & 1.000000000000000 \end{array}\right]$$

showing that we computed the Jordan structure $\alpha_1 = (0, 2)$ of β.

After regenerating A and B from the composite transformation matrices V and X, ($K(V) = 1.0$ and $K(X) = 4.73_{10}+4$),and the transformed A and B from steps 1-3, we obtained absolute errors in A and B of the size$5.3_{10}-14$and$2.3_{10}-13$, respectively, and the relative error is$3.5_{10}-16$for both A and B.

We also computed the JNF of $C_r^{(2)} = A_r^{(2)}(B_r^{(2)})^{-1}$ (see (6.21)) in order to compare the numerical behaviour of the two algorithms. Results:

$$J_r = \left[\begin{array}{cc|cc} 3.0 & 1.5173 & 0. & 0. \\ 0. & 3.0 & 0. & 0. \\ \hline 0. & 0. & 3.0 & 0.9124 \\ 0. & 0. & 0. & 3.0 \end{array}\right], \qquad K(S_r) = 1.0,$$

$$\|C_r^{(2)} S_r - S_r J_r\|_F = 1.4_{10}-13, \qquad \|C_r^{(2)} - S_r J_r S_r^{-1}\|_F = 1.4_{10}-13.$$

Here we obtain the complete Jordan decomposition of $C_r^{(2)}$. In this example the JNF consists of two Jordan blocks of order two, associated with the true multiple eigenvalue $\lambda_1 = 3.000000000000014$. Accidentally but gratifyingly we computed unitary Jordan bases, S_r! The size of the absolute errors reported above, is the distance from $C_r^{(2)}$ to a 4 by 4 matrix C with the computed JNF as the exact one.

8.2. A singular pencil. Prescribed Kronecker structure $\alpha = (\alpha_\infty, \alpha_\ell, \alpha_0, \alpha_+)$, where $\alpha_\infty = (1, 1)$, $\alpha_\ell = (2, 1)$, $\alpha_0 = (1, 0, 1)$ and $\alpha_+ = (2, 1, 0, 1)$ i.e. the KCF of $A - \lambda B$ consists of the following structure blocks: $J_1(\infty)$, $J_2(\infty)$, $2L_0^T$, $1L_1^T$, $J_1(0)$, $J_3(0)$, $2L_0$, $1L_1$ and $1L_3$. Other characteristics of $A - \lambda B$:

$$K(P) = 100.7, \qquad K(Q) = 254.3,$$

$$\|A\|_F = 108.9, \quad \|B\|_F = 93.4, \quad m = 15, \quad n = 16.$$

Results from RGSVD:

Step 1. *Compute* α_∞ *and* α_ℓ. We start to apply THM6.1 to $B^H - \mu A^H$ giving

$$(8.5) \qquad V_\infty^H (B^H - \mu A^H) X_\infty = \tilde{B}^H - \mu \tilde{A}^H = \left[\begin{array}{c|c} B_{\infty\ell}^H & B_{12}^H \\ \hline 0 & B_r^{(1)H} \end{array}\right] - \mu \left[\begin{array}{c|c} A_{\infty\ell}^H & 0 \\ \hline 0 & A_r^{(1)H} \end{array}\right]$$

where

$\tilde{B} =$

COLUMNS 1 THRU 8

$n_1 = 5$	0.0000	0.0000	−0.0000	0.0000	0.0000	−0.0000	−0.0000	0.0000
	0.0000	0.0000	−0.0000	−0.0000	0.0000	0.0000	0.0000	−0.0000
	−0.0000	−0.0000	0.0000	−0.0000	−0.0000	−0.0000	−0.0000	0.0000
	−0.0000	−0.0000	0.0000	0.0000	−0.0000	0.0000	−0.0000	−0.0000
	−0.0000	−0.0000	0.0000	−0.0000	−0.0000	−0.0000	−0.0000	−0.0000
$n_2 = 2$	−0.4169	−0.6342	0.6511	0.0000	−0.0000	−0.0000	0.0000	0.0000
	−1.7630	1.1495	−0.0092	−0.0000	−0.0000	0.0000	0.0000	0.0000
$n_3 = 0$	0.1096	−0.1286	−0.0551	−0.0569	0.0194	−0.0277	−0.0250	−0.0349
	−0.7155	−0.0437	−0.5008	−0.0945	0.0125	−0.1723	0.0794	0.0785
	0.1740	0.0315	0.1421	0.0673	−0.1453	−0.3284	−0.0186	0.2130
	1.0360	−0.5845	0.0940	−0.3729	0.1488	0.2336	−0.0711	0.0941
	0.6333	−0.9578	−0.5274	−0.0558	0.2658	−0.0010	−0.1823	0.2186
	0.0080	0.1511	0.1523	0.2440	0.7235	−0.0898	0.4016	−0.0524
	0.2091	−0.2185	−0.0789	−0.1403	0.3262	−0.4790	−0.1882	−0.1226
	−0.1851	−0.1994	−0.3128	−0.1923	−0.2360	−0.1220	0.5217	−0.6223
		$r_1 = 3$		$r_2 = 1$				

COLUMNS 9 THRU 16

0.0000	0.0000	−0.0000	0.0000	0.0000	0.0000	−0.0000	0.0000
−0.0000	−0.0000	0.0000	−0.0000	0.0000	0.0000	−0.0000	0.0000
0.0000	0.0000	−0.0000	0.0000	0.0000	−0.0000	−0.0000	−0.0000
0.0000	0.0000	−0.0000	0.0000	−0.0000	−0.0000	0.0000	−0.0000
0.0000	0.0000	−0.0000	0.0000	−0.0000	−0.0000	0.0000	−0.0000
0.0000	−0.0000	0.0000	0.0000	−0.0000	−0.0000	0.0000	−0.0000
−0.0000	0.0000	0.0000	−0.0000	−0.0000	−0.0000	−0.0000	−0.0000
−0.0283	0.0476	0.0577	0.0214	−0.0583	−0.1711	0.0604	0.1661
0.0130	0.0338	−0.2263	−0.0572	0.2713	−0.0831	0.2190	−0.0257
0.0476	−0.2470	−0.0620	0.1674	−0.0744	−0.0401	−0.1864	0.1061
−0.1623	−0.1071	−0.2322	0.2429	−0.0012	−0.0299	−0.1560	−0.0193
−0.2626	−0.5491	0.3013	−0.3422	0.0147	0.0419	0.2646	−0.0279
0.0945	−0.0342	−0.0469	0.0415	0.0607	0.0280	−0.2128	0.1981
−0.0357	0.1981	0.0559	0.2223	−0.4068	−0.0115	0.1784	−0.5509
−0.2671	−0.3933	0.0083	0.0330	−0.0604	−0.0017	0.0265	−0.0818

and

$\tilde{A} =$

COLUMNS 1 THRU 8

0.	0.	0.	0.	0.	0.	0.	0.
0.	0.	0.	0.	0.	0.	0.	0.
1.0000	0.	0.	0.	0.	0.	0.	0.
0.	1.0000	0.	0.	0.	0.	0.	0.
0.	0.	1.0000	0.	0.	0.	0.	0.
0.	0.	0.	0.	0.	0.	0.	0.
0.	0.	0.	1.0000	0.	0.	0.	0.
0.	0.	0.	0.	0.9607	0.	0.	0.
0.	0.	0.	0.	0.	0.8740	0.	0.
0.	0.	0.	0.	0.	0.	0.8201	0.
0.	0.	0.	0.	0.	0.	0.	0.7707
0.	0.	0.	0.	0.	0.	0.	0.
0.	0.	0.	0.	0.	0.	0.	0.
0.	0.	0.	0.	0.	0.	0.	0.
0.	0.	0.	0.	0.	0.	0.	0.

COLUMNS 9 THRU 16

0.	0.	0.	0.	0.	0.	0.	0.
0.	0.	0.	0.	0.	0.	0.	0.
0.	0.	0.	0.	0.	0.	0.	0.
0.	0.	0.	0.	0.	0.	0.	0.
0.	0.	0.	0.	0.	0.	0.	0.
0.	0.	0.	0.	0.	0.	0.	0.
0.	0.	0.	0.	0.	0.	0.	0.
0.	0.	0.	0.	0.	0.	0.	0.
0.	0.	0.	0.	0.	0.	0.	0.
0.	0.	0.	0.	0.	0.	0.	0.
0.	0.	0.	0.	0.	0.	0.	0.
0.4405	0.	0.	0.	0.	0.	0.	0.
0.	0.3766	0.	0.	0.	0.	0.	0.
0.	0.	0.0000	0.	0.	0.	0.	0.
0.	0.	0.	0.0000	0.	0.	0.	0.

showing that we computed $\alpha_\infty = (1, 1)$ and $\alpha_\ell = (2, 1)$.

Notice that the six zero columns of $\tilde{A}$ correspond to the original column nullity of A ($= \dim \mathcal{N}_c(A)$). Especially the zero s_i's in positions (14, 11) and (15, 12) of $\tilde{A}$ correspond to the number of L_0-blocks in the KCF. In addition the last four zero columns of $\tilde{A}$ correspond to the four $J_k(0)$-blocks and/or L_k-blocks of order $\geqq 1$. We cannot trace any further information concerning α_0 or α_s from this reduction. Other characteristics:

$$K(V_\infty) = 1.0, \qquad \|A - X_\infty^H \tilde{A} V_\infty^H\|_F / \|A\|_F = 2.9_{10^{-16}},$$

$$K(X_\infty) = 260.6, \qquad \|B - X_\infty^H \tilde{B} V_\infty^H\|_F / \|B\|_F = 2.8_{10^{-16}}.$$

Step 2. *Compute* α_0 *and* α_s. We apply THM6.1 to $A_r^{(1)} - \lambda B_r^{(1)}$ from (8.5) giving

$$\hat{V}_0^H (A_r^{(1)} - \lambda B_r^{(1)}) \hat{X}_0 = \tilde{A}_1 - \lambda \tilde{B}_1 = \left[\begin{array}{c|c} A_{0_s} & A_{12} \\ \hline 0 & A_r^{(2)} \end{array}\right] - \lambda \left[\begin{array}{c|c} B_{0_s} & 0 \\ \hline 0 & B_r^{(2)} \end{array}\right] \tag{8.6}$$

where

$\tilde{A}_1 =$

COLUMNS 1 THRU 8

$r_1 = 4$	−0.0000	−0.0000	0.0000	−0.0000	−0.0000	−0.0000	0.3348	−0.2610
	0.0000	−0.0000	0.0000	−0.0000	−0.0000	−0.0000	−0.3576	0.1021
	−0.0000	−0.0000	0.0000	−0.0000	−0.0000	−0.0000	0.8689	0.2242
	−0.0000	−0.0000	0.0000	0.0000	−0.0000	0.0000	−0.0711	0.9972
$r_2 = 2$	0.0000	0.0000	−0.0000	0.0000	0.0000	0.0000	−0.0000	0.0000
	−0.0000	−0.0000	0.0000	0.0000	−0.0000	0.0000	−0.0000	−0.0000
$r_3 = 2$	0.0000	−0.0000	−0.0000	0.0000	0.0000	−0.0000	−0.0000	−0.0000
	0.0000	−0.0000	0.0000	−0.0000	−0.0000	−0.0000	0.0000	0.0000
$r_4 = 0$	$n_1 = 6$						$n_2 = 3$	

COLUMNS 9 THRU 12

−0.9700	−0.3030	−0.5545	0.4118
0.5164	−0.0415	0.1195	−0.1978
0.5608	0.0954	0.2522	−0.2429
−0.3114	−0.0521	−0.1297	−0.0349
0.0000	−0.5534	−0.4796	−0.1271
−0.0000	−0.9445	0.2555	−0.4601
−0.0000	0.0000	−0.0000	0.8115
0.0000	−0.0000	0.0000	0.5843
	$n_3 = 2$		$n_4 = 1$

and

COLUMNS 1 THRU 8

$$\tilde{B}_1 = \left[\begin{array}{cccccc|cc} 0. & 0. & 1.0000 & 0. & 0. & 0. & 0. & 0. \\ 0. & 0. & 0. & 1.0000 & 0. & 0. & 0. & 0. \\ 0. & 0. & 0. & 0. & 1.0000 & 0. & 0. & 0. \\ 0. & 0. & 0. & 0. & 0. & 1.0000 & 0. & 0. \\ \hline 0. & 0. & 0. & 0. & 0. & 0. & 0. & 1.0000 \\ 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. \\ \hline 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. \\ 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. \end{array}\right]$$

COLUMNS 9 THRU 12

$$\left[\begin{array}{cccc} 0. & 0. & 0. & 0. \\ 0. & 0. & 0. & 0. \\ 0. & 0. & 0. & 0. \\ 0. & 0. & 0. & 0. \\ \hline 0. & 0. & 0. & 0. \\ 1.0000 & 0. & 0. & 0. \\ \hline 0. & 1.0000 & 0. & 0. \\ 0. & 0. & 1.0000 & 0. \end{array}\right]$$

showing that we computed $\alpha_0 = (1, 0, 1)$ and $\alpha_* = (2, 1, 0, 1)$. Notice that $\tilde{A}_1 - \lambda\tilde{B}_1$ in (8.6) has the same Kronecker structure as the introductory example studied in § 4 and is similar to the pivoted $A' - \lambda B'$ (see (4.2a-b)).

After regenerating A and B from the composite transformation matrices V and X ($K(V) = 260.6 (= K(X_\infty))$ and $K(X) = 5.0 (= K(\hat{X}_0))$, and the transformed matrices from step 1-2 we obtained absolute errors in A and B of size $3.8_{10^{-14}}$ and $3.1_{10^{-14}}$, respectively. The relative errors are $3.5_{10^{-16}}$ and $3.3_{10^{-16}}$, respectively. Since $\max\{K(P), K(Q)\}$ machep $= 3.5_{10^{-15}}$ and $\max\{K(V), K(X)\}$ machep $= 3.6_{10^{-15}}$, where P and Q are the transformation matrices that generated A and B (see (8.1a-b)), we interpret our computed results as being very satisfactory.

9. Conclusions and further research. For each Kronecker structure α (3.6), the set of pencils $\mathscr{E}_\alpha$ constitutes a manifold in the space of singular pencils $A - \lambda B$ where A and $B \in \mathbb{C}^{m\times n}$. When working in finite precision arithmetic it is impossible to decide whether a given pencil $A - \lambda B$ belongs to a certain $\mathscr{E}_\alpha$. It is more appropriate to require

$$A - \lambda B \in \mathscr{E}_\alpha(\varepsilon)$$

where

$$\begin{aligned} &\mathscr{E}_\alpha(\varepsilon) = \{\text{pencils } X - \lambda Y, X \text{ and } Y \in \mathbb{C}^{m\times n}: \\ &\|X - C\|/\|X\| + \|Y - D\|/\|Y\| \leqq \varepsilon \text{ and } C - \lambda D \in \mathscr{E}_\alpha\}. \end{aligned} \tag{9.1}$$

From the results computed by RGSVD and reported in § 8 we conclude:

Regular pencil $A - \lambda B \in \mathscr{E}_\alpha$ $(7.0_{10^{-16}})$,
Singular pencil $A - \lambda B \in \mathscr{E}_\alpha$ $(6.8_{10^{-16}})$.

In other words, the relative distance from the given $A - \lambda B$ to a pencil $C - \lambda D \in \mathscr{E}_\alpha$ is of order $7_{10^{-16}}$ for both examples. Since these distances are at the rounding level, RGSVD has computed well-determined Kronecker structures and associated canonical reductions, giving well-defined pairs of reducing subspaces (see § 7). For RGSVD the

dominating contributions to the relative perturbations in (9.1) originate from deleted singular values when computing column and row nullities of certain matrix pairs. The deleted generalized singular values accumulate from each deflation step k. The sensitivity of the computed structures e.g. α_0 and α_* are extremely dependent on the gap (β/δ) between the singular values we interpret as zero (δ) and nonzero (β) respectively. We get *well-determined structures* α_0 and α_* if the quotient β/δ is large enough and the ideal case is when δ is close to machep and β is of order 1. If there is no appreciable gap $\delta \approx \beta$, the problem of determining the Kronecker structure of $A-\lambda B$ is ill-conditioned in the sense that $A-\lambda B$ also belongs to $\mathscr{E}_{\alpha'}(\varepsilon')$ for another α' and ε' of the same size as ε. The choice of structure is then to some extent arbitrary and the pathological behaviour is inherent to the pencil $A-\lambda B$ itself. In RGSVD we at each step k seek $n_k = \dim \mathcal{N}(A^{(k)})$ and $n_k - r_k = \dim \mathcal{N}(A^{(k)}) \cap \mathcal{N}(B^{(k)})$ such that $\beta/\delta \geqq 1000$.

In the reduction Theorem 6.1 we prove that the problem of computing the Kronecker structures α_0 and α_* of a singular pencil $A-\lambda B$ associate with repeated deflations in terms of generalized singular value decompositions of matrix pairs $(A^{(k)}, B^{(k)})$. In fact we have proved an one-to-one correspondence between $A-\lambda B$ and GSVD which is similar to the one-to-one correspondence that exists between $A-\lambda I$ and SVD for the Jordan structure problem, (see Theorem 2.1). Furthermore the proof of Theorem 6.1 gives us an algorithm to compute α_0 and α_* based on singular value decompositions, that also makes it attractive from a numerical point of view. Theorem 4.1 in [18] gives a picture of the sensitivity of one step of RGSVD. It is possible to make use of that analysis and derive a priori bounds of the distance from a given $A-\lambda B$ to $C-\lambda D \in \mathscr{E}_\alpha$. It is also interesting to make the corresponding analysis for RGQZD (see § 6.3). This will be the topic of a forthcoming paper. We also plan to produce software in Fortran for stably computing the Kronecker structure and reducing subspaces for uncertain data, where RGSVD will be the basic algorithm.

Acknowledgments. I am grateful to Jim Wilkinson for his comments on different aspects of the problem and to James Demmel, Axel Ruhe and Charles Van Loan for their constructive comments on the manuscript.

Parts of this work was done while the author during the winter quarter 1983 enjoyed the kind hospitality of Gene Golub and the nice and stimulating atmosphere at the Computer Science department of Stanford University.

Financial support has partly been received from the Swedish Institute of Applied Mathematics (ITM).

Finally I am indebted to Lena Carneland and Inga-Lena Olsson for their rapid and careful typing of the manuscript.

REFERENCES

[1] C. DAVIS AND W. M. KAHAN, *The rotation of eigenvectors by a perturbation* III, SIAM J. Numer. Anal., 7 (1970), pp. 1-46.

[2] J. DEMMEL, *The condition number of equivalence transformations that block diagonalize matrix pencils*, in Kågström and Ruhe [16], pp. 2-16, SIAM J. Numer. Anal., 20 (1983), pp. 599-610.

[3] ———, *A numerical analyst's Jordan canonical form*, Report PAM-156, Thesis, Center for Pure and Applied Mathematics, Univ. California, Berkeley, 1983.

[4] F. R. GANTMACHER, *The Theory of Matrices, Vol.* I *and* II (Transl.), Chelsea, New York, 1959.

[5] B. S. GARBOW, J. M. BOYLE, J. J. DONGARRA AND C. B. MOLER, *Matrix Eigensystem Routines*— EISPACK *Guide Extension*, Lecture Notes in Computer Science 51, Springer-Verlag, Berlin, 1977.

[6] G. H. GOLUB AND J. H. WILKINSON, *Ill-conditioned eigensystems and the computation of the Jordan canonical form*, SIAM Rev. 18 (1976), pp. 578-619.

[7] W. KAHAN, *Conserving confluence curbs ill-condition*, Tech. Report 6, Dept. Computer Science, Univ. California, Berkeley, 1972, pp. 1-54.

[8] V. N. KUBLANOVSKAYA, *On a method of solving the complete eigenvalue problem for a degenerate matrix*, USSR Comput. Math. Phys., 6, 4 (1968), pp. 1-14.

[9] ———, *The AB-algorithm and its modifications for the spectral problem of linear pencils of matrices*, Numer. Math., 43 (1984), pp. 329-342.

[10] ———, *On algorithms for the solution of spectral problems of linear matrix pencils*, LOMI-preprint E-1-82, USSR Academy of Sciences, Leningrad, 1982, pp. 1-43.

[11] ———, *An approach to solving the spectral problem of* $A-\lambda B$, in Kågström and Ruhe [16], pp. 17-29.

[12] B. KÅGSTRÖM AND A. RUHE, *An algorithm for numerical computation of the Jordan normal form of a complex matrix*, ACM Trans. Math. Software, 6 (1980), pp. 398-419.

[13] ———, *ALGORITHM* 560, JNF, *An algorithm for numerical computation of the Jordan normal form of a complex matrix* [*F2*], ACM Trans. Math. Software, 6 (1980), pp. 437-443.

[14] B. KÅGSTRÖM, *How to compute the Jordan normal form—the choice between similarity transformations and methods using the chain relations*, Report UMINF. 91-81, Inst. of Information Processing, Umeå Univ., Sweden, 1981, pp. 1-48.

[15] ———, *On computing the Kronecker canonical form of regular* $(A-\lambda B)$*-pencils*, in Kågström and Ruhe [16], pp. 30-57.

[16] B. KÅGSTRÖM AND A. RUHE, *Matrix Pencils*, Proceedings, Pite Havsbad, 1982, Lecture Notes in Mathematics 973, Springer-Verlag, Berlin, 1983.

[17] B. KÅGSTRÖM, RGSVD—*An algorithm for computing the Kronecker structure and reducing subspaces of singular* $A-\lambda B$ *pencils*, Report UMINF-112.83, Inst. of Information Processing, Umeå Univ., Sweden, 1983, pp. 1-72.

[18] ———, *The generalized singular value decomposition and the general* $(A-\lambda B)$*-problem*, BIT 24 (1984) pp. 568-583.

[19] C. MOLER, *MATLAB—An interactive matrix laboratory*, Dept. Computer Science, Univ. New Mexico, Albuquerque, 1982.

[20] C. B. MOLER AND G. W. STEWART, *An algorithm for the generalized eigenvalue problem*, SIAM J. Numer. Anal., 10 (1973), pp. 241-256.

[21] C. C. PAIGE AND M. A. SAUNDERS, *Towards a generalized singular value decomposition*, SIAM J. Numer. Anal., 18 (1981), pp. 398-405.

[22] C. C. PAIGE, *A note on a result of Sun Ji-Guang: Sensitivity of the* CS *and* GSV *decompositions*, SIAM J. Numer. Anal., 21 (1984), pp. 186-191.

[23] R. F. SINCOVEC, B. DEMBART, M. A. EPTON, A. M. ERISMAN, J. W. MANKE AND E. L. YIP, *Solvability of large scale descriptor systems*, Final Report DOE Contract ET-78-C-01-2876, Boeing Computer Services Co., Seattle, WA, 1979.

[24] G. W. STEWART, *Error and perturbation bounds for subspaces associated with certain eigenvalue problems*, SIAM Rev. 15 (1973), pp. 752-764.

[25] ———, *Perturbation theorems for the generalized eigenvalue problem*, in Recent Advances in Numerical Analysis, C. de Boor and G. H. Golub, eds., Academic Press, New York, 1978, pp. 193-206.

[26] ———, *A method for computing the generalized singular value decomposition*, in Kågström and Ruhe [16], pp. 207-220.

[27] ———, *Computing the CS decomposition of a partitioned orthonormal matrix*, Numer. Math., 41 (1982), pp. 297-306.

[28] JI-GUANG SUN AND L. ELSNER, *Perturbation theorems for the generalized eigenvalue problem*, Linear Alg. Appl., 48 (1982), pp. 341-357.

[29] J-G. SUN, *Perturbation analysis for the generalized eigenvalue and the generalized singular value problem*, in Kågström and Ruhe [16], pp. 221-244.

[30] ———, *Perturbation analysis for the generalized singular value problem*, SIAM J. Numer. Anal., 20 (1983), pp. 611-625.

[31] P. VAN DOOREN, *The computation of Kronecker's canonical form of a singular pencil*, Linear Alg. Appl., 27 (1979), pp. 103-141.

[32] ———, *The generalized eigenstructure problem in linear system theory*, IEEE Trans. Automat Control, AC-26 (1981), pp. 111-129.

[33] ———, *ALGORITHM* 590, DSUBSP *and* EXCHQZ: *Fortran subroutines for computing deflating subspaces with specified spectrum*, ACM Trans. Math. Software, 8 (1982), pp. 376-382.

[34] ———, *Reducing subspaces: definitions, properties and algorithms*, in Kågström and Ruhe [16], pp. 58-73.

[35] C. VAN LOAN, *A general matrix eigenvalue algorithm*, SIAM J. Numer. Anal., 12 (1975), pp. 819-834.

[36] ———, *Generalizing the singular value decomposition*, SIAM J. Numer. Anal., 13 (1976), pp. 76-83.

[37] ———, *Computing the CS and the generalized singular value decompositions*, TR 84-614, Dept. Comp. Science, Cornell Univ., New York, 1984, pp. 1-17.

[38] J. H. WILKINSON, *Linear differential equations and Kronecker's canonical form*, in Recent Advances in Numerical Analysis, C. de Boor and G. Golub, eds., Academic Press, New York, 1978, pp. 231-265.

[39] ———, *Kronecker's canonical form and the QZ algorithm*, Linear Alg. Appl., 28 (1979), pp. 285-303.

[40] ———, *The algebraic eigenvalue problem—background theory*, Lecture notes, Stanford Univ., Stanford, CA, 1983.

SIAM J. SCI. STAT. COMPUT.
Vol. 7, No. 1, January 1986

015

A HAMILTONIAN *QR* ALGORITHM*

RALPH BYERS†

Abstract. This paper presents a variant *QR* algorithm for calculating a Hamiltonian-Schur decomposition [10]. It is defined for Hamiltonian matrices that arise from single input control systems. Numerical stability and Hamiltonian structure are preserved by using unitary symplectic similarity transformations. Following a finite step reduction to a Hessenberg-like condensed form, a sequence of similarity transformations yields a permuted triangular matrix. As the iteration converges, it deflates into problems of lower dimension. Convergence is accelerated by varying a scalar shift. When the Hamiltonian matrix is real, complex arithmetic can be avoided by using an implicit double shift technique. The Hamiltonian-Schur decomposition yields the same invariant subspace information as a Schur decomposition but requires significantly less work and storage for problems of size greater than about 20.

Key words. *QR* algorithm, Hamiltonian matrices, symplectic matrices, algebraic Riccati equation, optimal control

Introduction. A matrix $H \in \mathbb{C}^{2n\times 2n}$ is Hamiltonian [1], [7], [10] if $JH = H^*J = 0$ where

$$J = \begin{bmatrix} 0_n & I_n \\ -I_n & 0_n \end{bmatrix}. \tag{1}$$

The superscript asterisk denotes the conjugate transpose of a matrix, I_n denotes the n-by-n identity matrix and 0_n denotes the n-by-n matrix of zeros. Throughout this paper, H will denote a Hamiltonian matrix. A bold face $\mathbf{H}$ will denote the set of Hamiltonian matrices. Partitioned into n-by-n blocks H is of the form

$$H = \begin{bmatrix} A^* & G \\ F & -A \end{bmatrix} \tag{2}$$

where $G = G^*$ and $F = F^*$. We will write $H = \mathrm{HAM}(A, G, F)$ for the matrix of (2).

Hamiltonian matrices arise in mechanics and control theory. For example, consider the optimal control problem

$$\text{minimize} \int_0^\infty \mathbf{y}^*\mathbf{y} + \mathbf{u}^*\mathbf{u}\, dt \tag{3}$$

subject to the control system

$$\begin{aligned} d\mathbf{x}/dt &= A\mathbf{x} + B\mathbf{u}, \\ \mathbf{y} &= C\mathbf{x} \end{aligned} \tag{4}$$

where $A \in \mathbb{C}^{n\times n}$, $B \in \mathbb{C}^{n\times p}$, $C \in \mathbb{C}^{q\times n}$ and $\mathbf{x}(0) = \mathbf{x}_0 \in \mathbb{C}^n$ are given. Set $G = C^*C$ and $F = BB^*$. It is well known [5] that under mild conditions (3) is minimized by

$$\mathbf{u} = -B^*X\mathbf{x}$$

where $X \in \mathbb{C}^{n\times n}$ satisfies the algebraic Riccati equation

$$G + A^*X + XA - XFX = 0_n. \tag{5}$$

With this choice of $\mathbf{u}$, $\mathbf{x}$ obeys

$$\frac{d\mathbf{x}}{dt} = (A - FX)\mathbf{x}.$$

* Received by the editors October 11, 1983, and in revised form October 19, 1984. This work was part of the author's doctoral dissertation while a graduate student supported by Cornell University. It merited co-award of the Householder Prize V (1983) for the best thesis in Numerical Algebra.

† Department of Mathematical Sciences, Northern Illinois University, DeKalb, Illinois 60115.

Denote the eigenvalues of a matrix $M \in \mathbb{C}^{n\times n}$ by $\lambda(M)$ and the open left half complex plane by $\mathbb{C}^-$. In general there are many solutions X of (5). The desired one is stabilizing in the sense that $\lambda(A-FX) \subset \mathbb{C}^-$.

Solutions of (5) are related to invariant subspaces of HAM (A, G, F) [11]. If Y, Z and $W \in \mathbb{C}^{n\times n}$ satisfy

$$\begin{bmatrix} A^* & G \\ F & -A \end{bmatrix}\begin{bmatrix} Y \\ Z \end{bmatrix} = \begin{bmatrix} Y \\ Z \end{bmatrix} W,$$

then $X = YZ^{-1}$ is a solution. The space spanned by the columns of $[Y^*, Z^*]^*$ is an invariant subspace of $H = \mathrm{HAM}(A, G, F)$. Any basis of the same invariant subspace yields the same value of X. We require $\lambda(-W) = \lambda(A-FX)$ to be contained in $\mathbb{C}^-$. If the eigenvalues of HAM (A, G, F) have nonzero real part, then the desired invariant subspace exists and is unique. Under mild conditions Z is nonsingular and $X = YZ^{-1}$ is unique [4].

Laub [6] shows how to obtain an orthogonal basis by using the Francis *QR* algorithm [3]. Although effective, Laub's method does not take advantage of the special structure of Hamiltonian matrices. $H \in \mathbf{H}$ is treated as a general $2n$-by-$2n$ matrix. Ignoring the Hamiltonian structure of H is inefficient in both work and storage. Rounding errors can produce some surprising results. Eigenvalues of real Hamiltonian matrices appear in plus/minus pairs, but eigenvalues calculated by the *QR* algorithm in the presence of rounding error may not [15]. This paper shows how to use unitary symplectic matrices to preserve structure, cutting work and storage requirements in half. A matrix $S \in \mathbb{C}^{2n\times 2n}$ is symplectic [1], [7], [10] if $S^*JS = J$. (J is defined in (1).) In the presence of rounding errors our algorithm produces the eigenvalues of a "nearby" Hamiltonian matrix.

Unfortunately, our algorithm is limited to Hamiltonian matrices in which $\mathrm{rank}(F) = 1$. Such matrices arise when (4) is a single input control system, i.e., when B consists of a single column.

Section 1 reviews the Schur decomposition, the Hamiltonian-Schur decomposition, and the symplectic Schur decomposition. The economical use of Householder symplectic and Jacobi symplectic matrices is discussed. Section 2 uses the Francis *QR* iteration to motivate the Hamiltonian *QR* iteration. Each Hamiltonian *QR* step is equivalent to a Francis *QR* step applied to an associated symplectic matrix. Section 3 describes a finite step reduction of a Hamiltonian matrix to a Hessenberg-like condensed form. The condensed form is preserved by the rest of the algorithm. A heuristic argument suggests that one step of the Hamiltonian *QR* algorithm is equivalent to two steps of the Francis *QR* algorithm. Section 4 develops the Hamiltonian *QR* step. Hamiltonian structure is preserved throughout, so no $2n$-by-$2n$ array is required. A numerically stable algorithm is presented. Section 5 discusses the details of deflation, choice of shift, and ordering of eigenvalues. Section 6 sketches how to modify the Hamiltonian *QR* algorithm to avoid the use of complex arithmetic. Like the double Francis *QR* step, the trick is to combine two successive Hamiltonian *QR* steps into one. Section 7 presents the results of some numerical experiments solving the algebraic Riccati equation. Especially for problems of high dimension, the Hamiltonian *QR* iteration represents a significant savings over the Francis *QR* iteration.

Capital letters denote matrices, boldface lower case denote vectors, nonboldface lower case letters denote scalars. The ith row and jth column of the matrix A are represented by A_{i*} and A_{*j} respectively. The norm $\|A\|$ may denote any reasonably well balanced consistent matrix norm such as the Frobenius norm,

$$\|A\| = (\mathrm{trace}\,(A^*A))^{(1/2)}.$$

1. Numerical tools. A matrix $S \in \mathbb{C}^{2n\times 2n}$ is symplectic [1], [7], [10] if $S^*JS = J$. (J is defined in (1).) A bold face $\mathbf{S}$ will denote the set of symplectic matrices. Products of symplectic matrices are symplectic. Symplectic matrices are nonsingular.

If $H \in \mathbf{H}$ and $S \in \mathbf{S}$, then $SHS^{-1} \in \mathbf{H}$. Following the suggestion of [10], we use symplectic similarity transformations to preserve Hamiltonian structure.

If Q is both symplectic and unitary, then it is of the form

$$Q = \begin{bmatrix} Q_1 & Q_2 \\ -Q_2 & Q_1 \end{bmatrix}$$

where $Q_1, Q_2 \in \mathbb{C}^{n\times n}$ [10]. The set of unitary symplectic matrices will be denoted $\mathbf{S}-\mathbf{U}$. Note that computer programs need store only the first n rows of Q.

If $V \in \mathbb{C}^{n\times n}$ is unitary, then $\operatorname{diag}(V, V) \in \mathbf{S}-\mathbf{U}$. A Householder symplectic matrix [10] is the matrix $P = P(k, m, \mathbf{x}) \in \mathbf{S}-\mathbf{U}$ defined by

$$P(k, m, \mathbf{x}) = \begin{bmatrix} \hat{P} & 0_n \\ 0_n & \hat{P} \end{bmatrix}$$

where

$$\hat{P} = \hat{P}(k, m, \mathbf{x}) = I_n - 2\mathbf{u}\mathbf{u}^*$$

and $\mathbf{u} \in \mathbb{C}^n$ is obtained from $\mathbf{x} \in \mathbb{C}^n$ by setting $\mathbf{u} = \mathbf{v}/\|\mathbf{v}\|_2$ where

$$v_j = \begin{cases} 0 & \text{if } 1 \leqq j < \min(k, m) \text{ or } \max(k, m) < j \leqq n, \\ x_j & \text{if } \min(k, m) < j < \max(k, m) \text{ or } j = m, \\ x_k + (x_k/|x_k|)(|x_k|^2 + \cdots + |x_m|^2)^{(1/2)} & \text{if } j = k. \end{cases} \tag{6}$$

If $x_k = 0$, then the quotient $x_k/|x_k|$ may be set equal to 1. By convention, if $\mathbf{v} = 0$, then $\hat{P} = I_n$. We admit the possibility $k > m$. The Householder reflection $\hat{P}$ is a Hermitian unitary matrix such that if $\mathbf{y} = \hat{P}\mathbf{x}$, then

$$y_j = \begin{cases} x_j & \text{if } 1 \leqq j < \min(k, m) \text{ or } \max(k, m) < j \leqq n, \\ 0 & \text{if } \min(k, m) < j < \max(k, m) \text{ or } j = m, \\ -(x_k/|x_k|)(|x_k|^2 + \cdots + |x_m|^2)^{(1/2)} & \text{if } j = k. \end{cases}$$

Algorithms using Householder reflections depend upon their ability to zero some components of a vector while leaving others unchanged. Specifically, suppose $\mathbf{v} \in \mathbb{C}^n$, $\mathbf{w} = \hat{P}(k, m, \mathbf{x})\mathbf{v}$ and $\mathbf{y} = \hat{P}(k, m, \mathbf{x})\mathbf{x}$. If $1 \leqq j < \min(k, m)$ or $\max(k, m) < j \leqq n$, then $w_j = v_j$ and $y_j = x_j$. Furthermore, if $j = m$ or $\min(k, m) < j < \max(k, m)$, then $y_j = 0$.

Define a flop as the computational effort to execute the FORTRAN statement

$$A(I, J) = A(I, J) - S^*A(K, J).$$

We will speak of a real flop when A and S are of REAL type and a complex flop when A and S are of COMPLEX type.

Our algorithms derive much of their efficiency from the following well-known economies of work and storage. Let $\mathbf{u}$ be as in (6). Set

$$\hat{\mathbf{u}} = \mathbf{u}/u_k$$

and

$$s = 2u_k^2/(\mathbf{u}^*\mathbf{u}).$$

Note $\hat{u}_k = 1$. $\hat{P}(k, m, \mathbf{x})$ can be stored on a computer as s and the $m-k$ nontrivial components of $\hat{\mathbf{u}}$. When $\hat{P}$ is stored in this way, a vector $\mathbf{v} \in \mathbb{C}^n$ can be replaced by $\hat{P}\mathbf{v}$

without explicitly forming $\hat{P}$ [13, p. 234], [17, pp. 157-159] by setting

$$\mathbf{v} := \mathbf{v} - \hat{\mathbf{u}}(s(\hat{\mathbf{u}}^*\mathbf{v})). \tag{7}$$

It is not necessary to perform multiplications by $\hat{u}_k = 1$ nor by those components of $\hat{\mathbf{u}}$ that are known to be zero. So (7) requires only $2|m-k|+1$ flops.

The Hermitian matrix M may be overwritten by $\hat{P}M\hat{P}$ without the use of an n-by-n array [17, p. 292] by

$$\begin{aligned} \mathbf{v} &:= s(M\hat{\mathbf{u}}), \\ \mathbf{w} &:= \mathbf{v} - (s/2)(\hat{\mathbf{u}}^*\mathbf{v})\hat{\mathbf{u}}, \\ M &:= M - \hat{\mathbf{u}}\mathbf{w}^* - \mathbf{w}\hat{\mathbf{u}}^*. \end{aligned} \tag{8}$$

Notice that only the upper triangle of M need be stored. If we do not bother to multiply by those components of $\hat{\mathbf{u}}$ that are known to be zero or one, then (8) requires approximately $2(n+1)|m-k|+n+2$ flops.

We will use a superscript "$\hat{}$" to designate Householder reflections that are stored and used as described above.

A Jacobi symplectic matrix [10] $J(k, c, s)$ is a unitary symplectic matrix of the form

$$J(k, c, s) = \begin{bmatrix} C & S \\ -S & C \end{bmatrix} \tag{9}$$

where $|c|^2+|s|^2=1$, $\bar{c}s \in \mathbb{R}$ and

$$\begin{aligned} C &= \operatorname{diag}(\underbrace{1, 1, 1, \cdots, 1}_{k-1}, c, 1, \cdots, 1), \\ S &= \operatorname{diag}(\underbrace{0, 0, 0, \cdots, 0}_{k-1}, s, 0, \cdots, 0). \end{aligned} \tag{10}$$

Given $\mathbf{v}, \mathbf{w} \in \mathbb{C}^n$ such that $\bar{v}_k w_k \in \mathbb{R}$, there is a Jacobi symplectic matrix $Q = J(k, c, s)$ such that if

$$\begin{bmatrix} \mathbf{x} \\ \mathbf{y} \end{bmatrix} = Q \begin{bmatrix} \mathbf{v} \\ \mathbf{w} \end{bmatrix},$$

then $y_k = 0$ and for $j \neq k$, $x_j = v_j$ and $y_j = w_j$.

Let $H' = \mathrm{HAM}(A', G', F') = J(k, c, s)\,\mathrm{HAM}(A, G, F)J^*(k, c, s)$. The matrices A', G' and F' can be obtained directly from A, G and F without using an extra square array simply by forming the obvious linear combinations of the kth rows and columns of A, G and F. No more than a few n flops are required.

Care must be taken to avoid overflows, underflows and excessive rounding errors when constructing Householder and Jacobi symplectics. See [13] or [17] for details.

The following theorem describes a decomposition which displays the invariant subspace structure of a Hamiltonian matrix. Our algorithm calculates this decomposition.

THEOREM 1 (Hamiltonian-Schur decomposition (HSD) [10]). *If the eigenvalues of* $H = \mathrm{HAM}(A, G, F)$ *have nonzero real part, then there exists* $Q \in \mathbf{S}-\mathbf{U}$ *such that*

$$Q^*HQ = \begin{bmatrix} T_1^* & T_2 \\ 0_n & -T_1 \end{bmatrix} \tag{11}$$

where T_1 *is lower triangular and* $T_2 = T_2^*$. Q *can be chosen so that* $\lambda(-T_1) \subset \mathbb{C}^-$.

We will refer to a matrix $T \in \mathbf{H}$ with the zero structure of the right-hand side of (11) as Hamiltonian triangular.

A useful corollary is

COROLLARY 2 (Symplectic-Schur decomposition (SSD)). *If $S \in \mathbb{C}^{2n\times 2n}$ is symplectic and has no eigenvalue of magnitude* 1, *then there is a matrix $Q \in \mathbf{S}-\mathbf{U}$ such that*

$$Q^*SQ = R = \begin{bmatrix} R_1^* & R_2 \\ 0_n & R_1^{-1} \end{bmatrix} \tag{12}$$

where R_1 is lower triangular and R_2R_1 is Hermitian.

Proof. Let $H = (S+I_{2n})(S-I_{2n})^{-1}$. It is easy to verify from the definition that H is Hamiltonian. The eigenvalues of H are of the form $(\lambda+1)/(\lambda-1)$ where $\lambda \in \lambda(S)$. By hypothesis $|\lambda| \neq 1$, so each eigenvalue of H has nonzero real part. Also note that $1 \notin \lambda(H)$. Apply Theorem 1 to H to get $Q \in \mathbf{S}-\mathbf{U}$ and Hamiltonian triangular $T = Q^*HQ$. Equation (12) holds with this value of Q and $R = (T+I_{2n})(T-I_{2n})^{-1}$. It follows directly from the symplectic structure of R [8, Eq. 14], that R_2R_1 is Hermitian. □

A matrix $R \in \mathbf{S}$ with the zero structure of (12) will be called symplectic triangular.

The hypotheses of Theorem 1 and Corollary 2 are not the weakest possible [2], [10], but are sufficient for the purposes of this paper. Not all Hamiltonian matrices have an HSD, nor do all symplectic matrices have an SSD. The matrix J of (1) is both Hamiltonian and symplectic, but obviously cannot be factored as (11) or (12).

As the proof of the corollary suggests, there is a duality between the HSD and the SSD. If k is not an eigenvalue of $H = \mathrm{HAM}(A, G, F)$ and $\mathrm{Re}(k) \neq 0$, then

$$S(k) = (H + \bar{k}I_{2n})(H - kI_{2n})^{-1}$$

is symplectic and

$$H = (kS(k) + \bar{k}I_{2n})(S(k) - I_{2n})^{-1}.$$

By analogy with the QR algorithm [3], we refer to the scalar k as a shift of origin. Observe that $Q^*HQ = T$ is an HSD if and only if $Q^*S(k)Q = R$ is an SSD. The problem of calculating the HSD of H is equivalent to calculating the SSD of $S(k)$.

The HSD and SSD are special cases of the well-known (complex) Schur decomposition (CSD).

THEOREM 3 (Schur decomposition (CSD)). *If $M \in \mathbb{C}^{n\times n}$, there exists a unitary matrix $Q \in \mathbb{C}^{n\times n}$ such that $Q^*MQ = T$ is upper triangular.*

Reversing the last n rows and columns of the matrices in Theorem 1 and Corollary 2 gives Schur decompositions. The force of the theorem and corollary is that Q can be chosen to be symplectic as well as unitary.

2. Outline of the Hamiltonian *QR* iteration. The QR algorithm [3] is a reliable method of calculating the CSD of a general square matrix M. It is based upon the QR iteration:

$$\begin{aligned} &M^{(0)} := M, \\ &\text{FOR } j := 1, 2, 3, \cdots \\ &\qquad \text{Factor } Q^{(j)}R^{(j)} := M^{(j)} \quad Q^{(j)} \text{ unitary}, \\ &\qquad M^{(j+1)} := R^{(j)}Q^{(j)}. \qquad\quad R^{(j)} \text{ triangular} \end{aligned} \tag{13}$$

At the end of each iteration $M^{(j)} = U^{(j)*} M U^{(j)}$ where $U^{(j)}$ is the unitary matrix $U^{(j)} = Q^{(0)} Q^{(1)} \cdots Q^{(j-1)}$. Under mild conditions [16] the iterates $M^{(k)}$ converge to triangular form in the sense that if $i < j$, then

$$\lim_{k \to \infty} m_{ij}^{(k)} = 0.$$

As stated, (13) is not a practical algorithm. The QR factorizations are too expensive. Convergence is slow. Practical algorithms are modifications of (13).

In order to calculate a HSD, one is led to specialize the QR iteration to symplectic matrices. Every symplectic matrix can be factored into the product of a unitary symplectic matrix and a symplectic triangular matrix [2]. The algorithms in § 4 construct the unitary symplectic factor. Hamiltonian matrices do not admit such a factorization. In principle, but not in practice, the HSD of a matrix $H \in \mathbf{H}$ can be obtained through the SSD of $S = (H + \bar{k} I_{2n})(H - k I_{2n})^{-1}$ by the symplectic QR iteration:

$$\begin{aligned} &S^{(0)} := S(k) := (H + \bar{k} I_{2n})(H - k I_{2n})^{-1}, \\ &\text{FOR } j := 1, 2, 3, \cdots \\ &\qquad \text{Factor } Q^{(j)} R^{(j)} = S^{(j)} \quad Q \text{ symplectic unitary,} \\ &\qquad S^{(j+1)} := R^{(j)} Q^{(j)}. \qquad\qquad R \text{ symplectic triangular} \end{aligned} \tag{14}$$

At the end of each iteration $S^{(j)} = U^{(j)*} S(k) U^{(j)}$ where U is the unitary symplectic matrix $U^{(j)} = Q^{(0)} Q^{(1)} \cdots Q^{(j-1)}$. Since (14) is a special case of (13), usually $S^{(j)}$ will converge to symplectic triangular form. In that case $H^{(j)} = U^{(j)*} H U^{(j)}$ converges to Hamiltonian triangular form. Note that the $S^{(j)}$'s are symplectic and the $H^{(j)}$'s are Hamiltonian.

Iteration (14) does not provide a practical algorithm for calculating a HSD. Rounding errors involved in forming $S(k)$ can be disastrous. In § 5 we explain that it is desirable to choose k to be close to an eigenvalue of H. Since $H - k I_{2n}$ is almost singular, it is difficult to form $S(k)$ accurately. Even when $H - k I_{2n}$ is not almost singular, there is a possibility of cancellation error. Furthermore, the symplectic QR factorization, costing $O(n^3)$ flops [2], is prohibitively expensive. The iteration converges too slowly.

The Hamiltonian QR algorithm is a modification of (14) that avoids the above problems. One Hamiltonian QR step corresponds to the similarity transformation

$$H^{(j+1)} := Q^{(j)*} H^{(j)} Q^{(j)}$$

where $Q^{(j)}$ is as in (14) and $H^{(0)} = H$ is the original Hamiltonian matrix. The Hamiltonian QR algorithm performs symplectic similarity transformations directly to H without forming $S(k)$. The cost of each similarity transformation is reduced to $O(n^2)$ flops by reducing H to a Hessenberg-like condensed form. As the iteration converges, the problem deflates to sub-HSD's and sub-CSD's of smaller dimension. Convergence is accelerated by varying k from one iteration to the next.

In broad outline the Hamiltonian QR algorithm is

$H := V^* H V$ where $V \in \mathbf{S} - \mathbf{U}$ is chosen to give H a Hessenberg-like zero structure
Do While H is not nearly Hamiltonian triangular
    Choose $k \in \mathbb{C}$ to be an approximate eigenvalue of H
    $H := Q^* H Q$ where $Q \in \mathbf{S} - \mathbf{U}$ is chosen to make $Q^* S(k)$ symplectic triangular.

3. Hamiltonian–Hessenberg form. A matrix M is said to be upper (lower) Hessenberg if $m_{ij}=0$ for $i>j-1$ $(i<j+1)$. The use of Hessenberg in place of full matrices saves much work in the QR algorithm [3].

The Hamiltonian QR iteration requires a condensed form similar to Hessenberg form. This section describes such a condensed form: Hamiltonian-Hessenberg form. An algorithm is presented that transforms a Hamiltonian matrix to Hamiltonian-Hessenberg form. It is restricted to Hamiltonian matrices $H=\mathrm{HAM}(A, G, F)$ with $\operatorname{rank}(F)=1$. Such Hamiltonian matrices arise naturally from single input control systems. It is shown that the Hamiltonian-Hessenberg form is invariant through a Hamiltonian QR step. The proof suggests that a Hamiltonian QR step is equivalent to two Francis QR steps.

A Hamiltonian matrix H has Hamiltonian-Hessenberg form, if it has the zero structure of a $2n$-by-$2n$ upper Hessenberg matrix with the order of the last n rows and columns reversed. In other words, $H=\mathrm{HAM}(A, G, F)$ is Hamiltonian-Hessenberg when A is lower Hessenberg and F is zero except possibly for f_{nn}.

Let $H=\mathrm{HAM}(A, G, F)$ be a Hamiltonian matrix with $\operatorname{rank}(F)=1$. We now describe an algorithm for reducing H to Hamiltonian-Hessenberg form with a unitary symplectic similarity transformation. An $n=5$ example is used for illustration. Since F is rank 1 and Hermitian, there is a vector $\mathbf{f}\in\mathbb{C}^5$ such that

$$F=\pm\mathbf{f}\mathbf{f}^*.$$

If $\hat{P}^{(1)}=\hat{P}(5, 1, \mathbf{f})$, then

$$H:=\hat{P}^{(1)}H\hat{P}^{(1)}=\begin{bmatrix}\times&\times&\times&\times&\times&\times&\times&\times&\times&\times\\ \times&\times&\times&\times&\times&\times&\times&\times&\times&\times\\ \times&\times&\times&\times&\times&\times&\times&\times&\times&\times\\ \times&\times&\times&\times&\times&\times&\times&\times&\times&\times\\ \times&\times&\times&\times&\times&\times&\times&\times&\times&\times\\ 0&0&0&0&0&\times&\times&\times&\times&\times\\ 0&0&0&0&0&\times&\times&\times&\times&\times\\ 0&0&0&0&0&\times&\times&\times&\times&\times\\ 0&0&0&0&0&\times&\times&\times&\times&\times\\ 0&0&0&0&\times&\times&\times&\times&\times&\times\end{bmatrix}.$$

Set $\hat{P}^{(2)}=\hat{P}(4, 1, A_{*5})$. Replace H with $\hat{P}^{(2)}H\hat{P}^{(2)}$. H now has the zero structure

$$H:=\hat{P}^{(2)}H\hat{P}^{(2)}=\begin{bmatrix}\times&\times&\times&\times&\times&\times&\times&\times&\times&\times\\ \times&\times&\times&\times&\times&\times&\times&\times&\times&\times\\ \times&\times&\times&\times&\times&\times&\times&\times&\times&\times\\ \times&\times&\times&\times&\times&\times&\times&\times&\times&\times\\ 0&0&0&\times&\times&\times&\times&\times&\times&\times\\ 0&0&0&0&0&\times&\times&\times&\times&0\\ 0&0&0&0&0&\times&\times&\times&\times&0\\ 0&0&0&0&0&\times&\times&\times&\times&0\\ 0&0&0&0&0&\times&\times&\times&\times&\times\\ 0&0&0&0&\times&\times&\times&\times&\times&\times\end{bmatrix}.$$

Set $\hat{P}^{(3)} = \hat{P}(3, 1, A_{*4})$. Update H to give

$$H := \hat{P}^{(3)} H \hat{P}^{(3)} = \begin{bmatrix} \times & \times & \times & \times & \times & \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times & \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times & \times & \times & \times & \times & \times \\ 0 & 0 & \times & \times & \times & \times & \times & \times & \times & \times \\ 0 & 0 & 0 & \times & \times & \times & \times & \times & \times & \times \\ 0 & 0 & 0 & 0 & 0 & \times & \times & \times & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \times & \times & \times & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \times & \times & \times & \times & 0 \\ 0 & 0 & 0 & 0 & 0 & \times & \times & \times & \times & \times \\ 0 & 0 & 0 & 0 & \times & \times & \times & \times & \times & \times \end{bmatrix}.$$

Finally, set $\hat{P}^{(4)} = \hat{P}(2, 1, A_{*2})$. Replacing H by $\hat{P}^{(4)} H \hat{P}^{(4)}$ puts H into Hamiltonian-Hessenberg form.

The unitary symplectic similarity transformation relating the original Hamiltonian matrix to the final Hamiltonian-Hessenberg matrix is

$$Q = \hat{P}^{(1)} \hat{P}^{(2)} \hat{P}^{(3)} \hat{P}^{(4)}.$$

Notice that Q is of the form diag (V, V).

The following algorithm summarizes the reduction to Hamiltonian-Hessenberg form.

ALGORITHM 1. Reduction to Hamiltonian-Hessenberg form.
INPUT: $A, G \in \mathbb{C}^{n \times n}$; $\mathbf{f} \in \mathbb{C}^n$
OUTPUT: $V \in \mathbb{C}^{n \times n}$, $V^{-1} = V^*$ chosen so that HAM $(A', G', \mathbf{f}'\mathbf{f}'^*) =$ diag (V^*, V^*) HAM $(A, G, \mathbf{f}\mathbf{f}^*)$ diag (V, V) is Hamiltonian-Hessenberg. A is overwritten by A'; G is overwritten by G'; $\mathbf{f}$ is overwritten by $\mathbf{f}'$.

$\hat{P}^{(1)} := \hat{P}(n, 1, \mathbf{f})$
$A := \hat{P}^{(1)} A \hat{P}^{(1)}$; $G := \hat{P}^{(1)} G \hat{P}^{(1)}$; $\mathbf{f} := \hat{P}^{(1)} \mathbf{f}$
FOR $j = 2, 3, 4, \cdots, n-1$
  $\hat{P}^{(j)} := \hat{P}(n-j+1, 1, A_{*,j+1})$
  $A := \hat{P}^{(j)} A \hat{P}^{(j)}$; $G := \hat{P}^{(j)} G \hat{P}^{(j)}$
$V := I_n$
FOR $j = n-1, n-2, n-3, \cdots, 1$
  $V := \hat{P}^{(j)} V$.

The elementary reflectors $\hat{P}^{(j)}$ can be stored in the unused, i.e., the zeroed, locations of A and f and one auxiliary n-vector. Because of the zero structure of V, multiplication by $\hat{P}^{(j)}$ affects only the leading $(n-j+1)$-by-$(n-j+1)$ principal submatrix so only that submatrix need be calculated. Similarly, the similarity transformation $\hat{P}^{(j)} A \hat{P}^{(j)}$ affects only the first $j+1$ columns of A, so only those columns need to be calculated. With this economy Algorithm 1 requires approximately $10n^3/3$ flops.

To serve as an effective condensed form, the Hamiltonian-Hessenberg form must be invariant through a Hamiltonian QR iteration as described in § 2. The correspondence between the Francis QR iteration and the symplectic QR iteration is enough to prove this, but the following proof, similar to [17, p. 529], is more illuminating.

THEOREM 4 (Invariance of Hamiltonian-Hessenberg form). *Suppose* $H \in \mathbf{H}$ *is Hamiltonian-Hessenberg,* $k \notin \lambda(H)$ *and* $S(k) := (H + \bar{k}I_{2n})(H - kI_{2n})^{-1}$. *If* $Q \in \mathbf{S}-\mathbf{U}$ *and* $R = Q^*S(k)$ *is symplectic triangular, then* $H' = Q^*HQ$ *is Hamiltonian-Hessenberg.*

Proof. Let K be I_{2n} with the order of the last n rows reversed. We will show that $KH'K$ is Hessenberg.

Define $S^{(0)} = KSK$, $Q^{(0)} = KQK$, $R^{(0)} = KRK$, and $H^{(0)} = KHK$. $H^{(0)}$ is upper Hessenberg. $R^{(0)}$ is upper triangular. Define upper Hessenberg matrices $H^{(1)}$, $H^{(2)}$; unitary matrices $Q^{(1)}$, $Q^{(2)}$; and triangular matrices $R^{(1)}$, $R^{(2)}$ by

$$\begin{aligned} Q^{(1)}R^{(1)} &= H^{(0)} + \bar{k}I_{2n}, \\ H^{(1)} &= R^{(1)}Q^{(1)} - \bar{k}I_{2n}, \\ R^{(2)}Q^{(2)} &= H^{(1)} - kI_{2n}, \\ H^{(2)} &= Q^{(2)}R^{(2)} + kI_{2n}. \end{aligned} \tag{15}$$

The first two equations form a single shifted Francis QR step [17, Chap. 8], [13, Chap. 7]. The third equation calls for an RQ (triangular-unitary) factorization instead of the usual QR factorization. The second two equations form what might be called an RQ step. $Q^{(1)}$ is Hessenberg because it is the product of the Hessenberg matrix, $H^{(0)} + kI_{2n}$ and the triangular matrix, $R^{(1)^{-1}}$. Similarly $H^{(1)}$, $Q^{(2)}$ and $H^{(2)}$ are also Hessenberg. Some manipulation of (15) gives

$$[Q^{(1)}Q^{(2)*}][R^{(2)^{-1}}R^{(1)}] = (H^{(0)} + \bar{k}I)(H^{(0)} - kI)^{-1}.$$

So $[Q^{(1)}Q^{(2)*}]$ forms the unitary part and $[R^{(2)^{-1}}R^{(1)}]$ forms the triangular part of the QR factorization of $(H^{(0)} + \bar{k}I_{2n})(H^{(0)} - kI_{2n})^{-1}$. It follows that apart from a unitary diagonal factor

$$\begin{aligned} Q &= KQ^{(1)}Q^{(2)*}K, \\ H' &= KH^{(2)}K. \end{aligned}$$

Thus, $KH'K$ is Hessenberg. □

Equation (15) shows that one step of the Hamiltonian QR iteration is equivalent to a QR step followed by an RQ step. This suggests that one Hamiltonian QR step is equivalent to two Francis QR steps.

4. Hamiltonian *QR* step. Given a Hamiltonian-Hessenberg matrix H, a unitary symplectic matrix U and a scalar $k \in \mathbb{C}$, one step of the Hamiltonian QR iteration consists of replacing H by Q^*HQ and U by UQ where $Q \in \mathbf{S}-\mathbf{U}$ is chosen so that

$$R = Q^*S(k) = Q^*(H + \bar{k}I_{2n})(H - kI_{2n})^{-1} \tag{16}$$

is symplectic triangular. This section presents a practical procedure for performing a Hamiltonian QR step. Neither $S(k)$ nor R is explicitly constructed. The procedure is numerically stable, requires no work arrays and uses $O(n^2)$ flops.

As the next lemma shows, Q can be factored into the product of three simpler symplectic-unitary factors. The Hamiltonian QR step consists of three substeps, one for each factor of Q. The first step resembles a single shifted Francis QR step applied to A. It uses symplectic reflections. The second step uses a symplectic rotation to extend the QR step to include all of $H = \mathrm{HAM}(A, G, F)$. The final step resembles another single shifted Francis QR step. It uses symplectic reflections to return H to Hamiltonian-Hessenberg form.

The formula for Q is given in the following notation. $H = \text{HAM}(A, G, F)$ is Hamiltonian-Hessenberg and $k \notin \lambda(H)$. For simplicity we will assume rank $(F) = 1$, i.e., $F \neq 0_n$. $V \in \mathbb{C}^{n\times n}$ is a unitary matrix such that

$$T = V^*(A^* + \bar{k}I_n) \tag{17}$$

is upper triangular with real diagonal components. Observe that V is upper Hessenberg. Define $\tilde{c}$, $\tilde{s} \in \mathbb{R}$ by

$$\tilde{c} = -t_{nn}/r, \qquad \tilde{s} = f_{nn}/r \tag{18}$$

where

$$r = (t_{nn}^2 + f_{nn}^2)^{1/2}. \tag{19}$$

Note that $\tilde{c}^2 + \tilde{s}^2 = 1$ and

$$[f_{nn} - t_{nn}]\begin{bmatrix} \tilde{c} & \tilde{s} \\ -\tilde{s} & \tilde{c} \end{bmatrix} = [0 \quad r].$$

Similarly define c, $s \in \mathbb{R}$ so that $c^2 + s^2 = 1$ and

$$\begin{bmatrix} c & s \\ -c & s \end{bmatrix}\begin{bmatrix} t_{nn} - f_{nn}V_{*n}^*GV_{*n} \\ f_{nn}(k + \bar{k}) \end{bmatrix} = \begin{bmatrix} * \\ 0 \end{bmatrix}. \tag{20}$$

Define diagonal matrices C, S, $\tilde{C}$, $\tilde{S} \in \mathbb{R}^{n\times n}$ as in (10) by

$$J(c, s, n) = \begin{bmatrix} C & S \\ -S & C \end{bmatrix}, \qquad J(\tilde{c}, \tilde{s}, n) = \begin{bmatrix} \tilde{C} & \tilde{S} \\ -\tilde{S} & \tilde{C} \end{bmatrix}.$$

Let $W \in \mathbb{C}^{n\times n}$ be a unitary matrix chosen so that

$$U = W^*[-S \quad C]\begin{bmatrix} T & V^*GV \\ V^*F & (k + \bar{k})I - V^*T^* \end{bmatrix}\begin{bmatrix} \tilde{S} \\ -\tilde{C} \end{bmatrix} \tag{21}$$

is upper triangular.

LEMMA 5. *The matrix*

$$Q = \begin{bmatrix} V & 0_n \\ 0_n & V \end{bmatrix}\begin{bmatrix} C & S \\ -S & C \end{bmatrix}\begin{bmatrix} W & 0_n \\ 0_n & W \end{bmatrix} \tag{22}$$

satisfies (16).

The proof appears in the appendix.

The Hamiltonian QR step has three parts. The first part performs the similarity transformations

$$A := V^*AV, \quad G := V^*GV, \quad F := V^*FV. \tag{23}$$

The definition of V (17) shows this to be little more than a (single) shifted QR step. We use [13, p. 378] modified to calculate a single QR step instead of a double one. V is expressed as the product $\hat{V}^{(1)}\hat{V}^{(2)}\hat{V}^{(3)} \cdots \hat{V}^{(n-1)}$. Note, T is not explicitly constructed. The lower Hessenberg structure of A is preserved. The matrix F on the left-hand side of (23) is zero except for the trailing 2-by-2 principal submatrix.

The second part of the Hamiltonian QR step consists of the similarity transformation

$$\begin{bmatrix} A^* & G \\ F & -A \end{bmatrix} := \begin{bmatrix} C & S \\ -S & C \end{bmatrix}\begin{bmatrix} A^* & G \\ F & -A \end{bmatrix}\begin{bmatrix} C & S \\ -S & C \end{bmatrix}^*. \tag{24}$$

The value of F on the left-hand side of (24) must have rank 1 so that HAM (A, G, F) can be returned to Hamiltonian-Hessenberg form in the third part of the algorithm. There are two solutions for the scalars c and s that leave rank $(F) = 1$. The extraneous one is $c = \pm 1$, $s = 0$. The other is given by

$$2c[\text{Re}\,(f_{n-1,n}\bar{a}_{n-1,n}) + \text{Re}\,(f_{n-1,n-1}a_{nn})] - s(f_{n-1,n-1}g_{nn} + |a_{n-1,n}|^2) = 0.$$

The notation Re (z) indicates the real part of $z \in \mathbb{C}$. Note that (24) affects only the last rows and columns of A, G and F.

The third part of the Hamiltonian QR step returns HAM (A, G, F) to Hamiltonian-Hessenberg form with the similarity transformations

$$A := W^*AW, \quad G := W^*GW, \quad F := W^*FW.$$

We use Algorithm 1 modified to take advantage of the zero structure of A and F. A is already lower Hessenberg and F is zero except for the trailing 2-by-2 principal submatrix. Reflections of the form $\hat{P}(j, j+1, A_{j-1,*})$ can be used in place of $\hat{P}(n-j+1, 1, A_{*,j+1})$.

In the next algorithm the matrix $U = [U_1 \;\; U_2] \in \mathbb{C}^{n\times 2n}$ represents the unitary symplectic matrix

$$\tilde{U} = \begin{bmatrix} U_1 & U_2 \\ -U_2 & U_1 \end{bmatrix}.$$

ALGORITHM 2. Hamiltonian QR step

INPUT: A, G, $F \in \mathbb{C}^{n\times n}$ such that HAM (A, G, F) is Hamiltonian-Hessenberg; $U \in \mathbb{C}^{n\times 2n}$; a scalar shift $k \in \mathbb{C}$

OUTPUT: A, G, F, U are overwritten by A', G', F', U' where HAM $(A', G', F') = Q^*$ HAM $(A, G, F)Q$ and $U' = UQ$ for Q defined by (22)

Part 1

$\hat{V} := \hat{P}(1, 2, (A + \bar{k}I_n)_{1*})$
$A := \hat{V}A\hat{V}$; $G := \hat{V}G\hat{V}$; $U := U \text{ diag}\,(\hat{V}, \hat{V})$
For $j = 2, 3, 4, 5, \cdots, n-1$
  $\hat{V} := \hat{P}(j, j+1, A_{j-1,*})$
  $A := \hat{V}A\hat{V}$; $G := \hat{V}G\hat{V}$; $U := U \text{ diag}\,(\hat{V}, \hat{V})$
$F := \hat{V}F\hat{V}$

Part 2

$x := 2[\text{Re}\,(f_{n-1,n}\bar{a}_{n-1,n}) + \text{Re}\,(f_{n-1,n-1}, a_{nn})]$
$y := f_{n-1,n-1}g_{nn} + |a_{n-1,n}|^2$
Choose c, $s \in \mathbb{R}$ so that

$$\begin{bmatrix} c & s \\ -s & c \end{bmatrix}\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} * \\ 0 \end{bmatrix},$$

$$[U_{*n} \;\; U_{*,2n}] := [U_{*n} \;\; U_{*,2n}]\begin{bmatrix} c & -s \\ s & c \end{bmatrix}$$

For $j = 1, 2, 3, \cdots, n-1$

$$\begin{bmatrix} a_{nj} \\ g_{nj} \end{bmatrix} := \begin{bmatrix} c & s \\ -s & c \end{bmatrix}\begin{bmatrix} a_{nj} \\ g_{nj} \end{bmatrix},$$

$g_{jn} := g_{nj}$,

$$\begin{bmatrix} \bar{a}_{n-1,n} \\ f_{n-1,n} \end{bmatrix} := \begin{bmatrix} c & s \\ -s & c \end{bmatrix} \begin{bmatrix} \bar{a}_{n-1,n} \\ f_{n-1,n} \end{bmatrix},$$

$$\begin{bmatrix} \bar{a}_{nn} & g_{nn} \\ f_{nn} & -a_{nn} \end{bmatrix} := \begin{bmatrix} c & s \\ -s & c \end{bmatrix} \begin{bmatrix} \bar{a}_{nn} & g_{nn} \\ f_{nn} & -a_{nn} \end{bmatrix} \begin{bmatrix} c & -s \\ s & c \end{bmatrix}$$

Part 3

IF $|f_{n-1,n-1}| > |f_{nn}|$ THEN $\hat{M} := \hat{P}(n, n-1, F_{*,n-1})$
ELSE $\hat{M} := \hat{P}(n, n-1, F_{*n})$
$A := \hat{M}A\hat{M}$; $G := \hat{M}G\hat{M}$; $F := \hat{M}F\hat{M}$; $U := U \operatorname{diag}(\hat{M}, \hat{M})$
FOR $j = n-1, n-2, n-3, \cdots, 2$
$\hat{M} := \hat{P}(j, j-1, A_{*,j+1})$
$A := \hat{M}A\hat{M}$; $G := \hat{M}G\hat{M}$; $U := \tilde{S} \operatorname{diag}(\hat{M}, \hat{M})$

The reflections should be used economically as described in § 1. Also note that A is lower Hessenberg throughout the algorithm, so no component above the codiagonal needs to be explicitly calculated. Only the trailing 2-by-2 principal submatrix of F must be calculated and stored. The other components of F are zero. With these economies Algorithm 2 requires approximately $12n^2$ complex flops.

From $1.5n$ to $2n$ iterations of Algorithm 2 usually transform H to within rounding error of being Hamiltonian triangular. See the discussion of deflation in the next section.

Algorithms 1 and 2 are stable in the presence of rounding errors. Standard error analysis of unitary similarity transformations [2], [17, Chap. 3] can be used to show that using t digit base b arithmetic, rounding errors cause the original Hamiltonian matrix $H = \mathrm{HAM}(A, G, \mathbf{ff}^*)$, the calculated similarity transformation Q, and the final Hamiltonian triangular matrix $T = \mathrm{HAM}(T_1, T_2, 0_n)$ to satisfy

$$\|HQ - QT\| \leqq c_1 b^{1-t} \|H\|$$

and

$$\|\hat{Q}^*\hat{Q} - I_n\| = \|Q^*JQ - I_{2n}\| \leqq c_2 b^{1-t}.$$

The constants c_1 and c_2 may depend upon n and the choice of norm but not upon H or Q. Thus the eigenvalues of T are those of a Hamiltonian matrix that differs from H by a "small" relative error. The first n columns of Q span an invariant subspace of a matrix that differs from H by a "small" relative error.

5. Computational details. Three details remain to be discussed: deflation, choice of shift and ordering of eigenvalues.

As the Hamiltonian QR iteration converges, the codiagonal components of A tend to zero. Once a codiagonal element becomes negligible compared to $\|H\|$, it may be set to zero. So H assumes the form

$$H = \begin{bmatrix} A_{11}^* & A_{21}^* & G_{11} & G_{12} \\ 0 & A_{22}^* & G_{21} & G_{22} \\ 0 & 0 & -A_{11} & 0 \\ 0 & F_{22} & -A_{21} & -A_{22} \end{bmatrix}. \tag{25}$$

Let $A_{11}^* = UT_{11}^*U^*$ be a CSD of A_{11}^*, and let

$$\begin{bmatrix} A_{22}^* & G_{22} \\ F_{22} & -A_{22} \end{bmatrix} \begin{bmatrix} V_1 & V_2 \\ -V_2 & V_1 \end{bmatrix} = \begin{bmatrix} V_1 & V_2 \\ -V_2 & V_1 \end{bmatrix} \begin{bmatrix} T_{22}^* & T_{24} \\ 0 & -T_{22} \end{bmatrix}$$

be a HSD of HAM (A_{22}, G_{22}, F_{22}). Define $Q \in \mathbf{S}-\mathbf{U}$ by

$$Q = \begin{bmatrix} U & 0 & 0 & 0 \\ 0 & V_1 & 0 & V_2 \\ 0 & 0 & U & 0 \\ 0 & -V_2 & 0 & V_1 \end{bmatrix}.$$

A calculation shows that Q^*HQ is in Hamiltonian triangular form. Thus (25) deflates into two smaller problems: a CSD of A_{11}^* and a HSD of HAM (A_{22}, G_{22}, F_{22}).

When there are two consecutive small codiagonal elements, a partial deflation can occur. The situation is illustrated by the following diagram.

$$H = \begin{bmatrix} A^* & G \\ F & -A \end{bmatrix} = \begin{bmatrix} \times & \times & \times & \times & \times & \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times & \times & \times & \times & \times & \times \\ 0 & \bar{d} & \times & \times & \times & \times & \times & \times & \times & \times \\ 0 & 0 & \bar{e} & \times & \times & \times & \times & \times & \times & \times \\ 0 & 0 & 0 & \times & \times & \times & \times & \times & \times & \times \\ 0 & 0 & 0 & 0 & 0 & \times & \times & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \times & \times & -d & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \times & \times & \times & -e & 0 \\ 0 & 0 & 0 & 0 & 0 & \times & \times & \times & \times & \times \\ 0 & 0 & 0 & 0 & \times & \times & \times & \times & \times & \times \end{bmatrix}.$$

The consecutive small components are d and e. Let W be the leading 3-by-3 submatrix of A. Let P be a 3-by-3 unitary matrix such that $P^*(W-kI)$ is lower triangular. Define $Q \in \mathbf{S}-\mathbf{U}$ by

$$Q = \begin{bmatrix} P & 0 & 0 & 0 \\ 0 & I_2 & 0 & 0 \\ 0 & 0 & P & 0 \\ 0 & 0 & 0 & I_2 \end{bmatrix}.$$

Replacing H by Q^*HQ performs a Francis QR step on W, but a_{42} becomes nonzero. The magnitude of the fill-in is $|de|/(|d|^2+|a_{33}-k|^2)^{(1/2)}$. When this quantity is negligible compared to $\|H\|$, it may be ignored.

It is important for computer programs to deflate when possible. If shifts are chosen from the leading 2-by-2 principal submatrix of A_{11} as suggested below, they will not in general be good choices for HAM (A_{22}, G_{22}, F). Convergence will be slow.

Consider the choice of the shift $k \in \mathbb{C}$. So far it has been convenient to assume that $k \notin \lambda(H)$. Now suppose $k \in \lambda(H)$. The fact that $(H-kI_{2n})$ is singular does not cause Algorithms 1 and 2 to fail. There is no attempt to divide by zero or to invert a singular matrix. Returning to the notation of Theorem 4, the QR-RQ sequence of (15) remains defined when $k \in \lambda(H)$. Both $(H^{(0)}+\bar{k}I_{2n})$ and $(H^{(1)}-kI_{2n})$ are singular [1], [7], [10]. Hence some diagonal element of $R^{(1)}$ and of $R^{(2)}$ must be zero. If the problem has been deflated as described above, then $H^{(0)}$ is unreduced, i.e., all codiagonal elements are nonzero. It follows that $r_{nn}^{(1)} = 0$ and $r_{11}^{(2)} = 0$. This gives H' the zero structure

$$H' = \begin{bmatrix} \times & \times & \times & \times & \times & \times & \times & \times \\ 0 & \times & \times & \times & \times & \times & \times & \times \\ 0 & \times & \times & \times & \times & \times & \times & \times \\ 0 & 0 & \times & \times & \times & \times & \times & \times \\ 0 & 0 & 0 & 0 & \times & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \times & \times & \times & 0 \\ 0 & 0 & 0 & 0 & \times & \times & \times & \times \\ 0 & 0 & 0 & \times & \times & \times & \times & \times \end{bmatrix}.$$

Furthermore, $k = \overline{h'_{11}} = -h'_{n+1,n+1}$. No difficulty arises when $k \in \lambda(H)$. In fact it is desirable to have $k \in \lambda(H)$, because the problem deflates immediately.

This suggests that the shift $k \in \mathbb{C}$ be chosen to be h_{11}, since as the iteration converges, h_{11} converges to an eigenvalue. A better strategy is to choose k to be an eigenvalue of the leading 2-by-2 principal submatrix of H.

Now consider the problem of ordering eigenvalues. Let $H = \mathrm{HAM}(A, G, F)$ have Hamiltonian-Schur form

$$\begin{bmatrix} A^* & G \\ F & -A \end{bmatrix} \begin{bmatrix} Q_1 & Q_2 \\ -Q_2 & Q_1 \end{bmatrix} = \begin{bmatrix} Q_1 & Q_2 \\ -Q_2 & Q_1 \end{bmatrix} \begin{bmatrix} T_1^* & T_2 \\ 0 & -T_1 \end{bmatrix}.$$

If Q_2 is nonsingular, then $X = -Q_1 Q_2^{-1}$ satisfies the algebraic Riccati equation (5). The eigenvalues of $(A - FX)$ are the eigenvalues of $-T_1^*$. For most purposes, X must be chosen so that $(A - FX)$ is stable, i.e., $\lambda(A - FX) \subset \mathbb{C}^-$. When shifts are chosen as described above, the eigenvalues may appear in any order along the diagonal of T. It is necessary to use a unitary symplectic similarity transformation to order the eigenvalues so that $\lambda(-T_1) \subset \mathbb{C}^-$.

Suppose $\lambda \in \lambda(-T_1)$ has positive real part. We must interchange it with the eigenvalue $-\bar{\lambda}$ of T_1^*. The method of [14] can be used to find a unitary matrix $V \in \mathbb{C}^{n \times n}$ such that $-\bar{\lambda}$ is the (n, n) entry of $V^* T_1^* V$. Replace T by $\mathrm{diag}(V^*, V^*) T \,\mathrm{diag}(V, V)$. Now we must interchange the (n, n) entry of $-T_1$ with that of T_1^*. Choose $c, s \in \mathbb{R}$ so that

$$\begin{bmatrix} c & s \\ -s & c \end{bmatrix} \begin{bmatrix} t_{n,2n} \\ \lambda + \bar{\lambda} \end{bmatrix} = \begin{bmatrix} * \\ 0 \end{bmatrix}.$$

Let $K = J(n, c, s)$ (equation (9)). The interchange is effected by replacing T by KTK^*.

6. Double Hamiltonian QR step with implicit shifts. Complex arithmetic requires twice the storage and at least four times the work of real arithmetic. Furthermore, rounding errors will cause quantities which should appear in complex conjugate pairs to fail to do so. It is preferable to use real arithmetic when possible. When the original data A, G and F are real, complex arithmetic can be avoided by combining two successive Hamiltonian QR steps into one. The Francis implicit QR algorithm employs a similar trick [3].

The reasoning and numerical techniques used for the double implicit step are similar to those used for the simpler algorithm of §§ 4 and 5. This section outlines the differences. A full account appears in [2].

Observe first that when Algorithm 1 is applied to a real Hamiltonian matrix, all intermediate quantities and the final Hamiltonian-Hessenberg matrix are real. Algorithm 1 does not need to be modified to avoid complex arithmetic.

Apply Algorithm 2 with shift k to the real Hamiltonian-Hessenberg matrix $H = \mathrm{HAM}(A, G, F)$ to get a not necessarily real Hamiltonian-Hessenberg matrix $H' = Q'^* H Q'$. Apply Algorithm 2 with shift $\bar{k}$ to H' to get a Hamiltonian-Hessenberg matrix $H'' = Q''^* H' Q''$. Define $\tilde{S}(k)$ by

$$\tilde{S}(k) = [(H + kI_{2n})(H + \bar{k}I_{2n})][(H - kI_{2n})(H - \bar{k}I_{2n})]^{-1}.$$

Set $Q = Q'Q''$. It is easy to show that

$$R = Q^* \tilde{S}(k) \tag{26}$$

is symplectic triangular. Note that $S(k)$ is real, so by analogy with the general QR (orthogonal-triangular) factorization Q and R may be chosen to be real [17]. Hence H'' is real.

To obtain H'' directly from H we need to calculate the similarity transformation $Q^* H Q$ where $Q \in \mathbb{R}^{2n \times 2n}$ is orthogonal and satisfies (26). Using the method of Lemma 5, it can be shown that

$$Q = \mathrm{diag}(V, V) K \, \mathrm{diag}(W, W).$$

The matrices V and $W \in \mathbb{R}^{n \times n}$ are orthogonal. K is the product of a Householder symplectic and three Jacobi symplectic matrices.

The matrix V is chosen so that

$$L = V^T[(A + \bar{k}I_n)(A + kI_n)]$$

is upper triangular. The first part of the double implicit step is essentially a double implicit Francis QR step applied to A.

The matrix K plays the role of the rotation in Lemma 5, but its construction is more complicated. Define

$$H_+ = \mathrm{diag}(V^*, V^*)(H + kI_{2n})(H - \bar{k}I_{2n}) \, \mathrm{diag}(I_n, V),$$

$$H_- = (H - kI_{2n})(H - \bar{k}I_{2n}) \, \mathrm{diag}(I_n, V),$$

and

$$N = KH_+H_-^{-1}.$$

The relationship between H_+, H_-, N, $\tilde{S}(k)$ and Q is

$$Q^* \tilde{S}(k) = \mathrm{diag}(W^*, W^*) KH_+H_-^{-1} = \mathrm{diag}(W^*, W^*) N.$$

So K must be chosen to make $(KH_+H_-^{-1})$ block upper triangular with n-by-n blocks. The trailing 2-by-2 principal submatrices of the n-by-n blocks of H_+ and H_- determine K.

The final orthogonal matrix W is chosen to return H to Hamiltonian-Hessenberg form. Again we use Algorithm 1 modified to take advantage of the zero structure of A, G and F. This time reflections of the form $\hat{P}(j, j-2, A_{j-1,*})$ are used.

7. Numerical examples and conclusions. The double Hamiltonian QR algorithm was implemented in FORTRAN using DOUBLE PRECISION arithmetic on Cornell's IRM370/168. We also wrote a similar program to calculate a real Schur decomposition [3]. We used both algorithms to solve several algebraic Riccati equations as described in the introduction.

In order to minimize the effect of programming style on our observations, we used our own implementation of the QR algorithm in preference to one of the fine programs from EISPACK [12]. By using the BLAS [9] and several subroutines written in the

spirit of the BLAS, we arranged for many of the innermost loops of both algorithms to be executed by the same subroutines. Our QR algorithm ran somewhat slower than EISPACK's, but the difference could be attributed almost entirely to extra subroutine calls. When subroutines were expanded in line, the execution times of our QR algorithm and EISPACK's differed by only a few percent.

Test problems were constructed by performing random orthogonal similarity transformations of the following 5-by-5 patterns.

$$A=\begin{bmatrix} -1 & 1 & 1 & 1 & 1 \\ 0 & -2 & 1 & 1 & 1 \\ 0 & 0 & -3 & 1 & 1 \\ 0 & 0 & 0 & -4 & 1 \\ 0 & 0 & 0 & 0 & -5 \end{bmatrix}, \quad F=\begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix},$$

$$G=\begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 6 \end{bmatrix}, \quad X=\begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}.$$

Similar problems were run with dimensions $n=5$, 10, 20 and 30.

Both algorithms produced solutions of comparable accuracy. Table 1 compares execution times. For small problems the two methods used approximately the same amount of time. For larger problems, the advantage of the Hamiltonian-Schur decomposition is unmistakable.

TABLE 1
Execution time of Francis QR algorithm versus Hamiltonian QR algorithm.

	milliseconds			
dimension	5	10	20	30
Francis QR algorithm	47	226	1474	4639
Hamiltonian QR algorithm	49	181	969	2681

The Francis QR algorithm requires two $2n$-by-$2n$ arrays for a total of $8n^2$ storage. The Hamiltonian QR algorithm requires about $7n^2/2$ storage.

Both the Francis QR algorithm and the Hamiltonian QR algorithm produce the same invariant subspace information. For problems of size greater than about 20, the Hamiltonian QR algorithm requires significantly less work and storage. It is restricted to problems that arise from single input control systems. However, the restriction is necessary only at the first reduction to condensed form, so some modification either in the condensed form or in the algorithm by which it is obtained may lead to a more generally applicable algorithm.

Appendix A. Proof of Lemma 5. In the notation of § 3, the matrix

$$Q=\begin{bmatrix} V & 0_n \\ 0_n & V \end{bmatrix}\begin{bmatrix} C & S \\ -S & C \end{bmatrix}\begin{bmatrix} W & 0_n \\ 0_n & W \end{bmatrix} \tag{27}$$

satisfies (16).

Proof. Define $L \in \mathbb{C}^{2n\times 2n}$ in n-by-n blocks by

$$L = \begin{bmatrix} L_1 & L_2 \\ L_3 & L_4 \end{bmatrix} = \begin{bmatrix} A^* - kI_n & G \\ F & -A - kI_n \end{bmatrix} \begin{bmatrix} I_n & 0_n \\ 0_n & V \end{bmatrix} \begin{bmatrix} \tilde{C} & \tilde{S} \\ -\tilde{S} & \tilde{C} \end{bmatrix}. \tag{28}$$

The choice of V in (17) and $\tilde{c}$ and $\tilde{s}$ in (18) make $L_3 = 0_n$ and L_4 lower triangular.

Define $M \in \mathbb{C}^{2n\times 2n}$ by

$$M = \begin{bmatrix} V^* & 0_n \\ 0_n & V^* \end{bmatrix} \begin{bmatrix} A^* + \bar{k}I_n & G \\ F & -A + \bar{k}I_n \end{bmatrix} \begin{bmatrix} I_n & 0_n \\ 0_n & V \end{bmatrix} \begin{bmatrix} \tilde{C} & \tilde{S} \\ -\tilde{S} & \tilde{C} \end{bmatrix}.$$

Partition M into n-by-n blocks and use (17) to get

$$M = \begin{bmatrix} M_1 & M_2 \\ M_3 & M_4 \end{bmatrix} = \begin{bmatrix} T & V^*GV \\ V^*F & (k+\bar{k})I_n - V^*T^* \end{bmatrix} \begin{bmatrix} \tilde{C} & \tilde{S} \\ -\tilde{S} & \tilde{C} \end{bmatrix}. \tag{29}$$

Recall that $\tilde{S}$ is zero except for s_{nn}, C is diagonal, T is upper triangular, V is upper Hessenberg, and F is zero except for f_{nn}. Hence M_1 is upper triangular and M_4 is lower Hessenberg. M_3 is zero except for the last two components of the nth column. An $n = 5$ example of the zero structure is

$$M = \begin{bmatrix} M_1 & M_2 \\ M_3 & M_4 \end{bmatrix} = \begin{bmatrix} \times & \times & \times & \times & \times & \times & \times & \times & \times & \times \\ 0 & \times & \times & \times & \times & \times & \times & \times & \times & \times \\ 0 & 0 & \times & \times & \times & \times & \times & \times & \times & \times \\ 0 & 0 & 0 & \times & \times & \times & \times & \times & \times & \times \\ 0 & 0 & 0 & 0 & \times & \times & \times & \times & \times & \times \\ 0 & 0 & 0 & 0 & 0 & \times & \times & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \times & \times & \times & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \times & \times & \times & \times & 0 \\ 0 & 0 & 0 & 0 & \times & \times & \times & \times & \times & \times \\ 0 & 0 & 0 & 0 & \times & \times & \times & \times & \times & \times \end{bmatrix}. \tag{30}$$

Solving for m_{nn}, $m_{2n-1,n}$ and $m_{2n,n}$ in (29) gives

$$\begin{aligned} m_{nn} &= t_{nn}\tilde{c} - V_{*n}^* G V_{*n} \tilde{s}, \\ m_{2n-1,n} &= v_{n,n-1}(f_{nn}\tilde{c} + t_{nn}\tilde{s}), \\ m_{2n,n} &= v_{nn}(f_{nn}\tilde{c} + t_{nn}\tilde{s}) - (k+\bar{k})\tilde{s}. \end{aligned} \tag{31}$$

The choice of $\tilde{c}$ and $\tilde{s}$ in (18) makes $m_{2n-1,n} = 0$ and $m_{2n,n} = -(k+\bar{k})\tilde{s}$. The (9, 5) component of (30) is zero.

Using (18) to simplify (31) further gives

$$\begin{aligned} m_{nn} &= (t_{nn}^2 - f_{nn} V_{*n}^* G V_{*n})/r, \\ m_{2n,n} &= (k+\bar{k}) f_{nn}/r \end{aligned}$$

where r is given by (19). Define N by

$$N = \begin{bmatrix} N_1 & N_2 \\ N_3 & N_4 \end{bmatrix} = \begin{bmatrix} W^* & 0_n \\ 0_n & W^* \end{bmatrix} \begin{bmatrix} C & S \\ -S & C \end{bmatrix} \begin{bmatrix} M_1 & M_2 \\ M_3 & M_4 \end{bmatrix}. \tag{32}$$

The definition (20) of c and s shows that $N_3 = 0_n$, and, from definition (21), N_4 is lower triangular.

With Q given by (27), equations (28) through (32) combine to show

$$NL^{-1} = Q^*S(k) = Q^*(H + \bar{k}I_{2n})(H - kI_{2n})^{-1}.$$

Recall N and L^{-1} are block upper triangular with n-by-n blocks. Hence $Q^*S(k)$ is block upper triangular. The (2, 2) block of $Q^*S(k)$ is lower triangular since the (2, 2) blocks of N and L^{-1} are. Thus $Q^*S(k)$ is symplectic triangular.

Acknowledgments. I am grateful to Charles Van Loan for his guidance and for helpful comments on earlier drafts of this paper.

REFERENCES

[1] N. BURGOYNE AND R. CUSHMAN, *Normal forms for real linear Hamiltonian systems*, The 1976 Ames Research Center (NASA) Conference of Geometric Control Theory, C. Martin and R. Hermann, eds., Math. Sci. Press, Brookline, MA, 1976, pp. 483-529.

[2] R. BYERS, *Hamiltonian and symplectic algorithms for the algebraic Riccati equation*, Doctoral dissertation, Cornell Univ., Ithaca, NY, 1983.

[3] J. FRANCIS, *The QR transformation—Part* II, Comput. J., 5 (1962), pp. 332-345.

[4] V. KUCERA, *A contribution to matrix quadratic equations*, IEEE Trans. Automat. Control, AC-17 (1972), pp. 344-347.

[5] H. KWAKERNAAK AND R. SIVAN, *Linear Optimal Control Systems*, Wiley-Interscience, New York, 1972.

[6] A. LAUB, *A Schur method for solving algebraic Riccati equations*, IEEE Trans. Automat. Control, AC-24 (1979), pp. 913-925.

[7] A. LAUB AND K. MEYER, *Canonical forms for symplectic and Hamiltonian matrices*, Celestial Mech., 9 (1974), pp. 213-238.

[8] A. LAUB, *Schur techniques in invariant imbedding methods for solving two point boundary value problems*, 21st IEEE Conference Decision and Control, Orlando, Vol. 1, Institute of Electrical and Electronics Engineers, Inc., New York, 1982, pp. 56-61.

[9] C. LAWSON, R. HANSON AND D. KINCAID, *Basic linear algebra subprograms for FORTRAN usage*, ACM Trans. Math. Software, 3 (1979), pp. 308-323.

[10] C. PAIGE AND C. VAN LOAN, *A Schur decomposition for Hamiltonian matrices*, Linear Algebra Appl., 41 (1981), pp. 11-32.

[11] J. POTTER, *Matrix quadratic solutions*, SIAM J. Appl. Math., 14 (1966), pp. 496-501.

[12] B. SMITH, J. BOYLE, B. GARBOW, Y. IKEBE, V. KEMA AND C. MOLER, *EISPACK Guide*, Springer-Verlag, New York, 1974.

[13] G. STEWART, *Introduction to Matrix Computations*, Academic Press, New York, 1973.

[14] G. STEWART, HQR3 *and* EXCHNG: *FORTRAN subroutines for calculating and ordering the eigenvalues of a real upper Hessenberg matrix*, ACM Trans. Math. Software, 2 (1975), pp. 275-280.

[15] C. VAN LOAN, *A symplectic method for approximating all the eigenvalues of a Hamiltonian matrix*, Numerical Analysis Report No. 71, Dept. Mathematics, Univ. Manchester, Manchester, England, 1982.

[16] D. WATKINS, *Understanding the QR algorithm*, SIAM Rev., 24 (1982), pp. 427-440.

[17] J. WILKINSON, *The Algebraic Eigenvalue Problem*, Clarendon Press, Oxford, 1965.

SIAM J. SCI. STAT. COMPUT.
Vol. 7, No, 1, January 1986

016

FRONT TRACKING APPLIED TO RAYLEIGH–TAYLOR INSTABILITY*

J. GLIMM†, O. MCBRYAN‡, R. MENIKOFF§ AND D. H. SHARP§

Abstract. A numerical solution of the two-fluid incompressible Euler equation is used to study the Rayleigh–Taylor instability. The solution is based on the method of front tracking, which has the distinguishing feature of being a predominantly Eulerian method in which sharp interfaces are preserved with zero numerical diffusion. In this paper validation of the method is obtained by comparison with existing numerical solutions based on conformal mapping. An initial study of heterogeneity is presented.

Key words. front tracking, hydrodynamic instability, Rayleigh–Taylor instability

1. Introduction. The Rayleigh–Taylor problem is the study of the fingering instability of an interface between two fluids of different densities; it occurs when the light fluid pushes the heavy fluid. In its simplest formulation the governing equations are the two-fluid, two-dimensional Euler equations; these equations appear to be ill-posed unless formulated with the interface in a space of analytic functions. This can be seen from the fact that the unstable perturbations of wave number k on an initially flat interface have exponential growth rates proportional to $k^{1/2}$. Thus, at least within the framework of the linearized theory, the class of C^∞ interface perturbations, which are characterized as having polynomially decaying amplitudes as a function of k, are not mapped into the space of distributions under time evolution for any $t > 0$.

There is a complex phenomenology associated with this problem including the formation of spikes and bubbles, the development of Helmholtz instability on the side of the spikes, competition between bubbles leading to their amalgamation, formation of droplets, entrainment and turbulent mixing, and a possible chaotic limit with a fractalized interface.

A variety of physical factors influence the development of Rayleigh–Taylor instability. These factors are reflected in a modification of the incompressible Euler equations and include surface tension, viscosity, compressibility, three-dimensional effects, material properties, the equation of state and a variety of forms of heterogeneity.

In many cases of interest, these extra physical factors occur multiplied by a small parameter. In other words they occur on small length scales, inaccessible to feasible calculations. There are two obstacles to a correct calculation in such circumstances. The small parameter and small length scale effects must somehow be included by mathematical or computational modeling. At the same time, it is necessary to avoid the incorrect simulation of physical effects by numerical artifacts. In particular convergence under mesh refinement, while a necessary aspect of validation for such problems, is not sufficient. Validation requires quantitative agreement with an independently correct calculation, analytic solution or laboratory experiment.

The computational strategies which have been developed for this problem fall into two groups: special purpose codes and general purpose codes. The special purpose

* Received by the editors March 29, 1984, and in revised form December 3, 1984.

† Courant Institute of Mathematical Sciences, New York University, New York, New York 10012. The work of this author was supported in part by the National Science Foundation under grant DMS-8312229 and by the U.S. Department of Energy under grant DEA 0276ER03077.

‡ Courant Institute of Mathematical Sciences, New York University, New York, New York 10012. The work of this author was supported in part by the the U.S. Department of Energy under grant DEA 0276ER03077. This author is an Alfred P. Sloan Foundation Fellow.

§ Los Alamos National Laboratory, Los Alamos, New Mexico 87545. The work of this author was supported by the U.S. Department of Energy.

codes are ones not readily adaptable to include the variety of physical factors mentioned above. Outside the scope of these codes are effects associated with physical degrees of freedom in the interior of the fluid, such as variable mass or vorticity distributions or diffusion phenomena. This narrower scope permits analytic simplifications, which are utilized to a maximum degree. Two notable examples of such codes have been developed by Menikoff and Zemach [1], [2] and by Baker, Meiron, and Orszag [3]. The Menikoff-Zemach code is based on conformal mappings. In the case of an infinite density ratio and two fluids in an infinite strip, a time-dependent conformal mapping takes the region occupied by the heavy fluid into an infinite half strip. In this half strip, the known Green's function for Laplace's equation is used to express the interface velocity as a quadrature.

The Baker, Meiron and Orszag code is based on boundary integral techniques, in which the velocity potential is expressed as an integral over a dipole sheet distributed over the fluid interface. Coupled Fredholm integral equations can then be derived which determine the strength of the dipole sheet, and its time development.

The strong point of these special purpose codes is accuracy and speed. Thus they can be used to validate general purpose codes. Of course, these codes have been applied in their own right to study several interesting questions. For example, the code of Baker et al [3] has been used to study the Taylor instability of a thin fluid layer [4], and both codes have been used to confirm results on rising plane bubbles originally obtained by Birkhoff and Carter [5].

There have been numerous calculations of Rayleigh-Taylor instability using codes which solve the full (two-dimensional) Euler or Navier-Stokes equations. Notable examples include the work of F. Harlow and J. Welch [6], B. J. Daly [7], [8], W. P. Crowley [9], J. R. Freeman, M. J. Clauser and S. L. Thompson [10], K. A. Meyer and P. J. Blewett [11], [12] as well as several papers which compute Taylor instability of laser driven fusion targets [13].

The goal of front tracking is to achieve the accuracy of a special purpose calculation within the context of a general purpose method. In this paper we report on the degree to which this goal has been realized for the Rayleigh-Taylor problem. The main idea of front tracking is to introduce as a computational degree of freedom an interface or front consisting of a (co-dimension one) set of curves, composed for example of piecewise linear bonds joining vertices, or nodes on the front. This front is then propagated, using the velocity and acceleration fields of the fluid in the present case of a fluid interface discontinuity. In the case of a shock or other type of fluid wave, the front is propagated with the corresponding wave speed. From this point of view, front tracking in the Rayleigh-Taylor problem is a mixed Eulerian-Lagrangian approach, with the front being a Lagrangian degree of freedom and all other grid points being Eulerian. Because of the assumption of incompressibility, the equations are elliptic. In particular it is necessary to solve elliptic equations at each time step for the pressure and stream function. The unequal density leads to elliptic equations which are singular exactly on the interface, either in their coefficients or in their source terms or both. Accurate solution of such equations requires a special grid construction, whereby the grid lines are displaced (aligned) so as to coincide with the interface.

There are a variety of adaptive grid strategies available. Our grid strategy was designed to be robust and effective even in a deep nonlinear regime. This construction is outlined in § 3. Another method, not used here, is elliptic grid generation, whereby a subsidiary elliptic equation is solved to give the grid lines and coordinate system for the original problem. Usually the grids constructed by this method map the interface onto a grid line. This construction provides advantages over our grids for not too badly

distorted interfaces, as there is a smooth adjustment of the grid to the interface location over several mesh blocks. However it appeared to the authors that this elliptic grid construction would have three drawbacks, associated with the constraint of mapping the interface onto a grid line. First, in the deep nonlinear regime, the interface is highly fingered and so a mapping onto a grid line would lead to distorted grids inside the finger region. Second, the solution of an extra elliptic equation is presumably expensive. Third, in the case of interface breakup (droplet formation) it is topologically impossible to map the interface onto a single grid line. The mesh alignment grid construction used here maps finite element boundaries (horizontal, vertical or diagonal grid line segments) onto the interface. The allowed extra freedom in choice of grid mappings gives the desired properties of robustness and effectiveness even in the deep nonlinear regime.

In § 2, we cast the two-fluid Euler equations for incompressible flow into the form used for our numerical computation. In § 3, we discuss the solution method. Section 4 is a preliminary validation study. Section 5 presents calculations run with finite density ratio. Section 6 gives examples of the effect of heterogeneity on the solution. Conclusions are in § 7. Preliminary announcements of some of the results presented in this paper were made in [14]-[16].

2. Derivation of equations. The calculation solves the incompressible two-fluid Euler equations in two dimensions. In this section we cast the equations into the form which is implemented numerically. We introduce the following notation. Let $\hat{z}$ be a unit vector perpendicular to the $x-y$ plane of the fluid flow. Let $\hat{s}$ be the unit tangent to the interface such that as one traverses the interface the light fluid is on the left and the heavy fluid is on the right. Let $\hat{n}$ be the unit normal to the interface oriented so that $\hat{n}$ points from the heavy fluid to the light fluid. We denote by f^+ and f^- the value of a discontinuous function on the left and right of the interface. Let $[f]=f^+-f^-$ denote the jump at the discontinuity and $\bar{f}=\frac{1}{2}(f^++f^-)$ the mean value.

2.1. The velocity equation. We assume each fluid is incompressible and irrotational. As a result, the velocity field $\mathbf{v}$ is globally divergence free and, except on the interface which separates the two fluids, it is irrotational. Let χ_w denote the characteristic function of the heavy fluid region. We define a delta function concentrated on the interface by

$$\delta_I(x, y, t)=-\hat{n}\cdot\nabla\chi_w(x, y, t)=\int ds\,\delta(x-x(s, t))\delta(y-y(s, t)) \tag{2.1}$$

where $x(s, t)$, $y(s, t)$ are the co-ordinates of a point on the interface parameterized by arc length s.

The vorticity has the form

$$\gamma\delta_I(x, y, t)\hat{z}, \tag{2.2}$$

where the vorticity density on the interface is

$$\gamma=-\hat{s}\cdot[\mathbf{v}]. \tag{2.3}$$

In this notation, the equations for $\mathbf{v}$ are

$$\nabla\cdot\mathbf{v}=0, \tag{2.4}$$

$$\nabla\times\mathbf{v}=\gamma\delta_I\hat{z}. \tag{2.5}$$

Introducing a streamfunction Ψ, we have

$$\mathbf{v} = \nabla \times (\Psi \hat{z}), \tag{2.6}$$

$$\Delta \Psi = -\gamma \delta_I. \tag{2.7}$$

2.2. The pressure equation. The equation for the pressure P expresses balance of forces. Introducing the acceleration $\mathbf{a}$, density ρ, gravitational potential $\Phi = -g\rho y$, and total time derivative D/Dt, we have

$$\mathbf{a} = \frac{D\mathbf{v}}{Dt} = \frac{\partial \mathbf{v}}{\partial t} + (\mathbf{v} \cdot \nabla)\mathbf{v} = -\frac{1}{\rho}\nabla P - \nabla\left(\frac{\Phi}{\rho}\right). \tag{2.8}$$

Since

$$\nabla \cdot \frac{\partial \mathbf{v}}{\partial t} = \frac{\partial \nabla \cdot \mathbf{v}}{\partial t} = 0,$$

$$-\nabla \cdot \left(\frac{1}{\rho}\right)\nabla P = \nabla \cdot (\mathbf{v} \cdot \nabla)\mathbf{v}. \tag{2.9}$$

This equation is the pressure equation, and the right-hand side is its source term. It is necessary to interprete these equations carefully using distribution derivatives to account for the discontinuities at the interface.

2.3. Singularities in the pressure source term. The ∇ is a derivative to be taken in the distribution sense, i.e. including delta functions which may result from differentiation of jump discontinuities across the interface. The expression $\nabla \cdot (\mathbf{v} \cdot \nabla)\mathbf{v}$ contains products of distributions. The general theory does not guarantee that these products are well defined. To check that they are meaningful, it is necessary to determine the singularities in $\nabla \cdot (\mathbf{v} \cdot \nabla)\mathbf{v}$.

Near the interface $\hat{s}$ and $\hat{n}$ define orthogonal curvilinear co-ordinates. It is convenient to use the normal and tangential components of vector fields which we denote with the subscripts n and s. From the geometry the derivatives of $\hat{s}$ and $\hat{n}$ are given by

$$\frac{\partial \hat{s}}{\partial s} = -\frac{\hat{n}}{R}, \qquad \frac{\partial \hat{n}}{\partial s} = \frac{\hat{s}}{R},$$

$$\frac{\partial \hat{s}}{\partial n} = 0, \qquad \frac{\partial \hat{n}}{\partial n} = 0, \tag{2.10}$$

$$\frac{\partial \hat{s}}{\partial t} = \frac{\partial v_n}{\partial s}\hat{n}, \qquad \frac{\partial \hat{n}}{\partial t} = -\frac{\partial v_n}{\partial s}\hat{s},$$

where R is the radius of curvature, and is positive if the normal points away from the side where the center of curvature is located. We write the velocity as

$$\mathbf{v} = v_s \hat{s} + v_n \hat{n}. \tag{2.11}$$

Denoting an unspecified continuous function by "continuous," we have

$$v_n = \text{continuous}, \tag{2.12}$$

$$v_s = \gamma \chi_w + \text{continuous}. \tag{2.13}$$

We extend a function defined on the interface to a region about the interface in the

natural way, $f(s, n) = f(s)$. From (2.10) and (2.11) the velocity derivatives are

$$\frac{\partial \mathbf{v}}{\partial s} = \left(\frac{\partial v_s}{\partial s} + \frac{v_n}{R}\right)\hat{s} + \left(\frac{\partial v_n}{\partial s} - \frac{v_s}{R}\right)\hat{n}, \tag{2.14}$$

$$\frac{\partial \mathbf{v}}{\partial n} = \frac{\partial v_s}{\partial n}\hat{s} + \frac{\partial v_n}{\partial n}\hat{n}. \tag{2.15}$$

Substituting

$$\nabla = \hat{s}\,\partial_s + \hat{n}\,\partial_n \tag{2.16}$$

into (2.4) and (2.5) and using (2.14) and (2.15), we obtain

$$\frac{\partial v_n}{\partial n} = -\frac{\partial v_s}{\partial s} - \frac{v_n}{R}, \tag{2.17}$$

$$\frac{\partial v_s}{\partial n} = -\gamma\delta_I + \frac{\partial v_n}{\partial s} - \frac{v_s}{R}. \tag{2.18}$$

Furthermore evaluation of jump discontinuities commutes with tangential derivatives, so we see from (2.12) and (2.13) and a piecewise smoothness assumption on the solution that

$$\frac{\partial v_n}{\partial s} = \text{continuous}, \tag{2.19}$$

$$\frac{\partial v_s}{\partial s} = \frac{\partial \gamma}{\partial s}\chi_w + \text{continuous}. \tag{2.20}$$

Substitution of (2.19) and (2.20) into (2.17) and (2.18) gives expressions for the normal derivatives of the velocity components,

$$\frac{\partial v_n}{\partial n} = -\frac{\partial \gamma}{\partial s}\chi_w + \text{continuous}, \tag{2.21}$$

$$\frac{\partial v_s}{\partial n} = -\gamma\delta_I - \frac{\gamma\chi_w}{R} + \text{continuous}. \tag{2.22}$$

Also, note that

$$v_s^2 = 2\bar{v}_s\gamma\chi_w + \text{continuous}. \tag{2.23}$$

Next, we consider

$$\begin{aligned}(\mathbf{v}\cdot\nabla)\mathbf{v} &= \left(v_n\frac{\partial}{\partial n} + v_s\frac{\partial}{\partial s}\right)(v_n\hat{n} + v_s\hat{s}) \\ &= \left(v_n\frac{\partial v_n}{\partial n} + v_s\frac{\partial v_n}{\partial s} - \frac{v_s^2}{R}\right)\hat{n} + \left(v_n\frac{\partial v_s}{\partial n} + v_s\frac{\partial v_s}{\partial s} + \frac{v_s v_n}{R}\right)\hat{s} \\ &= \left(\left\{\gamma\frac{\partial v_n}{\partial s} - v_n\frac{\partial \gamma}{\partial s} - \frac{2\gamma\bar{v}_s}{R}\right\}\chi_w + \text{continuous}\right)\hat{n} \\ &\quad + \left(-v_n\gamma\delta_I + \frac{\partial}{\partial s}(\gamma\bar{v}_s)\chi_w + \text{continuous}\right)\hat{s}.\end{aligned} \tag{2.24}$$

Thus, $(\mathbf{v}\cdot\nabla)\mathbf{v}$ is well defined as a distribution. Basically, the possibly singular term $\chi(\partial/\partial s)\chi$ is not $\chi\delta$ which is ill-defined but 0 because $(\partial/\partial s)\chi_w = 0$.

Finally, the pressure source term is

$$\nabla \cdot \mathbf{a} = \nabla \cdot (\mathbf{v} \cdot \nabla)\mathbf{v} = 2\gamma \left(\frac{\bar{v}_s}{R} - \frac{\partial v_n}{\partial s} \right) \delta_I + \text{continuous} \cdot \chi_w + \text{continuous}. \tag{2.25}$$

Away from the interface, because the fluid is incompressible, equation (2.4), the pressure source term reduces to $\sum (\partial v_j/\partial x_i)(\partial v_i/\partial x_j)$. Thus it depends on two derivatives of the stream function and not three. The acceleration has no δ function singularities (physically no impulses). As a result the coefficient of δ_I in (2.25) is

$$\hat{n} \cdot [\mathbf{a}] = 2\gamma \left(\frac{\bar{v}_s}{R} - \frac{\partial v_n}{\partial s} \right). \tag{2.26}$$

Even though the normal velocity is continuous at the interface for all time, the normal acceleration is discontinuous. The discontinuity in a_n has a physical interpretation. On the right-hand side of (2.26) the first term is the difference of the centripital acceleration and the second term is the difference of the convection. Both terms are proportional to γ and are due to the discontinuity in the tangential velocity.

2.4. The vorticity equation. The velocity is determined by the vorticity density on the interface from (2.4) and (2.5). It follows from (2.8) that when the vorticity has the form (2.5) at the initial time, it has this form at subsequent times. Thus we need only consider the vorticity density on the interface. We use (2.3) and (2.8) to determine the time rate of change of the vorticity density. Let the interface be parameterized by the variable α. Suppose we propagate the interface with the average velocity

$$\left. \frac{\partial \mathbf{r}}{\partial t} \right|_\alpha = \bar{\mathbf{v}}, \tag{2.27}$$

where $\mathbf{r}(\alpha)$ is a point on the interface. The interface motion is determined by the normal velocity. The average velocity is equivalent to a particular choice for the tangential velocity or a parameterization of the interface. From (2.3) and (2.10) we have

$$\left. \frac{\partial \gamma}{\partial t} \right|_\alpha = -\hat{s} \cdot \left[\left. \frac{\partial \mathbf{v}}{\partial t} \right|_\alpha \right] = -\hat{s} \cdot \left(\frac{D}{Dt}[\mathbf{v}] + (\bar{\mathbf{v}} - \mathbf{v}^+) \cdot \nabla \mathbf{v}^+ - (\bar{\mathbf{v}} - \mathbf{v}^-) \cdot \nabla \mathbf{v}^- \right).$$

Substituting (2.8) and using

$$\bar{\mathbf{v}} - \mathbf{v}^+ = \tfrac{1}{2}\gamma \hat{s}, \qquad \bar{\mathbf{v}} - \mathbf{v}^- = -\tfrac{1}{2}\gamma \hat{s},$$

we obtain

$$\left. \frac{\partial \gamma}{\partial t} \right|_\alpha = -\hat{s} \cdot \left[\frac{1}{\rho} \nabla P \right] - \gamma \hat{s} \cdot \frac{\partial \bar{\mathbf{v}}}{\partial s}. \tag{2.28}$$

We can relate the second term on the right-hand side of (2.28) to $\partial/\partial t$ $(\partial s/\partial \alpha)$ as follows:

$$\frac{\partial s}{\partial \alpha} = \left(\left(\frac{\partial x}{\partial \alpha} \right)^2 + \left(\frac{\partial y}{\partial \alpha} \right)^2 \right)^{1/2}. \tag{2.29}$$

Then using $\hat{s} = (\partial x/\partial \alpha, \partial y/\partial \alpha)/(\partial s/\partial \alpha)$,

$$\frac{\partial}{\partial t} \frac{\partial s}{\partial \alpha} = \frac{\dfrac{\partial x}{\partial \alpha} \dfrac{\partial \bar{v}_x}{\partial \alpha} + \dfrac{\partial y}{\partial \alpha} \dfrac{\partial \bar{v}_y}{\partial \alpha}}{\dfrac{\partial s}{\partial \alpha}} = \hat{s} \cdot \frac{\partial \bar{\mathbf{v}}}{\partial s} \Big/ \frac{\partial s}{\partial \alpha}. \tag{2.30}$$

Substituting (2.30) into (2.28), we obtain

$$\frac{\partial}{\partial t}\left(\gamma\frac{\partial s}{\partial \alpha}\right)\bigg|_{\alpha} = -\frac{\partial s}{\partial \alpha}\left[\frac{1}{\rho}\frac{\partial P}{\partial s}\right]. \tag{2.31}$$

The right-hand side represents the generation of vorticity at the interface. It arises because the gradients of the pressure and density are not parallel. The left-hand side is a convection term. When the interface is propagated with the average velocity, the total vorticity over any segment of the interface $\int \gamma\, ds$ would be conserved in the absence of vorticity generation.

For other parameterizations of the interface (2.31) would contain an additional convection term. In a numerical computation other parameterizations may also be obtained by redistributing the points on the interface every few cycles.

In the limit $\rho^+ \to 0$ (Atwood ratio 1), the interface becomes a free surface on which the pressure is constant. In addition, the limit of $(\partial P/\partial s)/\rho^+$ is finite and well defined. This case has been computed, for example, in [2], [3]. However, the pressure equation (2.9) is not suitable for computation in this limit. We have performed calculations for an air, water density ratio of approximately 1:600. This closely approximates the Atwood ratio 1 case over a large range of time.

3. Computational methods.

3.1. Grid construction. An accurate solution of the pressure and stream function equations requires the use of a grid which is aligned with the interface. In the case of finite elements, this means that the element boundaries must be aligned to coincide with the interface. Since it is difficult to fit curvilinear boundaries with rectangular elements, we employ triangles along the interface, and because of their greater accuracy, rectangles away from the interface. The construction of the mesh aligned grid is an essential feature of our method [17]. The construction begins with the choice of a rectangular grid. This rectangular grid may have irregular spacings, so that a type of one-dimensional or tensor product mesh refinement can be introduced at this point. The interface is defined as a set of points joined by linear or polynomial line segments. The points are called vertices and the segments are called bonds. Next a modified interface is formed by the introduction of new vertices at each intersection point of a bond and a segment from the regular rectangular grid. Each new vertex divides a bond in two, replacing it by two adjacent bonds. Going through this augmented interface in the order defined by the interface itself, one of two actions is taken at each vertex. Either a regular rectangular grid node is displaced, so as to have the same position as the vertex, or the vertex is eliminated from the interface by joining two adjacent bonds to form a single bond. In making these choices, care must be taken not to move the same rectangular grid node twice. Also it is necessary to check against bad aspect ratios which can result from this construction. If a triangle with a bad aspect ratio is detected, the grid is discarded and the construction begins afresh with the number of grid lines increased. The final grid constructed in this fashion is a grid of triangles and regular rectangles (rectangles whose nodes were not moved by the construction, due to their distance of at least a mesh spacing from the interface). The triangles result from the bisection of irregular quadralaterals (rectangles whose nodes were moved by the grid construction). If the interface passes through the quadralateral, it must by construction coincide with an edge or diagonal. In the latter case, the interface diagonal is the diagonal chosen to bisect the quadralateral into two triangles. Otherwise an interior diagonal is chosen; for convex quadralaterals with two interior diagonals, the shorter is chosen. This construction gives a grid with a rectangular index structure, so

that fast iterative methods based on the fast Fourier transform can be used in the solution algorithm. Figure 3.1 shows a typical grid constructed using these algorithms.

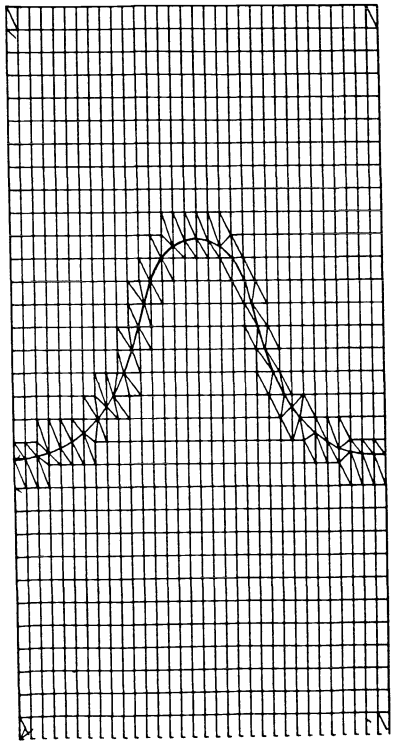


FIG. 3.1. *Typical grid constructed using the algrorithms described in* § 3.1.

The rectangular grid used to begin the mesh alignment construction does not need to be uniform. Using a nonuniform rectangular grid allows a type of one-dimensional or tensor product mesh refinement. This can be used to increase the number of mesh lines near the tip of the spike. In fact solution diagnostics show that the bubble is easier to resolve than the spike, especially for the large density ratio problems. Presumably the reason for this fact is that as the solution progresses, the bubble gets wider and the spike gets narrower. A nonuniform mesh is a method to compensate for this and to allow extra resolution within the spike.

3.2. Boundary data. The equations (2.6) and (2.8) are solved in a rectangle $0 < x < X$, $0 < y < Y$, with gravity pointing in the y direction. We impose Neumann (no flow) boundary conditions on the side boundaries $x = 0$ and $x = X$. Specifically, the pressure equation (2.8) has Neumann boundary conditions while the stream function equation has constant Dirichlet boundary data on these sides since $v_x = \partial_y \Psi$. On the top and bottom boundaries, $y = Y$ and $y = 0$, we impose Dirichlet boundary conditions

$$P_{y=0,Y} = \text{given}$$

on the pressure and the specified flow rate

$$(v_y)_{y=0,Y} = \text{given}$$

which translates into nonzero Dirichlet data for the stream function Ψ. The given

values for v_y at the top and bottom boundary are updated at each time step using (2.8). The pressure values at $y = 0$, Y must be given as part of the formulation of the problem.

In order to compare our finite strip calculation to the infinite strip calculations of [2], [3], one could think of taking a long finite strip and fixing a constant value of the pressure, say

$$P_{y=Y} = 0, \qquad P_{y=0} = P_0$$

and then a suitable value of the constant P_0 would allow comparison of the two solutions. It was found that this strategy was not practical. Even for moderately long finite strips, the influence of the boundaries could be observed in the solution. The value of P_0 varies in time because the center of mass of the fluid is moving. Hence for comparison and validation purposes only, we choose $P_{y=0}$ and $P_{y=Y}$ to coincide with the x, t dependent computed values obtained by Menikoff and Zemach. In the numerical solution of the discretized equations the nonzero boundary conditions are replaced by an equivalent problem with zero boundary conditions and an extra source term.

3.3. Finite element spaces. The elements currently implemented and tested for use in the solution of the pressure and velocity equations are linear, quadratic or cubic for triangles and bilinear, biquadratic or bicubic for rectangles. These elements are Lagrangian, or C^0. In order to resolve a curved interface, low order elements (and a consequently finer grid) are desirable. In order to compute reliable second derivatives of the stream function, as occur in the source term for the pressure equation, high order elements are desirable. Furthermore a thin boundary layer in the air around the falling spike, in which air velocities change rapidly, requires high resolution in the first derivatives of the stream function. The pressure equation is used to determine accelerations, which also change rapidly in a boundary layer around a falling spike. The cubic elements, because of their bandwidth, lead to the use of coarse grids, and so the compromise adopted in this paper has been to use quadratic elements. While we have implemented isoparametric quadratic and cubic elements with curved boundaries, these have not been used in the current validations. Because of the choice of matching Lagrangian elements on rectangles and triangles (quadratic-biquadratic or cubic-bicubic) the resulting discretization matrix on a grid of the type described in the previous section allows a natural rectangular indexing, leading to the possibility of efficient solution schemes.

3.4. Solution of the algebraic equations. On a scalar machine, the major fraction of the computational time is devoted to the solution of the algebraic equations which result from the discretization of the pressure and the stream function equations. On a vector machine or array processor this may no longer be the case. We have run our codes on a VAX 11/780, using an FPS 164 Array Processor or a CRAY 1-S computer to solve the algebraic equations. In these cases the algebraic computation steps become negligible (in real time) resulting in a substantial speedup over running the code on a VAX alone. We have solved these algebraic equations by direct methods (using the Yale sparse matrix software) and iterative methods (using the accelerated conjugate gradient method). The direct methods are faster than iterative methods on smaller grids; we found approximate cross over at about 25×25 grids for quadratic elements. The conjugate gradient scheme for linear elements can be efficiently accelerated by an inverse to Laplace's equation based on the FFT. For quadratic and cubic elements, we have had reasonable success with an acceleration based on the same fast Poisson solver, but for a finer grid—the grid of finite element nodal points. With this acceleration

operator, the number of iterations increases slowly with mesh refinement. A more effective strategy has been to use a limited form of multigrid, to obtain a good initial guess for the iteration. A hierarchy of grid levels is introduced, and the equations are solved on each grid level, using the answer from the next coarser grid as an initial guess for the current grid level [20].

The combination of a VAX and an FPS-164 was found to be especially effective for these problems, the FPS-164 being substantially more cost-effective than the CRAY for the linear equation solution. With the exception of the pre-conditioning acceleration, all of the steps in the conjugate gradient solution are highly vectorizeable. In fact, because the FFT fast solver vectorizes poorly, in some cases the un-preconditioned solution uses less CPU time on the CRAY or FPS, despite the much greater number of iterations involved. On a 64×64 mesh the speedup of the solution to the equations was a factor of about 12 with the FPS and about 45 with the CRAY as compared to single precision on the VAX. For efficiency it is important that the data conversion of numbers between the VAX and CRAY be performed in binary. Due to the word size of the FPS and CRAY compared to the VAX these attached processors have the additional advantage of providing double precision solutions at no extra cost.

3.5. Interpolation. The C^0 elements used for the solution of the pressure and stream function have less regularity than the solution requires. This is particularly noticeable near the interface. Although the first and even second derivatives of quadratic elements are well defined, the solution is only locally C^1. Its derivatives have discontinuities at element boundaries. In addition, with linear bonds for the interface the derivative of the velocity has a logarithmic singularity at the vertices because of the discontinuity in the curvature. To overcome this limitation of the solution, we evaluate velocities and accelerations (i.e. stream function and pressure gradients) at midpoints of finite element triangle boundaries along the interface. Using three successive midpoints, a Newton interpolation algorithm constructs a meaningful solution gradient at each point of the interface. For a triangle near the boundary of the computational region, a missing point for the Newton interpolation stencil can be reconstructed from the knowledge of the boundary conditions. This procedure has been analysed in [21], to which we refer the reader for further information.

3.6. Propagation and remeshing of the front. Velocities and accelerations computed from the stream function and pressure are used to advance the points on the front. These quantities are regularized as described above. The normal components determine the interface considered as a curve. The normal velocity is continuous and thus unique on the interface. The acceleration and tangential component of the velocity have prescribed jumps. The tangential components determine a parameterization or remeshing of the interface. For this reason either the left or right velocity or any combination of the two may be used to move the interface. The acceleration has to take into account the tangential motion. We have used the average of the left and right velocities at the interface. The average velocity has the advantage in that the advection of the vorticity simplifies. The appropriate acceleration for the average velocity is

$$\bar{\mathbf{a}} = \frac{1}{2}\left(\frac{\partial \mathbf{v}^+}{\partial t}\bigg|_\alpha + \frac{\partial \mathbf{v}^-}{\partial t}\bigg|_\alpha \right). \tag{3.1}$$

Performing the same sort of manipulations as used in § 2.4 and from (2.8), we obtain

$$\bar{\mathbf{a}} = -\overline{\left(\frac{\nabla P}{\rho}\right)} - \nabla\left(\frac{\Phi}{\rho}\right) - \frac{1}{4}\gamma\left(\frac{\partial \gamma}{\partial s}\hat{s} - \frac{\gamma}{R}\hat{n}\right). \tag{3.2}$$

We remark that (2.31) and (3.2) can be combined to give Eq. (1) of [3]. This is an equation for the change in vorticity in terms of the Atwood ratio and the average acceleration.

After a sufficient number of time steps, the interface points may become distributed nonuniformly. It is important to remesh the interface. In fact the interface is a Lagrangian degree of freedom, and it is well-known that occasional remeshing is necessary to preserve the quality of the Lagrangian mesh. The problem here is elementary, as the interface is one-dimensional. The remesh is performed automatically as the computation proceeds.

3.7. Improved finite element spaces. We are currently pursuing several strategies which we believe have a potential to improve substantially upon our current results. The strategies are the use of curved bonds for the interface, local mesh refinement and Hermite or C^1 finite elements in the region near the interface, where high resolution in the first and second derivatives of the solution is essential.

The physics of the two-fluid Euler equation leads to discontinuities at the interface in the derivative of the stream function and in the pressure and its derivative. With Lagrangian elements the normal derivatives at the interface depend on the solution at a distance one mesh block removed from the interface. We propose that elements with specified jump conditions permit the use of normal derivatives evaluated directly on the interface. Since we have not found a description of such jump discontinuous C^1 Hermite elements in the standard literature, we will indicate here how they may be constructed.

We consider jump conditions of the form

$$[\phi] = \alpha_0 \bar{\phi} + \beta_0, \qquad [\nabla_n \phi] = \alpha_1 \overline{\nabla_n \phi} + \beta_1.$$

If $\phi = \Psi$ is the stream function, then $-\beta_1 = \gamma$ is the vorticity on the interface. In this case $\alpha_0 = \beta_0 = \alpha_1 = 0$. If $\phi = P$ is the pressure, then $\beta_0 = -(\sigma/R)$, where σ is the surface tension. Also $\alpha_0 = 0$. To determine α_1 and β_1 we write

$$-\lambda = \left[\frac{1}{\rho} \nabla_n P\right]$$

where λ is the line source given by (2.26). Algebraic manipulation leads to the equations

$$[\nabla_n P] = -\rho^- \lambda - \rho^- \left[\frac{1}{\rho}\right] \nabla_n P^+ = -\rho^+ \lambda - \rho^+ \left[\frac{1}{\rho}\right] \nabla_n P^-$$

and so by a convex combination of these equations,

$$[\nabla_n P] = -\frac{\rho^+ \rho^-}{2\bar{\rho}} \lambda - \frac{\rho^+ \rho^-}{\bar{\rho}} \left[\frac{1}{\rho}\right] \overline{\nabla_n P}.$$

Thus

$$-\alpha_1 = \frac{\rho^+ \rho^-}{\bar{\rho}} \left[\frac{1}{\rho}\right]$$

which is twice the Atwood ratio and

$$-\beta_1 = \frac{\rho^+ \rho^-}{2\bar{\rho}} \lambda.$$

We imagine a grid composed of rectangles and triangles with the interface (and the jump conditions) confined to the triangle boundaries. Let ϕ_i be a basis element at

a node where a jump condition is imposed. By a node we mean a specification of a point on the triangle boundary and the appropriate derivative (ϕ, ϕ_x, etc.) to be evaluated there. Since a jump in the first derivative implies an unknown jump in higher derivatives, we require the specified derivatives at the nodes to be first order. Because of nodes at triangle vertices we require the normal to the interface to be continuous. Otherwise the jump in the normal derivative will result in the function being discontinuous. A continuous normal preserving the C^1 continuity can be obtained by using an isoparametric transformation to specify curved bonds as follows. We consider one grid with linear bonds in (ζ, η) space and transform to (x, y) space. The same Hermite finite element basis is used to express the co-ordinate transformation. We choose the values of (x, y) and (ζ, η) to be equal at the vertices and obtain the curved bonds by the choice of the derivatives. These may be obtained by fitting the x and y co-ordinates of vertices along the interface with a spline parameterized by arc length in the (ζ, η) plane. The choice of cubic splines results in the curvature being continuous in both magnitude and direction. By choosing the Jacobian of the transformation to be the identity for nodes away from the interface, the triangles join smoothly to the rectangles without the need for a further co-ordinate transformation. The normal derivatives of the co-ordinate transformation on the interface are arbitrary. One can use this freedom to minimize the variation in the Jacobian of the co-ordinate transformation. In this procedure noninterface bonds with a vertex on the interface cannot be chosen independently and in general will also be curved. The use of Hermite elements for the co-ordinate transformation enables its Jacobian to be continuous. Thus the transformation does not alter the C^1 continuity of the basis elements. However, the Jacobian has to be taken into account for the jump conditions in the (ζ, η) space. We note two exceptional cases. If the interface intersects itself, then there are not enough degrees of freedom in the derivatives of the transformation to independently specify the curvature on all bonds with the common vertex. Also, if two bonds of a triangle lie on the interface, then a continuous normal at the common vertex implies a singularity in the Jacobian (i.e. a nonsingular co-ordinate transformation cannot take a zero angle into a nonzero angle). This would apply to curved bonds for Lagrangian finite elements as well.

One set of elements with the appropriate properties consists of nodes at vertices with ϕ, ϕ_x, ϕ_y specified and nodes at the midpoints of bonds with ϕ_n specified. These are rather complicated and may lead to numerical difficulties. It may be sufficient to use nonconforming elements with C^1 continuity only at the nodes. An example of such a choice would be the standard Hermite serendipity bicubics on rectangles and the Hermite cubics on triangles. These have the advantage that the elements have compatible boundaries and transition elements are not needed. These elements are described in more detail in [18], [19].

The homogeneous jumps (proportional to the α's) are handled differently from the inhomogeneous jumps (proportional to the β's). To make this explicit, we write

$$\phi = \phi_i^H + \phi_i^P$$

where ϕ_i^H vanishes at all nodes not equal to i, satisfies homogeneous jumps at node i (and thus at all nodes) and $\bar{\phi}_i^H$ or $\overline{\nabla_n \phi}_i^H$ as appropriate has unit value at node i.

The particular element ϕ_i^P vanishes at all nodes not equal to i, and $\bar{\phi}_i^P$ or $\overline{\nabla_n \phi}_i^P$ as appropriate vanishes at node i. Thus ϕ_i^P satisfies homogeneous jump conditions at the nodes not equal to i. At node i, ϕ_i^P satisfies the inhomogeneous jump condition (and thus also the full jump condition at that node).

In solving the finite element equations, $\phi^P = \sum_i \phi_i^P$ is thought of as a source term, so that a variational principle is applied to $\phi^H = \sum_i \phi_i^H$ only. Finally we note that ϕ_i^H and ϕ_i^P, restricted to a particular triangle, are expressed as multiples of the conventional Hermite basis functions. These multiples depend on the α's and β's at that node, and can be regarded as connection coefficients. In this way discontinuous elements with specified jumps are represented in terms of conventional Hermite elements together with a set of connection coefficients.

4. Preliminary validation studies. The primary method which we use for validation is quantitative comparison to the conformal mapping solution of Menikoff and Zemach [2] for the case of an infinite density ratio. We are not set up to handle this limiting case directly in our front tracking scheme and so we have chosen the density parameters to be those of water and air, which are approximately in the ratio 600:1. For the time interval during which we follow the solution, the resulting difference between the two problems is not significant. The Menikoff-Zemach solution describes flow in an infinite strip,

$$a < x < b, \qquad -\infty < y < \infty.$$

This is a natural choice for a method based on integration of Green's functions. The front tracking code uses finite elements to invert elliptic operators. In this case a bounded domain with boundary conditions is the natural problem. To allow comparison to the Menikoff-Zemach solution, we determine the (time and space dependent) pressure of the Menikoff-Zemach solution at our upper and lower y boundaries and impose these values as boundary conditions on the front tracking solution. The calculations in [2] provide the solution only in the heavy fluid. However, the data on the interface is sufficient to determine the solution in the light fluid in the limit of 0 density. This extension is described in the appendix.

These two solutions are compared with respect to interface position as well as to a series of detailed diagnostic quantities such as velocity, acceleration and vorticity. In general we get excellent agreement for the interface position on fine grids through an amplitude to wavelength ratio greater than one, with divergence starting somewhat earlier on coarse grids and for the more sensitive diagnostics, especially those containing more derivatives of the solution. It is hoped that planned future improvements in the calculation will improve both the coarse grid and sensitive diagnostics.

We choose for a test problem the case with gravitational acceleration $g = 1$, fluid densities $\rho_{\text{water}} = 1$, $\rho_{\text{air}} = 0.0017$. At $t = 0$ we choose an initial interface $y = -\frac{1}{2}\cos(x)$ with zero velocity. In Fig. 4.1 we compare the time evolution of the interface for a 32×32 and 64×64 grid. We see that the solutions have converged under mesh refinement up to $t = 2.4$. The coarse grid solution shows discernable errors at $t = 3$ due to a loss of resolution in the spike region. The symptoms of this loss of accuracy are a decrease in the acceleration of the tip of the spike and a corresponding increase in its width. If the calculation were allowed to continue, the spike would develop a mushroom cap shape similar to what occurs at lower density ratios due to Kelvin-Helmholtz rollup.

In Fig. 4.2 we display the velocity and acceleration fields at $t = 2.4$ and 3. This shows the development of vorticity along the interface, especially on the sides of the spike. The velocity and acceleration at the top and bottom boundaries are small. Thus the mesh boundary has a small effect on the calculation.

In Fig. 4.3 we compare the position of the interface and the vorticity density along the interface to the corresponding quantities in the Menikoff-Zemach solution. The agreement between the interface positions is good except at the tip of the spike where

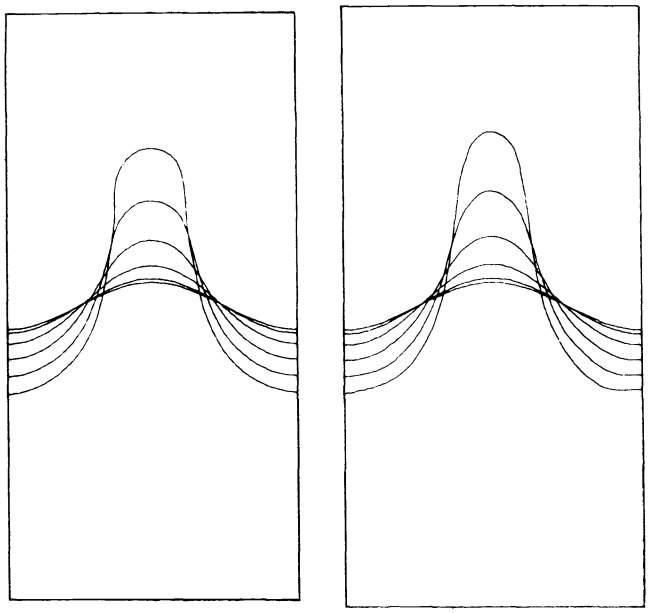


FIG. 4.1. *Time evolution of interface from* $t=0$ *to* 3 *in steps of* 0.6. *Heavy fluid is on the bottom.* a) 32×32 *run*, b) 64×64 *run.*

our solution is not accelerating quite as fast as it should. The vorticity density becomes sharply peaked along the sides of the spike. As pointed out in [2], this leads to numerical difficulties due to loss of resolution in the spike region.

Figure 4.4 is a comparison of the y velocities of the heavy fluid along the interface in the present solution and in the Menikoff-Zemach solution. The error in the vorticity

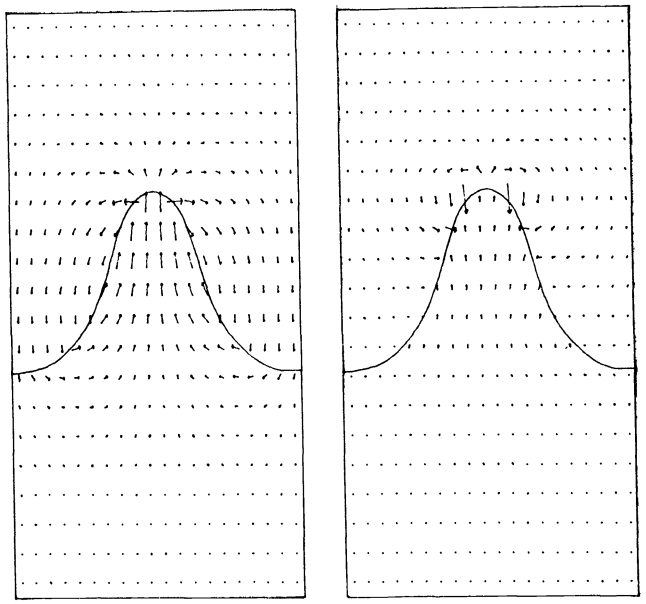


FIG. 4.2a. *The velocity and acceleration plots with interface superimposed, at the time* $t=2.4$ *from the* 64×64 *run of Fig.* 4.1b.

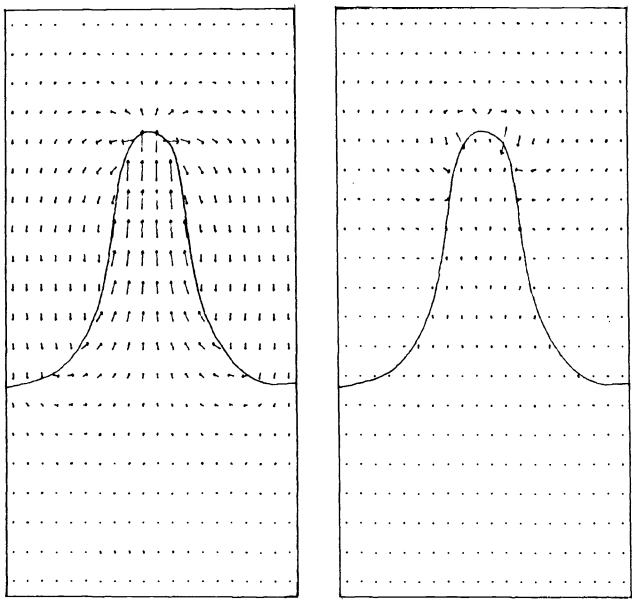


FIG. 4.2b. *The velocity and acceleration plots with interface superimposed, at the time* $t = 3.0$ *from the* 64×64 *run of Fig.* 4.1b.

density around the tip of the spike seen in Fig. 4.3 causes a lower velocity at the tip of the spike. The oscillations in the raw velocity presented in Fig. 4.4a occur because the finite elements for the stream function are only C^0. The velocity is discontinuous at the vertices on the interface. This difficulty is alleviated by smoothing the velocity as described in § 3.5. See Fig. 4.4b.

In Fig. 4.5 we compare the pressure along the interface. The slight asymmetry about the tip of the spike is due to a loss of resolution in the spike region and perhaps

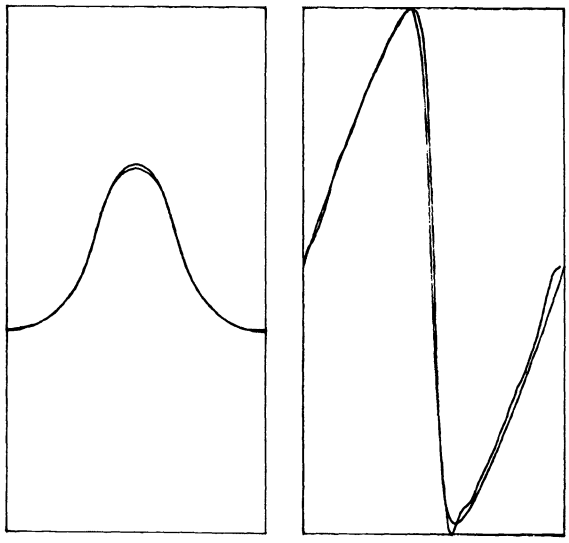


FIG. 4.3a. *Comparison between the* 64×64 *run of Fig.* 4.1b *and the Menikoff–Zemach solution. The interface position is shown on the left and the vorticity density on the right, at the time* $t = 2.4$.

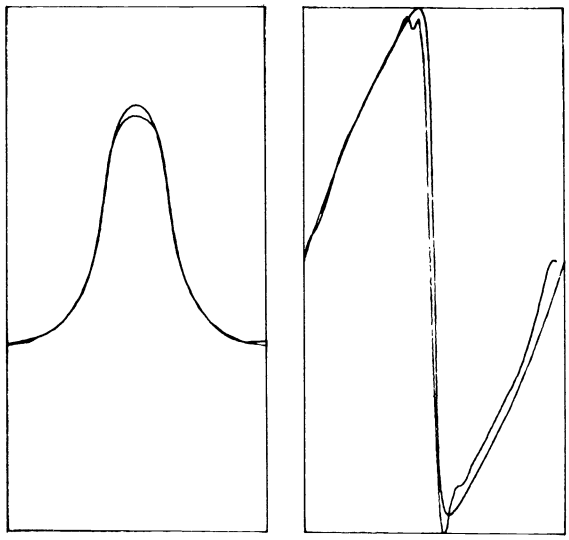


FIG. 4.3b. *Comparison between the* 64×64 *run of Fig.* 4.1b *and the Menikoff–Zemach solution. The interface position is shown on the left and the vorticity density on the right, at the time* $t = 3.0$.

also the grid construction algorithm, which does not try to enforce the symmetry. The fact that the pressure on the interface is very small compared to the driving pressure on the boundary and that there is good agreement in the interface pressures both indicate that the difference in the problems (large vs. infinite density ratio) is not significant at this time. At large time the pressure at the tip of the spike is on the order of $\rho_{\text{air}} v^2$ and would become important.

This example shows good agreement up to an amplitude to wavelength ratio of about 1. Furthermore, the solution is improved by mesh refinement.

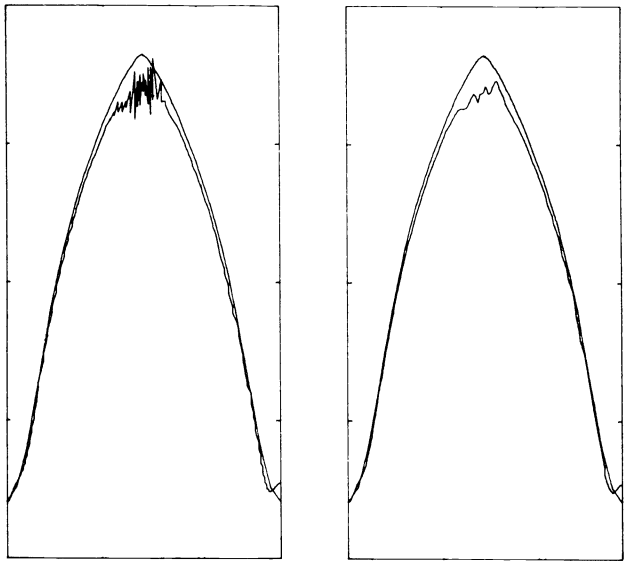


FIG. 4.4. *Comparison with Menikoff–Zemach solution of y velocity in heavy fluid on interface vs. arc length at* $t = 3.0$. a) *raw data,* b) *smoothed data.*

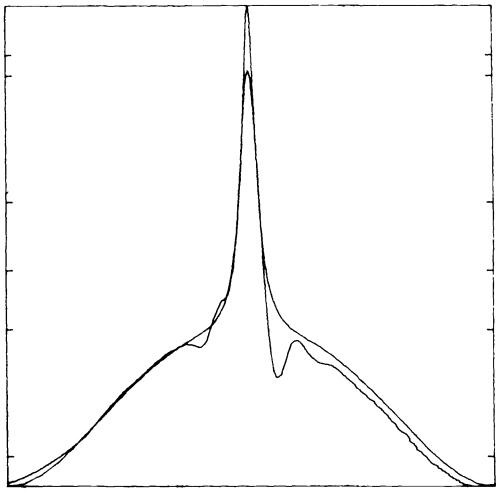


FIG. 4.5. *Comparison with Menikoff-Zemach solution of pressure on interface vs. arc length at* $t = 3.0$.

5. Finite density ratios. Moderate to low density ratios display qualitatively new phenomena, as compared to the high or infinite density ratios discussed in the previous section. In this case the spike develops a mushroom cap shape due to Kelvin-Helmholtz rollup along the side of the spike. Along the base of the mushroom cap there is a vortex rollup. This phenomenon occurs in principle for any finite density ratio, and the distinguishing feature of the low to moderate density ratio range is that the phenomenon occurs at an early stage in the development of the spike. Although the conformal mapping methods of [2] apply in principle to this case, they have not been so implemented. The boundary integral methods of [3] have been performed in this case, and are in qualitative agreement with our results. However because their results apply to the case of an infinite strip, detailed comparison with our results in a bounded domain with boundary conditions is not appropriate. Many of the general purpose codes cited in § 1 have been applied to the moderate to low density regime. We again agree with their calculations qualitatively, but should not and do not agree quantitatively due to the presence of numerical diffusion in such codes and the absence of it in ours.

For front tracking methods the low to moderate density ratio regime is an easier problem than the water-air case considered in the previous section. In fact the extra topological complications associated with vortex rollup (interface tangling, etc.) are a minor added difficulty while the decrease in singularity for the elliptic problem to determine the pressure is a major simplification. Thus we rely partly on the validation of the previous section for the present regime. Moreover we find that convergence is achieved with fairly coarse grids (e.g. 15×15 for a single finger). Meaningful coarse grid calculations are important in a study of multiple finger interactions.

Our main quantitative test of the front tracking code in the present regime is convergence under mesh refinement. We examine both the front position and various detailed diagnostics such as velocity and acceleration on the interface. The test runs reported here used a density ratio of 4:1 (Atwood ratio of .6). The initial conditions represent a fluid at rest with a sinusoidally disturbed interface. The gravitational acceleration is $g = 1$ and there is a constant pressure difference between the top and bottom of the mesh to balance the initial gravitational force. In Fig. 5.1 we show the development of a single finger, showing the interface position at a late time for a 46×46 grid.

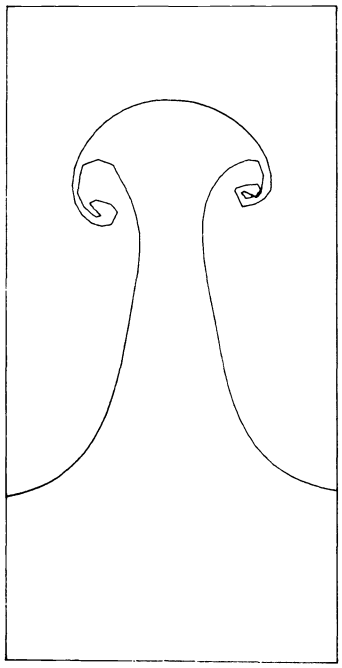


FIG. 5.1. *Interface position at a late time for density ratio* 4:1.

6. Heterogeneity. In a highly developed regime for the Rayleigh-Taylor instability one is interested in both the dynamics of a single finger and the interaction between fingers. Heterogeneity affects both. There are of course a large number of possible heterogeneity parameters. Examples include multiple frequencies in the initial interface position, variable material properties such as density, variable vorticity as a distributed external source or initial condition, and variable driving pressure in the externally imposed boundary conditions.

We can divide these possibilities into two groups. The subharmonic heterogeneity acts on length scales longer than that set by some dominant finger size. This case affects the interactions between bubbles. The superharmonic heterogeneity acts on length scales shorter than that set by the finger size. This case affects the finger shape. The present section contains a preliminary examination of both of these possibilities. The subharmonic example we present is the case of multiple frequencies in the initial interface position, and the superharmonic example is vorticity as an external driving source.

As a multiple frequency interface, we consider a combination of two sine waves with a frequency ratio of 8:1, leading to the formula

$$y = \text{constant} + \alpha \sin(k_1 x) + \beta \sin(k_2 x).$$

In Fig. 6.1 we show the time development of the interface position for the particular choice

$$k_1 = 1, \quad k_2 = \tfrac{1}{8}, \quad \alpha = .5, \quad \beta = 2.$$

We see that the long wavelength modulation of the initial interface excites a mechanism of bubble competition and probably eventually of bubble amalgamation. Since the bubble growth is a function of the bubble size, we see that the long wavelength

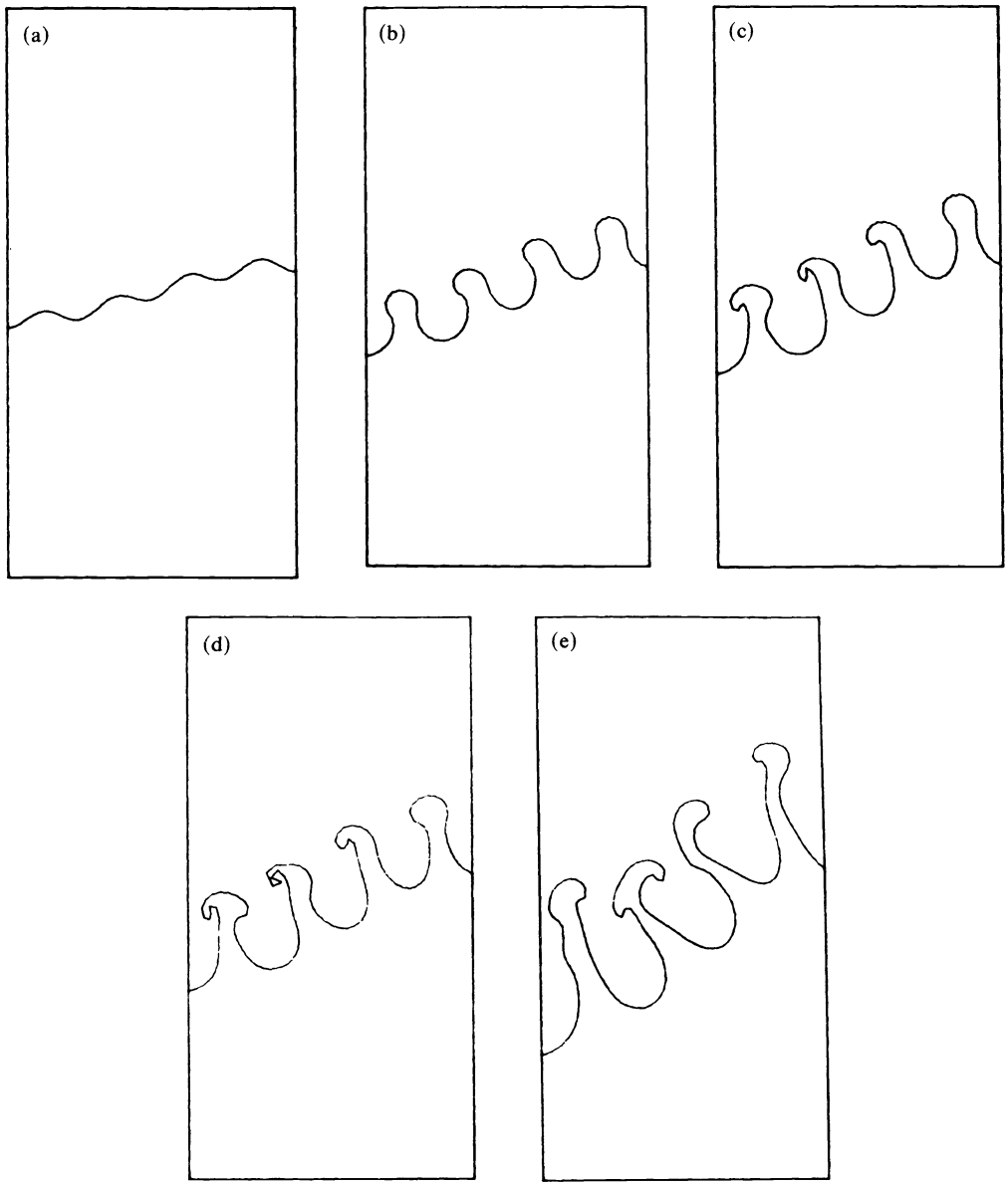


FIG. 6.1a-e. *Initial and four subsequent time steps in the evolution of a multifrequency interface. It appears that the high frequency modes grow rapidly initially and then saturate, while the low frequency mode is relatively unaffected by the high frequency fingers growing off it. In order to progress from Fig.* 6.1d to 6.1e, *some detail in the vortex rollup was suppressed, as can be inferred by comparison to Fig.* 5.1.

modulation produces an increased overall bubble growth rate. From another point of view, the short and long wavelengths grow to a large extend independently of one another. Once the short wavelength fingers are in a deeply nonlinear region, it can be seen that trailing vortices prefer to form on the down wind side, as determined by the long wavelength perturbation.

In two space dimensions vorticity can be regarded as a scalar. We construct a random vorticity function γ_{random} depending on three deterministic parameters and a

further random variable. The three deterministic parameters are an amplitude and an x and a y length scale. The x and y length scales determine a discrete mesh. At each point of this mesh we choose independent Gaussian random variables with mean 0 and variance given by the above amplitude. This construction defines an ensemble of functions on the discrete lattice. Finally an interpolation gives an ensemble of everywhere defined continuous random variables. The Gaussians at each mesh point are evaluated using a random number generator thereby producing a point in the ensemble, which is the desired vorticity function. A shift in the starting point of the random number generator or other change in the random number generator produces a new point in the ensemble, i.e. a new vorticity having the same values for its three deterministic parameters [22].

A sequence of single finger runs was performed in which the heterogeneity amplitude is continuously increased. The initial run in this series is the homogeneous (zero heterogeneity amplitude) run of Fig. 5.1. The density ratio was fixed at 4:1 and the heterogeneity length scales were $\frac{1}{5}$ of the finger wavelength, so the heterogeneity is superharmonic. The results were presented in [16], and will not be repeated here. The main conclusion is that heterogeneity can affect finger shape and if strong enough can even cause splitting of fingers.

7. Conclusions. We have developed the front tracking method for two-dimensional incompressible inviscid fluid flow. This code was applied to the problem of Rayleigh-Taylor instability. Preliminary validation studies comparing this code to existing special purpose codes have shown good results.

Several strategies for further improving the code have been identified. These include local mesh refinement and the use of Hermite elements with prescribed jump discontinuities in conjunction with isoparametric elements.

An initial study of the effect of statistically distributed heterogeneities was performed. This work suggests that such heterogeneities can substantially modify the flow. A systematic study of their effects is planned.

A capability of the code not utilized in this work is the existence of general data structures which permit one to keep track of interfaces of arbitrary topology. This will be brought into play in future studies concerning bubble amalgamation, disintegration of spikes into droplets and the breakup of shells.

Another capability not utilized is a framework which can incorporate noninterface physics. A use of this capability would be the modeling of diffusion of vorticity away from the front. We also contemplate treating fine scale structure such as tightly wound vortex spirals as interior or untracked degrees of freedom, while continuing to track the main interface.

Appendix: Extension of the conformal map solution. The use of conformal maps to numerically compute the solution of the Rayleigh-Taylor problem is described in [2]. There an explicit calculation is performed for a single semi-infinite fluid with a free surface. Here we briefly sketch how the data on the interface can be used to determine the flow in the complementary semi-infinite region in the limit in which the second fluid has 0 density.

We denote the first and second fluid with the superscripts $-$ and $+$ respectively. We suppose the position of the interface and the velocity of the first fluid are known from the calculation in [2]. We express the velocity as the gradiant of a potential ϕ which satisfies the Laplace equation separately in each fluid region. The normal velocity is continuous across the interface. Thus v_n^- provides the boundary condition for the potential problem determining ϕ^+. Using the methods in [2], we can compute the

conformal map for the region of fluid 2. (This map is different from the conformal map for the first fluid because of the asymmetry between the bubble and spike.) This determines the Green's function which allows us to compute by quadrature the tangential velocity from the normal velocity. The jump in the tangential velocity enables us to determine the vorticity density γ along the interface.

In order to calculate the pressure in the second fluid, we first need to calculate the normal acceleration. The potential is determined by Bernoulli's equation

$$\partial_t \phi = \frac{1}{2} v^2 + gy + \frac{P}{\rho}. \tag{A.1}$$

For the first fluid $\partial_t \phi$ can be calculated on the interface because for a free surface $P = 0$. Its tangential derivative may be obtained numerically. From the Green's function we can determine by quadrature

$$\partial_n(\partial_t \phi) = \hat{n} \cdot \partial_t \mathbf{v} \tag{A.2}$$

from $\partial_s(\partial_t \phi)$. The normal acceleration is

$$a_n = \hat{n} \cdot (\partial_t + \mathbf{v} \cdot \nabla)\mathbf{v}. \tag{A.3}$$

By using (2.17) and (2.18) this can be expressed in terms of quantities already known on the interface

$$a_n = \partial_n(\partial_t \phi) + v_s\, \partial_s v_n - v_n\, \partial_s v_s - \frac{v_s^2 + v_n^2}{R}. \tag{A.4}$$

From the jump in the acceleration equation (2.26) and the jumps in the velocity and its derivatives (2.12)-(2.23) it follows that

$$\partial_n[\partial_t \phi] = -(\gamma\, \partial_s v_n + v_n\, \partial_s \gamma). \tag{A.5}$$

Thus we can determine $\partial_n(\partial_t \phi)$ in the second fluid. The normal acceleration in the second fluid is determined from (A.4). We may also use the Green's function to determine by quadrature $\partial_s(\partial_t \phi)$ from $\partial_n(\partial_t \phi)$. This together with (A.1) for the second fluid determines $\partial_s(P/\rho)$. We can integrate this to determine P/ρ along the interface of the second fluid. It is important that P/ρ is finite in the limit as $\rho^+ \to 0$.

For a nonzero ρ^+ the pressure on the interface may be substituted back into (A.1) for the first fluid and this procedure can be iterated until the pressure converges. This would provide an alternate method for computing the two fluid problem to that described in [2].

Acknowledgments. It is a pleasure for the authors to thank the personnel in the computer centers at the Courant Institute and the Los Alamos National Laboratory for their friendly and effective cooperation.

REFERENCES

[1] R. MENIKOFF AND C. ZEMACH, *Methods for numerical conformal mapping*, J. Comp. Phys., 36 (1980), pp. 366-410.

[2] ———, *Rayleigh-Taylor instability and the use of conformal maps for ideal fluid flow*, J. Comp. Phys. 51 (1983), pp. 28-64.

[3] G. R. BAKER, D. I. MEIRON AND S. A. ORSZAG, *Vortex simulations of the Rayleigh-Taylor instability*, Phys. Fluids, 23 (1980), pp. 1485-1490.

[4] C. P. VERDON, R. L. MCCRORY, R. L. MORSE, G. R. BAKER, D. I. MEIRON AND S. A. ORSZAG, *Nonlinear effects of multifrequency hydrodynamic instabilities on ablatively accelerated thin shells*, Phys. Fluids, 25 (1982), pp. 1653-1674.

[5] G. BIRKHOFF AND D. CARTER, *Rising plane bubbles*, J. Math. Mech., 6 (1957), pp. 769-779.
[6] F. H. HARLOW AND J. E. WELCH, *Numerical study of large-amplitude free-surface motions*, Phys. Fluids, 9 (1966), pp. 842-851.
[7] B. J. DALY, *Numerical study of two fluid Rayleigh-Taylor instability*, Phys. Fluids, 10 (1967), pp. 297-307.
[8] ———, *Numerical study of the effect of surface tension on interface instability*, Phys. Fluids, 12 (1969), pp. 1340-1354.
[9] W. P. CROWLEY, *An empirical theory for large amplitude Rayleigh-Taylor instability*, Lawrence Livermore Laboratory, Report UCRL-72650, 1970.
[10] J. R. FREEMAN, M. J. CLAUSER AND S. L. THOMPSON, *Rayleigh-Taylor instabiities in inertial-confinement fusion targets*, Nucl. Fusion, 17 (1977), pp. 223-230.
[11] K. A. MEYER AND P. J. BLEWETT, *Some preliminary numerical studies of Taylor instability which include effects of material strength*, Los Alamos National Laboratory, Report LA-4754-MS, 1971.
[12] ———, *Numerical investigation of the stabiity of a shock-accelerated interface between two fluids*, Phys. Fluids, 15 (1972), pp. 753-759.
[13] Some examples are: J. D. Lindl and W. C. Mead, *Two-dimensional simulation of fluid instability in laser-fusion pellets*, Phys. Rev. Lett., 34 (1975), pp. 1273-1276; R. L. McCrory, L. Montierth, R. L. Morse and C. P. Verdon, *Nonlinear evolution of ablation-driven Rayleigh-Taylor instability*, Phys. Rev. Lett., 46 (1981), pp. 336-339; M. H. Emery, J. H. Gardner and J. P. Boris, *Rayleigh-Taylor and Kelvin-Helmholtz instabilities in targets accelerated by laser ablation*, Phys. Rev. Lett., 48 (1982), pp. 677-680; R. G. Evans, A. J. Bennett and G. J. Pert, *Rayleigh-Taylor instabilities in laser-accelerated targets*, Phys. Rev. Lett., 49 (1982), pp. 1639-1642.
[14] J. GLIMM, *The role of better algorithms*, in Frontiers of Supercomputing, Univ. California Press, Berkeley, to appear.
[15] O. MCBRYAN, *Computing discontinuous flows*, presented at the International Conference on Fronts, Patterns and Interfaces, Los Alamos, NM, 1983.
[16] D. H. SHARP, *An overview of Rayleigh-Taylor instability*, Physica, 12D (1984), pp. 3-18.
[17] O. MCBRYAN, *Elliptic and hyperbolic interface refinement*, in Boundary Layers and Interior Layers-Computational and Asymptotic Methods, J. Miller, ed., Boole Press, Dublin, 1980.
[18] K. HUEBNER, *The Finite Element Method for Engineers*, John Wiley, New York, 1975.
[19] L. LAPIDUS AND G. PINDER, *Numerical Solution of Partial Differential Equations in Science and Engineering*, John Wiley, New York, 1982.
[20] O. MCBRYAN, *Multi-grid methods for discontinuous equations*, Proc. International Multi-grid Conference, Copper Mountain, CO, 1983, to appear.
[21] J. GLIMM, B. LINDQUIST, O. MCBRYAN AND L. PADMANABHAN, *A front tracking reservoir simulator, five-spot validation studies and the water coning problem*, in The Mathematics of Reservoir Simulation, R. E. Ewing, ed., Frontiers in Applied Mathematics 1, Society for Industrial and Applied Mathematics, Philadelphia, 1983, pp. 107-135.
[22] J. GLIMM, E. ISAACSON, B. LINDQUIST, O. MCBRYAN AND S. YANIV, *Statistical fluid dynamics: the influence of geometry on surface instabilities*, in The Mathematics of Reservoir Simulation, R. E. Ewing, ed., Society for Industrial and Applied Mathematics, Philadelphia, 1983, pp. 137-160.

SIAM J. SCI. STAT. COMPUT.
Vol. 7, No. 1, January 1986

017

A FRONT TRACKING METHOD FOR ONE-DIMENSIONAL MOVING BOUNDARY PROBLEMS*

GUILLERMO MARSHALL†

Abstract. We introduce a front tracking method for the numerical solution of one-dimensional Stefan problems. It consists in the formulation of the Stefan problem as an ordinary differential initial-value problem for the moving boundary coupled with a parabolic partial differential equation for the distribution of temperatures. We present a variable time step procedure in which the initial-value problem is solved with a predictor-corrector scheme; in the corrector step the function evaluation is done, iteratively, through an implicit time discretization of the parabolic equation. Numerical results for one-dimensional, one-phase Stefan problems with straight and curved moving boundary trajectories are presented. For these cases the front tracking method presented gives greatly improved results.

Key words. initial-boundary value problem, Stefan problem, front tracking

1. Introduction. A moving boundary problem or Stefan problem is a nonlinear initial boundary value problem with a moving boundary whose position is unknown and has to be determined as part of the solution.

Among the various methods in current use for studying the Stefan problem, finite difference and finite element methods are widely used and have proven to be the most general. They can be classified into two main categories: the methods which explicitly track the interface or front thus solving the differential problem in a time varying domain, and those which use a fixed domain with the help of a weak or generalized formulation. By a suitable coordinate transformation it is possible to obtain methods that can be also included in the last category. An enlightening discussion of these methods can be found in the works of Crank [3], Furzeland [5] and Meyer [11] (see also Ockenden and Hodgkins [12]).

Finite difference one-dimensional moving boundary tracking methods, hereafter called front tracking methods, have been classified by Gupta [6] as fixed and variable grid methods. In the fixed grid method the space-time domain is subdivided into a finite number of equidistributed cells, and the trajectory of the front does not necessarily coincide with the cell nodes. In the variable grid method the space-time domain is subdivided into a finite number of rectangular cells with only one side equidistributed, the other side being subdivided in such a way that the trajectory of the front coincides with the cell nodes. Examples of fixed grid methods can be found, for instance, in the works of Crank [2], Meyer [10] and Furzeland [5]. Douglas and Gallie [4] were the first to introduce a variable grid method; they subdivide the space direction into an equally distributed mesh length and determine the time step in such a way that the trajectory of the front advances one space cell per time step. For the solution of the nonlinear system they introduce an iteration procedure by which, at every time step, an arbitrary initial guess for the time step is chosen and the parabolic equation is then solved using a strongly implicit time discretization scheme; for the next iteration the new time step is corrected using a quadrature of the integral form of the moving boundary condition. Stability and uniform convergence of the method in a suitable restricted domain were also proven. However, Gupta and Kumar [6] showed that the

* Received by the editors November 8, 1983, and in revised form November 26, 1984. Part of this work was done while the author was at the Courant Institute of Mathematical Sciences, New York University, New York.

† Consejo Nacional de Investigaciones, Científicas, Buenos Aires, Argentina. Present address: Centro de Cálculo Científico, Comisión Nacional de Energía Atómica, 1429 Buenos Aires, Argentina.

previous method did not converge under more rigorous conditions, and presented a new variable time step method which is closely related to the Douglas and Gallie method. It differs in the way in which the new time step is corrected: Gupta and Kumar use a first order finite difference approximation of the differential form of the moving boundary condition.

In the present work we propose a new front tracking method which shares some of the ideas discussed above. It consists in the formulation of the one-dimensional Stefan problem as an ordinary differential initial-value problem governing the moving boundary coupled with a parabolic partial differential equation describing the diffusion process. This slightly different formulation, that grants equal importance to the moving boundary and to the diffusion process, leads naturally to a consistent difference approximation. We introduce a variable time step procedure in which the initial-value problem is approximated with a second order predictor-corrector scheme; in the corrector step the function evaluation is performed, iteratively, through the use of an implicit time discretization of the parabolic partial differential equation. As will be shown, this front tracking method yields a simple, robust, fast and accurate numerical procedure. We present numerical results for two test cases involving straight line and curved line front trajectories and a comparison with other authors' results.

2. Formulation of the Stefan problem. We reformulate the Stefan problem using the test case presented by Douglas and Gallie [4]. One-dimensional, one-phase Stefan problems consist in finding the space-time curve $S(t)$ along which the front moves satisfying the initial-value problem

$$\dot{S} = \frac{dS}{dt} = -\frac{\partial u}{\partial x}, \qquad x = S(t), \quad t > 0, \tag{2.1}$$

$$S(t) = 0, \qquad t = 0, \tag{2.2}$$

and the value of the function $u(x, t)$ in the region $0 \leqq x \leqq S(t)$ satisfying the following parabolic partial differential equation for the function $u(x, t)$

$$\frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial^2 x}, \qquad 0 \leqq x \leqq S(t), \quad t > 0, \tag{2.3}$$

with initial and boundary conditions

$$u = 0, \qquad 0 \leqq x \leqq S(t), \quad t = 0, \tag{2.4}$$

$$\frac{\partial u}{\partial x} = -1, \qquad x = 0, \quad t > 0, \tag{2.5}$$

$$u = 0, \qquad x = S(t), \quad t > 0. \tag{2.6}$$

An alternative formulation is to replace (2.1) and (2.2) by its integral form

$$S(t) = t - \int_0^{S(t)} u \, dx. \tag{2.7}$$

It can easily be shown (see for instance Cohen and Tadjbakhsh [1]) that the Stefan problem is a nonlinear problem. Performing the coordinate transformation $z = x - S(t)$ system (2.1)-(2.6) becomes

$$\dot{S} = \frac{dS}{dt} = -\frac{\partial u}{\partial z}, \qquad z = 0, \quad t > 0, \tag{2.8}$$

$$s(t) = 0, \tag{2.9}$$

$$\frac{\partial u}{\partial t} = \dot{S}\frac{\partial u}{\partial z} + \frac{\partial^2 u}{\partial z^2}, \qquad -S(t) \leqq z \leqq 0, \quad t > 0, \tag{2.10}$$

$$u = 0, \qquad -S(t) \leqq z \leqq 0, \quad t = 0, \tag{2.11}$$

$$\frac{\partial u}{\partial z} = -1, \qquad z = -S(t), \quad t > 0, \tag{2.12}$$

$$u = 0, \qquad z = 0, \quad t > 0. \tag{2.13}$$

The convective term appearing in (2.10) has a coefficient which depends on the unknown $u(z, t)$.

3. The numerical problem. This section discusses numerical approximations for the Stefan problem (2.1)-(2.6) using front tracking methods. The space-time domain $0 \leqq x \leqq S(t)$ is discretized with a rectangular grid which distribution is discussed below. Space and time increments are indicated by h and k, respectively. As usual the value of the numerical solution for $x = ih$ and $t = nk$ (i and n integers) is given by $u(x, t) = u(ih, nk) = u_i^n$. Assuming that the space-time curve of the front is a parabola, the following strategies for the discretization of the domain can be envisaged: a) to make a uniform mesh distribution in space with h fixed for all times and a variable mesh distribution in time with k chosen in such a way that the front is always kept at the rightmost node at any time, see Fig. 3.1; b) to fix the value of the time step (not necessarily constant for all times) and to search for the position of the front using a fixed space step for all times, except for the last node which is variable, see Fig. 3.2.

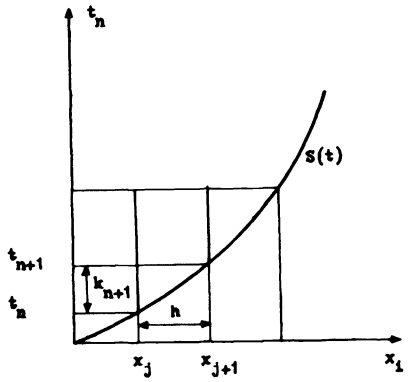


FIG. 3.1. *Uniform mesh distribution in space and variable mesh distribution in time.*

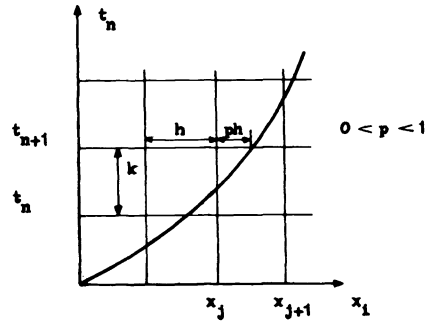


FIG. 3.2. *Fixed time step and uniform mesh in space except for the last cell which is variable.*

The advantages and inconveniences of these methods will be later discussed. Now we introduce the finite difference approximation utilized. Assuming that the front position and the function $u(x, t)$ are known at the n time level, as shown in Fig. 3.3, the problem

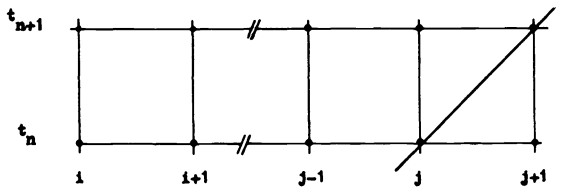


FIG. 3.3. *Front position at two consecutive time levels.*

is to obtain the front position and the function $u(x, t)$ at the next time level. To this end the initial-value problem (2.1)-(2.2) describing the front movement is approximated with a predictor-corrector scheme of the form

$$\frac{S^{*n+1}-S^n}{k}=\alpha F(S(t), u_x)^n, \tag{3.1}$$

$$\frac{S^{n+1}-S^n}{k}=\beta F(S^*(t), u_x^*)^{n+1}+(1-\beta)F(S(t), u_x) \tag{3.2}$$

where α and β are weighting factors lying in the interval $0 \leqq \alpha, \beta \leqq 1$, and $F(S(t), u_x)^n$ and $F(S(t^*), u_x^*)^{n+1}$ represent the right-hand side of equation (2.1) evaluated at the time level t_n and t_{n+1}, respectively. The intermediate function evaluation is obtained via the solution of (2.3) with an implicit time discretization scheme of the form

$$\frac{u_i^{*n+1}-u_i^n}{k}=\theta\,\delta u_i^{*n+1}+(1-\theta)\,\delta u_i^n, \tag{3.3}$$

where δ is the standard three-point centered finite difference operator and θ is a weighting factor lying in the interval $0 \leqq \theta \leqq 1$; for $\theta = 0, \frac{1}{2}$ and 1, explicit, Crank-Nicolson and strongly implicit schemes, respectively, are obtained. Thus, to advance the solution from the n to the $n+1$ time level, equations (3.1) to (3.3) are solved iteratively and convergence is achieved if in two successive iterations the values of the unknowns differ in less than a prescribed precision (usually 10^{-6}).

Returning to the discussion of the two methods described at the beginning of this section we note that in the first method, which is usually called a variable time step method, $t = t(S)$, therefore the initial-value problem (1.1)-(1.2) is formulated interchanging the role of the dependent and independent variables. In the second method, $S = S(t)$ and the initial-value problem remains in its original form. When the trajectory of the front has a slope $\dot{S}$ near unity any method will do; however, if $\dot{S}$ tends to zero (for instance when a stationary state is sought) clearly the first method cannot be used, since at some point of the calculations the independent variable will fall outside of the domain and the numerical method will fail to converge. Similarly, the second method will fail when $\dot{S}$ tends to infinity. In this work we discuss the first method; the second method is presented in Marshall and Smith [9]. Further details about the approximation used for the right-hand side of the predictor-corrector scheme (3.1)-(3.2) as well as for the difference scheme (3.3) are given in the appendix.

It can be shown that the finite difference system (3.1)-(3.2) and (3.3) is consistent with the differential problem (1.1)-(1.6). A linear stability analysis shows that a

necessary condition for the numerical stability of the corrector scheme (3.2) is

$$k \leqq \frac{2}{1-2\beta}\frac{1}{F_S} \quad \text{for } 0 \leqq \beta < \tfrac{1}{2}, \tag{3.4}$$

$$\text{no restriction} \quad \text{for } \beta \geqq \tfrac{1}{2}, \tag{3.5}$$

where F_S is the maximum value, between the time levels n and $n+1$, of the derivative of $F(S(t), u_x)$ with respect to S. This stability condition is valid only if F_S remains positive. In the same fashion a linear stability analysis shows that a necessary condition for the numerical stability of the implicit time discretization (3.3) is

$$k \leqq \frac{h^2}{2(1-2\theta)} \quad \text{for } 0 \leqq \theta < \tfrac{1}{2}, \tag{3.6}$$

$$\text{no restriction} \quad \text{for } \theta \geqq \tfrac{1}{2}. \tag{3.7}$$

The stability conditions (3.4) to (3.7) are valid for both methods except that, for the first method, in (3.4) F_S changes to F_t and the restriction is in the space step rather than in the time step. In passing we note that for concave up front trajectories F_t is always positive. Numerical evidence shows that the stability analysis presented remains valid for the nonlinear case.

4. Numerical results. We present numerical results obtained with the front tracking method applied to straight and curved front trajectories. The accuracy of the results obtained by the front tracking method is checked against exact and approximate analytical solutions; the efficiency of the method is compared with that of other numerical techniques.

The straight line front trajectory. This problem which is taken from Furzeland [5] and is due to Hoffman [7] is described by (1.1)-(1.6) with the following modification for the boundary condition (2.5)

$$\frac{\partial u}{\partial x} = -e^t, \qquad x = 0, \quad t > 0. \tag{4.1}$$

The exact solution of this problem is

$$u(x, t) = e^{t-x} - 1, \tag{4.2}$$

and the trajectory of the front in the x-t plane is the straight line

$$S(t) = t. \tag{4.3}$$

Figure 4.1 illustrates the exact solution in the x-t plane. The computations were done on an IBM 370/158 (under CMS-VM) with a Fortran compiler, in single precision, using the front tracking method. We found that the most accurate results were obtained using a three-point Lagrangian interpolation for the approximation of the front slope (all the calculations use this procedure). Table 1 shows the exact and computed position of the front, for different values of the mesh and of the numerical parameters. These results are shown to converge to 10^{-6} in seven iterations for $h = 0.1$, two iterations for $h = 0.025$ and one iteration for meshes not coarser than $h = 0.01$. This last rather remarkable result is mainly due to the presence of a predictor scheme which is consistent with the differential problem. The number of inner iterations is primarily a function of the precision sought. For fixed precision, the number of iterations decreases as the mesh is refined, for the simple reason that an initial guess based on the previous time level solution is closer to the final solution when the time step is smaller. Obviously,

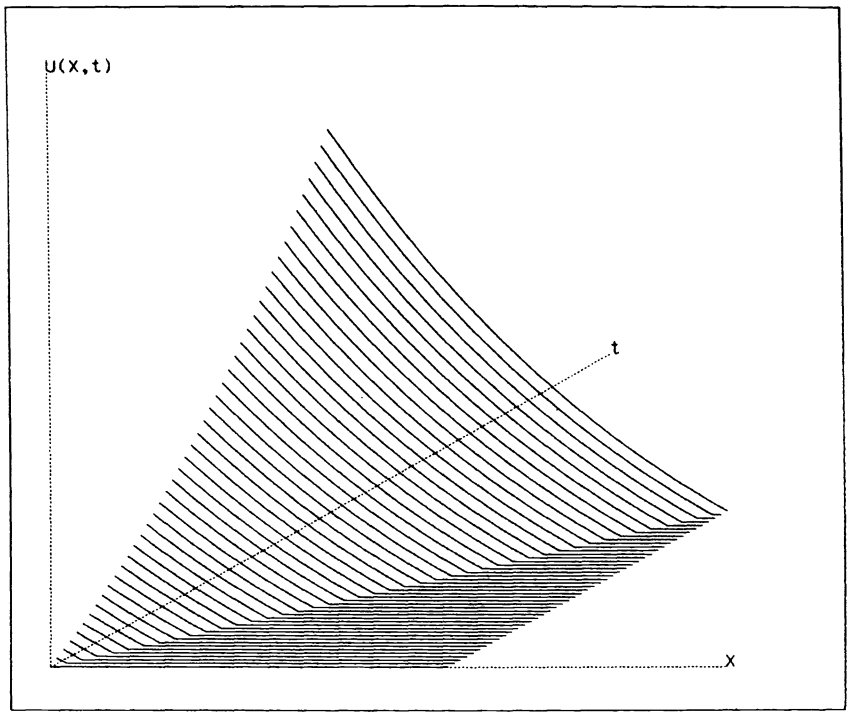


FIG. 4.1. *Exact solution in the x-t plane of the straight line front trajectory problem.*

TABLE 1
Position of the straight line front trajectory as calculated with the front tracking method.

Space	Time (computed) $h = 0.10$, $\beta = 0.88$, $\theta = 0.88$	Time (computed) $h = 0.025$, $\beta = 0.89$, $\theta = 0.89$	Time (computed) $h = 0.010$, $\beta = 0.90$, $\theta = 0.90$	Time (exact)
0.0	0.0000	0.0000	0.0000	0.0
0.1	0.1000	0.1001	0.1001	0.1
0.2	0.2000	0.2003	0.2002	0.2
0.3	0.3008	0.3005	0.3002	0.3
0.4	0.4013	0.4005	0.4002	0.4
0.5	0.5016	0.5005	0.5002	0.5
0.6	0.6017	0.6004	0.6002	0.6
0.7	0.7015	0.7003	0.7002	0.7
0.8	0.8012	0.8001	0.8001	0.8
0.9	0.9008	0.8999	0.9000	0.9
1.0	1.0002	0.9997	1.0000	1.0
CPU IBM-370/158 (seconds)	0.10	0.23	0.67	

for fixed precision and mesh size the number of inner iterations is a function of the numerical scheme used. It was also found that for fine meshes the weighting coefficients β and θ do not affect appreciably the number of inner iterations but do affect the accuracy of the results. In this connection it would be expected the most accurate

results to occur for values of β and θ near 0.5; however, the presence of oscillations, typical of Crank-Nicolson type schemes, makes that the most accurate results are attained for values of β and θ near 0.9 (Table 1 shows that the error—measured as the difference between the numerical and exact solutions—decreases in time to a minimum, changes its sign and increases thereafter). For values of β and θ equal to one the error increases monotonically (not shown here).

For comparative purposes we reproduce in Table 2 some of the results obtained by Furzeland [5] for the same problem using the Landau coordinate transformation, first with an implicit time discretization method (first and second columns), and then with the method of lines in space (third column). Regarding his first method, which is directly comparable to the front tracking method, Furzeland reports having reached a precision of 10^{-6} with two and three inner iterations. The front tracking method needs only one iteration and its results are more accurate, even more so considering that the word length used is shorter. The results of Furzeland's second method, the accuracy of which is nothing less than remarkable, can be compared with the front tracking results, for the same grid, in relation to accuracy and simplicity (leaving aside the computing time). In relation to accuracy, the superiority of Furzeland's results is only apparent considering that we have used single precision and a shorter word length (the NAG ODE package requires double precision arithmetic). The implementation of the method of lines in space is, indeed, very attractive: ease of programming and use of a robust ODE integrator package. So is the tracking method which consists in the solution of only one ODE involving the inversion of a tridiagonal system; it can be improved further using an automatic ODE solver rather than a predictor-corrector scheme. Moreover, the front tracking method has none of the inconveniences of the other methods discussed here. The latter need a coordinate transformation, and, in addition, the global accuracy of the method of lines depends on the spatial discretization. As this is refined, the ODE system increasingly stiffens. It is concluded in the light of the evidence presented that the front tracking method appears to be superior.

TABLE 2
Position of the straight line front trajectory as calculated by Furzeland [5].

	Space (computed)			
	Implicit time discretization			
Time	$h = 0.01$ $\theta = 1$	$h = 0.01$ $\theta = 0.5$	Method of lines $h = 0.025$	Space (exact)
0.1	0.1000	0.1003	0.1003	0.1
0.2	0.2001	0.2003	0.2003	0.2
0.5	0.5003	0.5003	0.5003	0.5
0.9	0.9006	0.9003	0.9003	0.9
1.0	1.0007	1.0003	1.0003	1.0
CPU ICL 1906A (seconds)	2	2	12	

The curved line front trajectory. This problem which is taken from Douglas and Gallie [4] is described by (1.1)-(1.6). An approximate analytic solution of this problem was given by Gupta and Kumar [6] using Goodman's integral method

$$u(x, t) = A(x - S) + B(x - S)^2, \tag{4.4}$$

where $S = S(t)$ and

$$A = \frac{1 - \text{sqrt}\,(1+4S)}{2S},$$

$$B = \frac{1+2S-\text{sqrt}\,(1+4S)}{4S^2}.$$

The trajectory of the front is given by

$$t = \frac{S}{6}(5+S+\text{sqrt}\,(1+4S)). \tag{4.5}$$

In Fig. 4.2 the approximate analytic solution is illustrated.

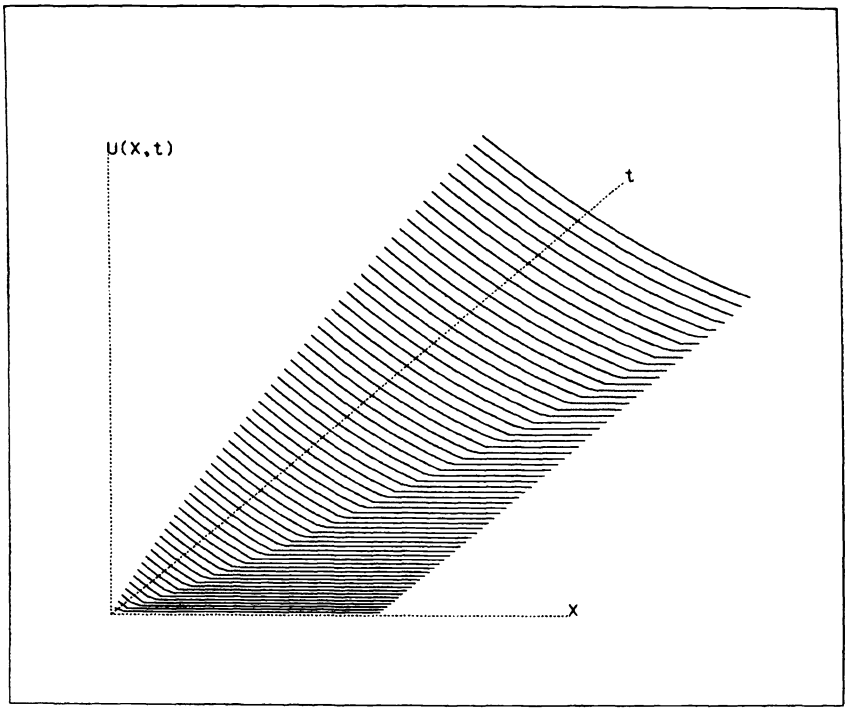


FIG. 4.2. *Approximate analytic solution in the x-t plane of the curved line front trajectory problem.*

Table 3 presents the results obtained with the front tracking method for different values of the mesh and the approximate analytic position of the front. We found that the optimum value of β and θ is one. The results converged to 10^{-6} in eight iterations for $h = 0.1$, four iterations for $h = 0.025$ and two iterations for $h = 0.01$. The number of inner iterations is higher as compared with the previous test problem because of the curved trajectory of the front. Moving along the rows of Table 3 (from coarser to finer meshes) it is possible to see that the results tend asymptotically to a fixed value, which is a practical proof of the convergence of the method. In passing we note that the approximate analytic solution gives good results only for values of $S(t)$ that are not too far from the origin. Table 4 shows the results obtained by Gupta and Kumar [6] for the same problem using an implicit time discretization and a first order approximation for the treatment of the front slope, together with the results obtained

TABLE 3
Position of the curved line front trajectory as calculated with the front tracking method.

Space	Time (computed) $h = 0.1$, $\beta = 1.0$, $\theta = 1.0$	Time (computed) $h = 0.025$, $\beta = 1.0$, $\theta = 1.0$	Time (computed) $h = 0.01$, $\beta = 1.0$, $\theta = 1.0$	Time (integral method)
0.0	0.0000	0.0000	0.0000	0.0000
0.3	0.3450	0.3421	0.3403	0.3392
0.6	0.7582	0.7491	0.7450	0.7444
0.8	1.0646	1.0507	1.0452	1.0568
1.0	1.3927	1.3745	1.3675	1.3727
1.5	2.3004	2.2724	2.2611	2.2865
2.0	3.3225	3.2862	3.2691	3.3333
2.5	4.4500	4.4062	4.3833	4.5069
3.0	5.6762	5.6257	5.5944	5.8028
CPU IBM-370/158 (seconds)	0.52	3.89	11.3	

TABLE 4
Position of the curved line front trajectory as calculated by Gupta and Kumar [6] *and with the front tracking method.*

Space	Time (computed) Gupta and Kumar [6] $h = 0.1$, $\theta = 1.0$	Time (computed) Gupta and Kumar [6] $h = 0.01$, $\theta = 1.0$	Time (computed) Tracking method $h = 0.1$, $\theta = 1.0$, $\beta = 1.0$	Time (computed) Tracking method $h = 0.01$, $\theta = 1.0$, $\beta = 1.0$	Time (integral method)
0.2	0.2091	0.2172	0.2230	0.2190	0.2181
0.6	0.7186	0.7406	0.7616	0.7458	0.7444
1.0	1.3285	1.3604	1.3937	1.3688	1.3727
1.6	2.3944	2.4413	2.4934	2.4559	2.4854
2.0	3.1993	3.2522	3.3176	3.2725	3.3333
2.6	4.5340	4.5916	4.6792	4.6211	4.7564
3.0	5.5035	5.5599	5.6696	5.5955	5.8028

with the front tracking method. The results of Gupta and Kumar converge to a precision of 0.5 percent apparently after several iterations (not specified by the authors); the tracking method results converge to the same precision in one iteration. Comparison with the results of the approximate analytic solution (for values near the origin), that can be considered the most accurate, reveals that of the two finite difference methods, front tracking is the most accurate and the fastest. This can be attributed to the use of a higher approximation for the treatment of the front slope, and of a predictor scheme which is consistent with the differential problem.

5. Conclusions. A new front tracking method for the numerical treatment of one-dimensional moving boundary problems has been presented. It consists in the formulation of the Stefan problem as an ordinary differential equation for the moving

boundary coupled to a parabolic equation for the diffusion process. A variable time step procedure has been introduced in which the ordinary differential equation is solved with a predictor-corrector scheme and the parabolic equation with an implicit time discretization. Numerical experiments with straight and curved front trajectories and comparison with other author's results evidenced that the front tracking method is an efficient, simple and accurate numerical procedure which produces greatly improved results.

Appendix. Here we discuss in some detail the implementation of the front tracking method. The right-hand side of (3.1) and (3.2) involves the approximation of the slope of the front as given by (2.1). This can be done, for instance, using linear or quadratic Lagrange polynomials:

$$\left(\frac{\partial u}{\partial x}\right)_j = (u_j - u_{j-1})/h, \tag{A.1}$$

$$\left(\frac{\partial u}{\partial x}\right)_j = (3u_j - 4u_{j-1} + u_{j-2})/(2h). \tag{A.2}$$

Equations (A.1) and (A.2) are two-point and three-point backward approximations of the first derivative of $u(x, t)$ at the j node. Using a fictitious node it is possible to obtain a centered approximation:

$$\left(\frac{\partial u}{\partial x}\right)_j = (u_{j+1} - u_{j-1})/(2h). \tag{A.3}$$

The fictitious centered approximation is locally the most accurate due to the nature of its truncation error; however, numerical evidence shows that with a quadratic Lagrange polynomial a better global accuracy is attained.

The implicit time discretization (3.3) of the parabolic equation (2.3) can be written (dropping the star superindex for convenience)

$$\frac{u_i^{n+1} - u_i^n}{k} = \frac{\theta}{h^2}(u_{i-1} - 2u_i + u_{i+1})^{n+1} + \frac{(1-\theta)}{h^2}(u_{i-1} - 2u_i + u_{i+1})^n, \tag{A.4}$$

which is valid for values of i lying in the interval $1 \leqq i \leqq j$. Rearranging (A.4) we obtain

$$-\theta r u_{i-1}^{n+1} + (1 + 2\theta r) u_i^{n+1} - \theta r u_{i+1}^{n+1} = D_i^n, \tag{A.5}$$

where

$$D_i^n = (1-\theta) r u_{i-1}^n + \{1 - 2(1-\theta) r\} u_i^n + (1-\theta) r u_{i+1}^n.$$

The boundary conditions are incorporated as follows. For $i = 1$ a centered difference approximation of (2.5) yields

$$\frac{u_2 - u_0}{2h} = -1. \tag{A.6}$$

Eliminating u_0 from (A.6) and (A.5) with $i = 1$, we get

$$(1 + 2\theta r) u_1^{n+1} - 2\theta r u_2^{n+1} = D_1^n, \tag{A.7}$$

where

$$D_1^n = \{1 - 2(1-\theta) r\} u_1^n + 2(1-\theta) r u_2^n - 2rh.$$

The right-hand boundary condition (2.6) at $i = j$ yields

$$-\theta r u_{j-1}^{n+1} + (1 + 2\theta r) u_j^{n+1} = D_j^n, \tag{A.8}$$

where D_j^n is the same expression as in (A.5) but for $i=j$. The value of u_{j+1}^n in (A.8) is obtained by an extrapolation procedure which uses (A.1), (A.2) or (A.3). Finally the tridiagonal system of order j becomes

$$\text{(A.9)} \qquad \begin{bmatrix} 1+2\theta r & -2\theta r & \\ -\theta r & 1+2\theta r & -\theta r \\ & -\theta r & 1+2\theta r \end{bmatrix} \begin{bmatrix} u_1^{n+1} \\ u_i^{n+1} \\ u_j^{n+1} \end{bmatrix} = \begin{bmatrix} D_1^n \\ D_i^n \\ D_j^n \end{bmatrix}.$$

This algebraic system can be easily inverted using an upper-lower triangular decomposition (see for instance Isaacson and Keller [8] or Richtmyer and Morton [13]).

It is worthwhile to examine the starting algorithm. Here we propose the following procedure. Let us assume the situation depicted in Fig. A1. We wish to compute the

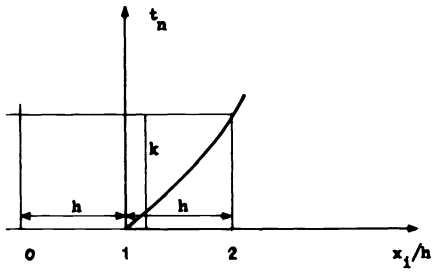


FIG. A1. *Scheme for the starting procedure.*

numerical solution at the time level t_{n+1}. To this end we write the following difference approximation at node $i=1$, for the initial-value problem (2.1) and for the parabolic equation (2.3) and its boundary conditions:

$$\text{(A.10)} \qquad \frac{S^{n+1}-S^n}{k} = -\frac{u_2^{n+1}-u_1^{n+1}}{h},$$

$$\text{(A.11)} \qquad -ru_0^{n+1}+(1+2r)u_1^{n+1}-ru_2^{n+1}=u_1^n,$$

$$\text{(A.12)} \qquad \frac{u_2^{n+1}-u_0^{n+1}}{2h}=-1,$$

$$u_2^{n+1}=u_1^n=0,$$

where we have used a first order approximation for the front slope and a strongly implicit time discretization for the parabolic equation. Recalling that in the variable time step method k is the unknown ($t_{n+1}-t_n=k$, time being the dependent variable), and $h=S^{n+1}-S^n$ is given, we obtain a nonlinear system of three equations with three unknowns: k, u_0^{n+1} and u_1^{n+1}. The nonlinearity of system (A.10)-(A.11)-(A.12) can easily be established replacing the value of $r=k/h^2$ obtained from (A.10) into (A.11). However, for this particular case, an explicit solution is available, by solving the quadratic equation in u_1^{n+1} (here called u for convenience)

$$\text{(A.13)} \qquad u^2+pu+q=0, \quad \text{where } p=2 \text{ and } q=-2h.$$

REFERENCES

[1] H. COHEN AND I. TADJBAKHSH, *Stefan problems in superconductivity and soldering*, Society of Engineering Science Meeting, Michigan State Univ., East Lansing, Nov. 2-4, 1964.

[2] J. CRANK, *Two methods for the numerical solution of moving-boundary problems in diffusion and heat flow*, Quart. J. Mech. Appl. Math., 10 (1957), pp. 220-231.

[3] J. CRANK, *How to deal with moving boundaries in thermal problems*, in Numerical Methods in Thermal Problems, R. W. Lewis and K. Morgan, eds., Pineridge Press, Swansea, 1979.

[4] J. DOUGLAS AND T. M. GALLIE, JR., *On the numerical integration of a parabolic differential equation subject to a moving boundary problem*, Duke Math. J., 22 (1955), pp. 557-571.

[5] R. M. FURZELAND, *A comparative study of numerical methods for moving boundary problems*, J. Inst. Maths. Applics., 26 (1980), pp. 411-429.

[6] R. S. GUPTA AND D. KUMAR, *A modified variable time step method for the one-dimensional Stefan problem*, Comp. Meth. Appl. Mech. Engng., 23 (1980), pp. 101-108.

[7] K. H. HOFFMAN, *Fachbereich Mathematik*, Vol. I-III, Berlin Freie Universitat, 1977.

[8] E. ISAACSON AND H. B. KELLER, *Analysis of Numerical Methods*, John Wiley, New York, 1966.

[9] G. MARSHALL AND L. SMITH, *A front tracking method for one-dimensional moving boundary problems* II, to appear.

[10] G. H. MEYER, *On a free interface problem for linear ordinary differential equations and the one-phase Stefan problem*, Numer. Math., 16 (1970), pp. 248-267.

[11] ———, *One-dimensional parabolic free boundary problems*, SIAM Rev., 19 (1977), pp. 17-34.

[12] J. R. OCKENDEN AND W. R. HODGKINS, eds., *Moving Boundary Problems in Heat Flow and Diffusion*, Clarendon Press, Oxford, 1967.

[13] R. D. RICHTMYER AND K. W. MORTON, *Finite Difference Methods for Initial-Value Problems*, Interscience, New York, 1967.

SIAM J. SCI. STAT. COMPUT.
Vol. 7, No. 1, January 1986

018

A GENERAL FAMILY OF NODAL SCHEMES*

J. P. HENNART†

Abstract. Nodal schemes have been originally developed in numerical reactor calculation, especially in the area of neutron diffusion problems. Broadly speaking, they constitute accurate and fast methods sharing many attractive features of the finite element as well as of the finite difference methods. In a previous work, some of the simplest nodal schemes were identified with nonstandard nonconforming finite element schemes exhibiting $O(h)$ convergence in H^1 norm. We present here a more general family of nodal schemes with $O(h^k)$ convergence for any positive integer k under appropriate smoothness assumptions. This new family is first introduced within a straightforward nonconforming finite element framework. Under special numerical quadrature schemes, we are then led to nodal schemes which can be obtained directly through basic physical principles. Finally, dimensionally reduced versions are obtained by transverse integration and stand as strong candidates to practical implementations of the alternating direction type.

Key words. nodal schemes, nonconforming finite elements, fast solvers, alternating direction schemes

1. Introduction. Modern coarse-mesh and (or) nodal methods have been developed during the 70's in numerical reactor calculation, especially in the area of neutron diffusion problems. The interested readers are referred to a recent review paper by J. Dorning [1] for a general discussion of the different methods available in that particular area of application as well as for the corresponding extensive bibliography.

As pointed out in [1], the distinction between coarse-mesh and nodal methods is so slight (and is in fact eliminated in the formalism proposed hereafter) that we shall exclusively speak in the following of "nodal" methods.

Broadly speaking, nodal methods are accurate and fast methods which share many attractive features of the finite element method (FEM) as well as of the finite difference method (FDM). With the FEM, they have in common the fact that the unknown function is approximated by a piecewise continuous, usually polynomial function over a given coarse mesh. Since the final equations are normally derived from physical considerations, the relationship with known finite element schemes is somewhat obscure and the tendency in the past has been to consider them as quite distinct methods. With the FDM, they share the fact that the resulting algebraic systems are in principle quite sparse and well structured: this is true of course for meshes not too irregular, like unions of rectangles, for which standard, i.e. conforming, finite elements would lead to much more coupled systems of equations.

In numerical reactor calculation, the domain $\Omega \subset \mathbb{R}^n$ with boundary $\Gamma = \bar{\Omega} - \Omega$ over which for instance the neutron group diffusion equation [2] are to be solved is usually fairly regular and can generally be discretized by a "triangulation" τ_h using a coarse rectangular mesh

$$\bar{\Omega} \sim \bar{\Omega}_h = \bigcup_{K \in \tau_h} K, \tag{1.1}$$

where the "nodes," "blocks," "cells" or "elements" are closed n-rectangles, which are as large as possible (whence the name "coarse-mesh methods") and over which homogenized properties (or "cross-sections" in the nuclear engineering jargon) are available. As nuclear reactors are actually quite heterogeneous, these homogenized

* Received by the editors November 29, 1983, and in final revised form September 24, 1984. This research was supported in part by the IBM Corporation Scientific Centers, Palo Alto, California and Mexico City, Mexico; by the Institut National de Recherche en Informatique et Automatique, France, and by the Mexican Consejo Nacional de Ciencia y Technología.

† IIMAS-UNAM, Apartado Postal 20-726, Delegación A. Obregón 01000 México, D.F. Mexico.

properties usually result from preliminary fine cell calculations at the subassembly level: after a nodal scheme has been applied to determine the coarse neutron flux distribution, a dehomogenization must normally be performed to recover the underlying fine flux distribution. The basic nuclear reactor type of calculation consists thus of three elementary stages, namely:

1. homogenization stage,
2. coarse mesh calculation stage,
3. dehomogenization stage.

In the potential applications to the oil reservoir simulation we have in mind, it seems well that the overall calculation boils down to the second stage, i.e. a coarse mesh calculation: the value of physical parameters of the reservoir such as porosity, permeability, etc. are not usually known with great accuracy and the typical oil engineering calculations assume that they are piecewise constant, i.e. that for each block only a constant or mean value of these parameters is available. The fact that most public [3] or private reservoir simulators still use finite differences certainly means that many practical situations can be modelled over domains of the type union of rectangles as above: for these domains, nodal schemes have proved to be far more efficient for a given accuracy than finite differences and standard (i.e. conforming) finite elements in static and dynamic diffusion calculations [1].

This certainly justifies the development and implementation of mathematically well founded nodal schemes which should hopefully lead to computer codes of the "fast solver" type, applicable to many practical oil reservoir situations.

In a recent paper [4], we established the equivalence between some simple nodal schemes, namely those described in [5], [6] and using sums of 1D polynomials as basis functions, with some nonconforming finite elements. These first though apparently limited results actually have deep practical consequences, as we shall see in § 2 of this paper. Understanding the limitations of these simple nodal schemes will in particular lead us to develop in § 3 an apparently new family of nodal schemes of any order, free of these limitations. As we shall show in the same section, the resulting algebraic equations already have much structure. This new family is introduced in § 3 within a straightforward nonconforming finite element framework, and the corresponding schemes are therefore named mathematical nodal schemes. If physical rather than mathematical arguments are used to derive the final algebraic equations, we obtain the physical nodal schemes of § 4, where it is shown in particular that any physical nodal scheme corresponds to a mathematical nodal scheme associated to some particular numerical quadrature of the matrix elements, leading to particularly simple (i.e. sparse) algebraic equations. In § 5, dimensionally reduced versions are obtained by transverse integration as in [1] and they clearly stand as strong candidates to practical implementations of the alternating direction type.

To keep this first paper within reasonable limits, we limited ourselves to a typical second order elliptic equation

$$Lu \equiv -\nabla \cdot p\nabla u + qu = f \quad \text{on } \Omega \subset \mathbb{R}^n \tag{1.2a}$$

subject to boundary conditions

$$u = 0 \quad \text{on } \Gamma_1, \tag{1.2b}$$

$$\frac{\partial u}{\partial n} = 0 \quad \text{on } \Gamma_2, \tag{1.2c}$$

where $\Gamma = \bar{\Omega} - \Omega = \Gamma_1 \cup \Gamma_2$ with meas $(\Gamma_1) \neq 0$ and $\Gamma_1 \cap \Gamma_2 = \varnothing$. Most of our results can

readily be extended to parabolic equations of the type

$$\frac{\partial u}{\partial t} + Lu = f. \tag{1.3}$$

Extensions to hyperbolic equations are at first sight less clear but quite feasible as we believe. If a fully nodal simulator is to be applied to oil reservoir simulation, we clearly need practical nodal schemes for each kind of equation, although a good elliptic and (or) parabolic fast solver would already be quite helpful for the pressure equation.

As a rule we assumed the coefficients p and q to be constant or at least piecewise constant: the considerations on the structure of the elementary mass and stiffness matrices found in § 3 and 4 are clearly valid in that case. This is also true for the structures obtained as the result of some particular quadrature schemes, independently of p and q. Since in the piecewise constant coefficient case the exact solution belongs at most to the Sobolev space $V = \{v \in H^1(\Omega);\ v = 0 \text{ on } \Gamma_1\}$, no extra smoothness can be assumed and the interest or efficiency of higher order schemes can only be tested numerically. The theoretical convergence aspects in presence of smoothness and the practical implementation problems with numerical results in realistic situations will appear subsequently, as we decided to first present this new family of nodal schemes with only qualitative statements on its convergence properties and numerical implementation aspects.

2. Some background material on nodal schemes. As we consider the typical elliptic problem described by (1.2), we shall in the following restrict ourselves to the two-dimensional case ($\Omega \subset \mathbb{R}^2$) for the sake of simplicity. The nodal schemes presented in [5] and [6] when applied to (1.2) consist in looking for a finite-dimensional approximation $u_h \in V_h$ of u, which in the so-called sum or Σ-schemes can be described as follows: the restriction $u_{h/K}$ of $u_h \in V_h$ to the node $K \in \tau_h$ are finite-dimensional spaces $P_K = \{u_{h/K};\ u_k \in V_h\}$ spanned by sums of one-dimensional polynomials of degree $k \geqq 2$, namely

$$P_K = \mathcal{Q}_{k,0} \cup \mathcal{Q}_{0,k}, \tag{2.1}$$

where $\mathcal{Q}_{k,l} \equiv \{x^\alpha y^\beta,\ \alpha \leqq k,\ \beta \leqq l\}$. In the original presentation [5], [6], $u_{h/K}$ was defined in terms of a set Σ_K of basic parameters which were the value of u_h at the barycenter of the node K (or the average of u_h over K) and some of the moments of u_h over K if $k > 2$, as well as the values of u_h at the midpoints of the faces, u_{h_L}, u_{h_R}, u_{h_D} and u_{h_U} (L, R, D & U standing for Left, Right, Down & Up, Front and Back being used also in 3D). For reasons which will be clear later, we shall use instead the mean values of u_h over the faces of the nodes. In accordance with § 3, where a precise definition of these moments will be given, including a quite convenient normalization, these parameters will be denoted m_B^0 where $B \equiv L, R, D$ or U and m_C^{ij}, C standing for Cell, m_C^{00} being in particular the average value of u_h over the given cell (see Fig. 1). In the general case, for the nodal scheme of type (k)

$$\begin{aligned} &P_K = \mathcal{Q}_{k,0} \cup \mathcal{Q}_{0,k}, \qquad \dim P_K = 2k+1 \\ &\Sigma_K = \{m_L^0, m_R^0, m_U^0, m_D^0, m_C^{ij},\ i = 0, \cdots, k-2 \text{ with } j = 0 \\ &\qquad\qquad \text{and } i = 0 \text{ with } j = 1, \cdots, k-2\}, \\ &\text{card } \Sigma_K = 2k+1, \end{aligned} \tag{2.2}$$

and it is easy to check that any polynomial in the space P_K is uniquely determined

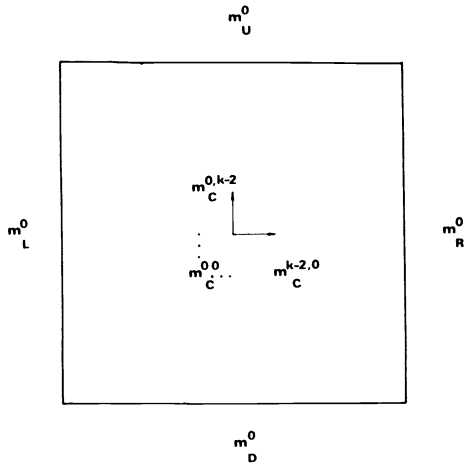


FIG. 1. Σ *nodal schemes of type* (k), $k \geqq 2$.

by its moments constituting Σ_K. In addition, only the inclusion

$$\mathcal{P}_1 \subset P_K \tag{2.3}$$

holds, independently of k, where $\mathcal{P}_k \equiv \{x^\alpha y^\beta,\ \alpha+\beta \leqq k\}$.

In most papers dealing with nodal methods, as for instance [5] and [6], the final nodal equations are derived in agreement with the physics of the problem, by expressing that the mean value of u_h and of the corresponding mean values of the normal flux $q\cdot n = -p\nabla u\cdot n$ are continuous through the faces of the nodes, and also that some weighted balance equations over the nodes are satisfied. With the above choice of parameters, the continuity of the mean values of u_h is trivial as soon as the same parameter is chosen on the common face of two adjacent nodes. The final nodal equations reduce thus to the mean normal flux continuity equations through the node faces:

$$\int_{\Gamma_{ij}=K_i\cap K_j} [p\nabla u_h|_{K_i} - p\nabla u_h|_{K_j}]\cdot 1_n\, ds = 0, \tag{2.4}$$

$\forall K_i, K_j \in \tau_h$ with $K_i \cap K_j = \Gamma_{ij} \neq \varnothing$ (in other words, K_i & K_j are adjacent nodes), and where 1_n is some (arbitrary) unit normal to ∂K_{ij}, and to the (eventually weighted) cell balance equations

$$\int_K w_{ij}[-\nabla\cdot p\nabla u_h + qu_h - f]\, dx = 0 \quad \forall K \in \tau_h, \tag{2.5}$$

where w_{ij}'s are weighting polynomials of degree $\leqq k-2$ in x or y, namely

$$w_{ij}(x,y) \equiv x^i y^j, \qquad i = 0,\cdots,k-2 \text{ with } j=0 \text{ and } i=0 \text{ with } j=1,\cdots,k-2. \tag{2.6}$$

Clearly the w_{ij}'s span $\mathcal{Q}_{k-2,0}\cup\mathcal{Q}_{0,k-2}$ and in (2.6) x and y are reduced coordinates over K. For the boundary nodes, the mean value on Γ_1 (and eventually the mean flux on Γ_2) is taken to be zero.

This physical way of deriving the nodal schemes of the Σ or sum type is certainly quite attractive as it automatically ensures some average balance and continuity properties. In the following, we shall speak of the physical nodal method (PNM) as we refer to these particular schemes. Clearly, the PNM is closely related to the FDM when the finite difference equations are obtained by box integration (see e.g. [7]).

A more mathematical way to approximate the solution of the problem at hand consists in considering the weak form of (1.2): in other words, $u \in V \equiv \{v \in H^1(\Omega), v = 0 \text{ on } \Gamma_1\}$ is looked for such that

$$a(u, v) = f(v) \quad \forall v \in V, \tag{2.7}$$

where $a(\cdot, \cdot)$ & $f(\cdot)$ are the usual bilinear and linear forms

$$(u, v) = \int_\Omega (p \nabla u \cdot \nabla v + quv)\, dx, \tag{2.8a}$$

$$f(v) = \int_\Omega fv\, dx. \tag{2.8b}$$

In standard, i.e. *conforming*, situations, a finite-dimensional V_h subspace of V is considered and a discretized form of the above problem is to find $u_h \in V_h$ such that

$$a(u_h, v_h) = f(v_h) \quad \forall v_h \in V_h. \tag{2.9}$$

If we choose to approximate V by the finite-dimensional space V_h defined at the beginning of this section by the restrictions of its members to a particular node K, $V_h \not\subset V$ because u_h is not continuous through the faces of K (only its mean value is). We are thus in a *nonconforming* situation, where the discrete equations (2.9) can still be used provided $a(\cdot, \cdot)$ is replaced by $a_h(\cdot, \cdot)$ where

$$a_h(u, v) = \sum_{K \in \tau_h} \int_K (p \nabla u \cdot \nabla v + quv)\, dx, \tag{2.10}$$

since $a(u, v)$ would have no meaning for $u, v \in V_h$. The convergence of this approximate solution, obtained by the so-called mathematical nodal method or MNM in the following, is guaranteed by the mean value continuity through the faces of the nodes: consequently, the "patch test" [8], [9] is passed, and this justifies a posteriori the minor modifications to the set Σ_K of basic parameters proposed hereabove.

In [4], we proved that the PNM was in fact a MNM in which the matrix elements were not calculated exactly but only approximately by using some numerical quadrature of the Radau type in a nonstandard way. Some similar results will be proven in § 4 for a much more general family of nodal schemes.

Once the simple nodal schemes of [5], [6] had been identified with finite element schemes combining two well-known "variational crimes," namely nonconformity ($V_h \not\subset V$) and numerical quadrature ($a_h(\cdot, \cdot) \sim \tilde{a}_h(\cdot, \cdot)$), the door was clearly open for a complete numerical analysis of these schemes, following for instance Chapter 4 of Ciarlet's book [10]. As far as we know, such an analysis was not yet available: it is sketched (with some minor mistakes) in [4] and given with more details in [11]. In practice, the numerical quadrature needed to relate the MNM and the PNM are always of high accuracy and their effect on the convergence of a given scheme is not crucial: they do however affect in a very definite way the structure of the resulting systems of algebraic equations and their impact on practical implementations is therefore quite important. If nonconformity only is considered as with the MNM, the error is by virtue of the second Strang lemma [10, p. 210]

$$\|u - u_h\|_h \leqq C \left(\inf_{v_h \in V_h} \|u - v_h\|_h + \sup_{w_h \in V_h} \frac{|a_h(u, w_h) - f(w_h)|}{\|w_h\|_h} \right), \tag{2.11}$$

where

$$\|v_h\|_h = \left(\sum_{k \in \tau_h} |v_h|^2_{1,K} \right)^{1/2}, \tag{2.12}$$

with

$$|v_h|_{1,K} = \left(\int_K \nabla v_h \cdot \nabla v_h \, dx \right)^{1/2}. \tag{2.13}$$

A priori the mapping

$$v_h \to \|v_h\|_h, \tag{2.14}$$

is only a semi-norm over the space V_h, but it can be shown to be actually a norm (see e.g. [11]) for nonconforming elements ensuring mean value continuity. In that case, the bound (2.11) is valid. Clearly, it is in two parts. The first term, apart from a different value for the constant, corresponds to the term in Cea's lemma [10, p. 104], i.e. the only one we obtain with conforming finite elements: it can be bounded by well-known interpolation theory arguments. Assuming that the exact solution has enough regularity, the nodal Σ-schemes of any index k will only be able to reproduce polynomials of degree 1 because of (2.3) and the first term in (2.11) will thus be of $O(h)$, independently of k. The second term in (2.11) arises from the nonconformity of the method and may be interpreted as a *consistency error.* The fact that the nodal Σ-schemes only ensure the continuity of the average u between adjacent elements can be shown [11] to lead to an estimate of $O(h)$ for the second term in (2.11), again independently of k. Aubin-Nitsche like arguments would thus lead at most to an error estimate in L^2 norm of $O(h^2)$, independently of k. Numerical results were given in [4], [11], which confirm this theoretical analysis and show that when k is increased, the approximation is eventually better but that because of the $O(h^2)$ estimate valid for all index k, some saturation effect is clearly noticeable. In numerical reactor calculation where piecewise continuous material properties over polygonal domains are the rule, boundary and interface singularities prevent the solution from being very smooth [12] and the above arguments are presumably valid only for the smooth part of the solution, which is well observed far from asymptotic [13]. In those cases, the practical efficiency of a given scheme can only be decided after extensive numerical experiments, where an important aspect is the ease of its practical implementation. For the nodal Σ-schemes recalled in this section, this was certainly a strong argument in their favor as they provided final algebraic systems quite reminiscent of the 5-point (or 7-point in 3D) finite difference equations. The readers interested in more details are referred to the original papers [5], [6]. In [4], we show how these particular simple algebraic structures arise from some reduced integration schemes [14, Chap. 20], quite similar to the ones proposed in § 3 and 4 of this paper.

The above discussion implies that the first nodal schemes exemplified by the nodal Σ-schemes of [5] and [6] have important built-in limitations. The nuclear engineers involved in their early development noticed that with these schemes the physical leakage through the faces of the nodes were only constant: with the terminology used in § 5, the "transverse leakage" is a constant for each face of a given cell. In more recent nodal schemes, the unknowns on the faces of the nodes basically remain the same ones as above, i.e. average or mid-node values, but the transverse leakage representation is improved by a quadratic fit involving the neighbor cells. This idea has been quite successful and is now considered as standard. It is however more difficult to formalize and analyze, and we prefer our approach of § 3, where transverse leakage of any

polynomial degree can be directly built into the individual cell. This is done by introducing higher order moments of u on the cell faces, say the $(k+1)$ first moments, so that a higher order patch test is passed and the consistency error due to nonconformity will now be of $O(h^{k+1})$. A consistent interpolation error is obtained if the inclusion (2.3) is replaced by

$$\mathcal{P}_{k+1} \subset P_K, \tag{2.15}$$

showing in particular that we need cross-terms like xy in P_K, a fact that most nuclear engineering papers do not admit. With moments up to order $(k+1)$ on the faces and the inclusion (2.15), we can expect errors in norm L_2 of $O(h^{k+2})$, as numerical experiments do confirm [15].

The main difficulty remains to pick up the correct spaces P_K such that a given set of linear forms Σ_K defined on K and with card $\Sigma_K = \dim P_K$ would lead to unisolvency. A family of such spaces of any index k is developed and its properties, analyzed in § 3.

Before leaving this section, a last comment should be made. Recognizing that some well-known or less well-known nodal schemes are particular nonconforming finite element schemes with consistent or reduced integration has a very practical consequence: these nodal schemes can actually be imbedded in general purpose finite element codes and thus compared with more or less classical finite element schemes in basically the same environment. After reflecting upon our past experiences reported in [16], [17], we wrote such a general purpose modular code during the spring of 1982. This code was used to produce the results reported in [4], [11] and [18] and a first global and fair comparison between the best finite element and nodal schemes finally seems to be at hand.

3. A general family of nodal schemes in 2D.

3.1. The general case. For a function $u(x, y)$ defined over the reference square cell $K \equiv [-1, +1] \times [-1, +1]$, we shall introduce the following linear forms (see Fig. 2):

$$m_L^i = \int_{-1}^{+1} P_i(y) u(-1, y)\, dy \Big/ \int_{-1}^{+1} P_i^2(y)\, dy, \qquad i = 0, \cdots, k, \tag{3.1a}$$

$$m_R^i = \int_{-1}^{+1} P_i(y) u(+1, y)\, dy \Big/ \int_{-1}^{+1} P_i^2(y)\, dy, \qquad i = 0, \cdots, k, \tag{3.1b}$$

$$m_D^i = \int_{-1}^{+1} P_i(x) u(x, -1)\, dx \Big/ \int_{-1}^{+1} P_i^2(x)\, dx, \qquad i = 0, \cdots, k, \tag{3.1c}$$

$$m_U^i = \int_{-1}^{+1} P_i(x) u(x, +1)\, dx \Big/ \int_{-1}^{+1} P_i^2(x)\, dx, \qquad i = 0, \cdots, k, \tag{3.1d}$$

as well as

$$m_C^{ij} = \int_{-1}^{+1} \int_{-1}^{+1} P_i(x) P_j(y) u(x, y)\, dx\, dy \Big/ \int_{-1}^{+1} P_i^2(x)\, dx \cdot \int_{-1}^{+1} P_j^2(y)\, dy, \tag{3.1e}$$

$$i, j = 0, \cdots, k,$$

where the subindices mean "left," "right," "down," "up," and "cell" respectively. In 3D, we shall also need "front" and "back:" this explains why we preferred "up" and "down" to "top" and "bottom." These linear forms constitute a set Σ with $N \equiv \text{card } \Sigma = (k+1)(k+5)$, where the P_i's are the normalized Legendre polynomials over $[-1, +1]$.

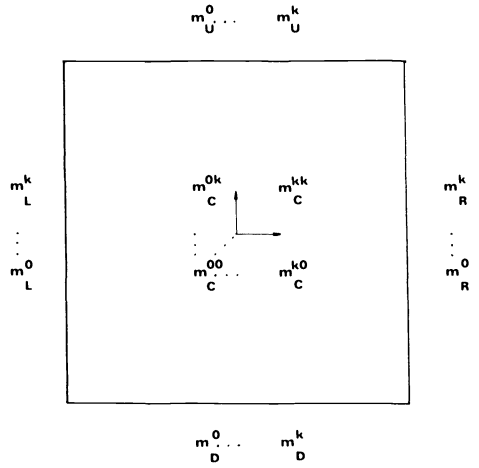


FIG. 2. *Nodal scheme of type* (k), $k \geqq 0$.

For a given integer $k \geqq 0$, the general nodal function of index k will be given as

$$u_h(x, y) = \sum_{i=0}^{k} [m_L^i \cdot u_L^i(x, y) + m_R^i \cdot u_R^i(x, y) + m_D^i \cdot u_D^i(x, y) + m_U^i \cdot u_U^i(x, y)] + \sum_{i=0}^{k} \sum_{j=0}^{k} m_C^{ij} \cdot u_C^{ij}(x, y), \tag{3.2}$$

where u_L^i, u_R^i, u_D^i, u_U^i and u_C^{ij} all belong to the space

$$P \equiv \mathcal{Q}_{k+2,k} \cup \mathcal{Q}_{k,\,k+2}, \tag{3.3}$$

$\mathcal{Q}_{k,l}$ being spanned by $x^\alpha y^\beta$, $\alpha \leqq k$, $\beta \leqq l$. Clearly

$$\begin{aligned} \dim (\mathcal{Q}_{k+2,k} \cup \mathcal{Q}_{k,k+2}) &= \dim \mathcal{Q}_{k+2,k} + \dim \mathcal{Q}_{k,k+2} - \dim \mathcal{Q}_{k,k} \\ &= (k+3)(k+1) + (k+1)(k+3) - (k+1)^2 \\ &= (k+1)(k+5), \end{aligned} \tag{3.4}$$

so that $\dim P = \operatorname{card} \Sigma$.

To show that the triple (K, P, Σ) is a finite element, it is sufficient to exhibit the corresponding basis functions since $\dim P = \operatorname{card} \Sigma$.

Let us first consider the basis functions associated to the boundary parameters (L, R, D or U) as for instance u_L^i. A general form for u_L^i is

$$u_L^i(x, y) = \sum_{j=0}^{k+2} p_{Lj}^i(x) P_j(y), \tag{3.5}$$

where

$$\begin{aligned} &p_{Lj}^i(x), \qquad j = 0, \cdots, k \in \mathcal{P}_{k+2}(x), \\ &p_{Lj}^i(x), \qquad j = k+1, k+2 \in \mathcal{P}_k(x), \end{aligned} \tag{3.6}$$

$\mathcal{P}_l(x)$ being the space of polynomials in x of degree less or equal to l. Clearly the number of free parameters in (3.5) is $(k+1)(k+5)$ as expected.

For u_L^i, we have in particular

a. *Left and right boundary values.*

$$\int_{-1}^{+1} P_l(y) \sum_{j=0}^{k+2} p_{Lj}^i(-1) P_j(y)\, dy = \frac{2}{2l+1}\delta_{il}, \qquad l=0,\cdots,k,$$

or

$$p_{Lj}^i(-1) = \delta_{ij}, \qquad j=0,\cdots,k, \tag{3.7}$$

where we used the fact that

$$\int_{-1}^{+1} P_l^2(y)\, dy = \frac{2}{2l+1}, \qquad \int_{-1}^{+1} P_l(y) \sum_{j=0}^{k+2} p_{Lj}^i(+1) P_j(y)\, dy = 0, \qquad l=0,\cdots,k,$$

or

$$p_{Lj}^i(+1) = 0, \qquad j=0,\cdots,k. \tag{3.8}$$

b. *Down and up boundary values.*

$$\sum_{j=0}^{k+2} \int_{-1}^{+1} p_{Lj}^i(x) P_l(x)\, dx = 0, \qquad l=0,\cdots,k, \tag{3.9}$$

and

$$\sum_{j=0}^{k+2} (-1)^j \int_{-1}^{+1} p_{Lj}^i(x) P_l(x)\, dx = 0, \qquad l=0,\cdots,k, \tag{3.10}$$

where we used the fact that $P_j(-1) = (-1)^j$.

c. *Cell values.*

$$\sum_{j=0}^{k+2} \int_{-1}^{+1} \int_{-1}^{+1} p_{Lj}^i(x) P_j(y) \cdot P_l(x) P_m(y)\, dx\, dy = 0, \qquad l, m = 0,\cdots,k$$

or

$$\int_{-1}^{+1} p_{Lj}^i(x) P_l(x)\, dx = 0, \qquad j, l = 0,\cdots,k. \tag{3.11}$$

From (3.9), (3.10) and (3.11), we directly deduce that for $j = k+1, k+2$, $p_{Lj}^i(x) \in \mathcal{P}_k(x)$ is orthogonal to $P_l(x)$, $l=0,\cdots,k$ and therefore identical to zero. For $j = 0,\cdots,k$, $p_{Lj}^i(x) \in \mathcal{P}_{k+2}(x)$ must satisfy (3.7) and (3.8) and is also orthogonal to $P_l(x)$, $l=0,\cdots,k$ because of (3.11). From these orthogonality relations, we deduce that

$$p_{Lj}^i(x) = \alpha_{Lj}^i P_{k+1}(x) + \beta_{Lj}^i P_{k+2}(x), \qquad j=0,\cdots,k, \tag{3.12}$$

where

$$\alpha_{Lj}^i (-1)^{k+1} + \beta_{Lj}^i (-1)^{k+2} = \delta_{ij}, \qquad j=0,\cdots,k, \tag{3.13}$$

and

$$\alpha_{Lj}^i + \beta_{Lj}^i = 0, \qquad j=0,\cdots,k. \tag{3.14}$$

Solving (3.13) and (3.14), we get

$$\begin{aligned} \alpha_{Lj}^i &= +\tfrac{1}{2}(-1)^{k+1}\delta_{ij}, \\ \beta_{Lj}^i &= -\tfrac{1}{2}(-1)^{k+1}\delta_{ij}, \end{aligned} \tag{3.15}$$

so that $p_{Lj}^i(x) \equiv 0$ unless $j = i$.

In summary,

$$u_L^i(x, y) = p_L^i(x) P_i(y), \tag{3.16}$$

where

$$p_L^i(x) = \alpha_L^i (P_{k+1}(x) - P_{k+2}(x)), \tag{3.17}$$

with

$$\alpha_L^i = \tfrac{1}{2}(-1)^{k+1}. \tag{3.18}$$

Similar arguments lead to

$$u_R^i(x, y) = p_R^i(x) P_i(y), \tag{3.19}$$

where

$$p_R^i = \alpha_R^i (P_{k+1}(x) + P_{k+2}(x)), \tag{3.20}$$

with

$$\alpha_R^i = \tfrac{1}{2}. \tag{3.21}$$

$u_D^i(x, y)$ and $u_U^i(x, y)$ are then obtained from $u_L^i(x, y)$ and $u_R^i(x, y)$ by a simple permutation of x and y in (3.16) and (3.19).

In order to obtain the basis functions u_C^{ij}, let us consider the more symmetric expression

$$\begin{aligned} u_C^{ij}(x, y) = & \sum_{m=0}^{k} \sum_{n=0}^{k} a_{mn}^{ij} P_m(x) P_n(y) \\ & + \sum_{m=1}^{2} \sum_{n=0}^{k} b_{mn}^{ij} P_{k+m}(x) P_n(y) \\ & + \sum_{m=0}^{k} \sum_{n=1}^{2} c_{mn}^{ij} P_m(x) P_{k+n}(y), \end{aligned} \tag{3.22}$$

where again the number of free parameters is $(k+1)(k+5)$.

For u_C^{ij}, we have

a. *Left and right boundary values.*

$$\int_{-1}^{+1} P_l(y) \cdot \left\{ \sum_{m=0}^{k} \sum_{n=0}^{k} a_{mn}^{ij} (-1)^m P_n(y) + \sum_{m=1}^{2} \sum_{n=0}^{k} b_{mn}^{ij} (-1)^{k+m} P_n(y) + \sum_{m=0}^{k} \sum_{n=1}^{2} c_{mn}^{ij} (-1)^m P_{k+n}(y) \right\} dy = 0, \qquad l = 0, \cdots, k,$$

or simply

$$\sum_{m=0}^{k} a_{ml}^{ij} (-1)^m + \sum_{m=1}^{2} b_{ml}^{ij} (-1)^{k+m} = 0, \qquad l = 0, \cdots, k, \tag{3.23}$$

and the corresponding expression on the right boundary

$$\sum_{m=0}^{k} a_{ml}^{ij} + \sum_{m=1}^{2} b_{ml}^{ij} = 0, \qquad l = 0, \cdots, k. \tag{3.24}$$

b. *Down and up boundary values.*

$$\int_{-1}^{+1} P_l(x) \cdot \left\{ \sum_{m=0}^{k} \sum_{n=0}^{k} a_{mn}^{ij} P_m(x) \cdot (-1)^n + \sum_{m=1}^{2} \sum_{n=0}^{k} b_{mn}^{ij} P_{k+m}(x) \cdot (-1)^n + \sum_{m=0}^{k} \sum_{n=1}^{2} c_{mn}^{ij} P_m(x) \cdot (-1)^{k+n} \right\} dx = 0, \cdots, \quad l = 0, \cdots, k,$$

or simply

$$\sum_{n=0}^{k} a_{ln}^{ij} (-1)^n + \sum_{n=1}^{2} c_{ln}^{ij} (-1)^{k+n} = 0, \qquad l = 0, \cdots, k, \tag{3.25}$$

and the corresponding expression on the right boundary

$$\sum_{n=0}^{k} a_{ln}^{ij} + \sum_{n=1}^{2} c_{ln}^{ij} (-1)^{k+n} = 0, \qquad l = 0, \cdots, k. \tag{3.26}$$

c. *Cell values.*

$$\int_{-1}^{+1} \int_{-1}^{+1} P_l(x) P_m(y) \cdot \left\{ \sum_{p=0}^{k} \sum_{q=0}^{k} a_{pq}^{ij} P_p(x) P_q(y) + \sum_{p=1}^{2} \sum_{q=0}^{k} b_{pq}^{ij} P_{k+p}(x) P_q(y) + \sum_{p=0}^{k} \sum_{q=1}^{2} c_{pq}^{ij} P_p(x) P_{k+q}(y) \right\} dx\, dy$$

$$= \frac{4}{(2l+1)(2m+1)} \delta_{il} \delta_{jm}, \qquad l, m = 0, \cdots, k,$$

or

$$a_{mn}^{ij} = \delta_{im} \delta_{jn}, \qquad m, n = 0, \cdots, k. \tag{3.27}$$

Clearly, the $(k+1)^2$ coefficients a_{lm}^{ij} are given by (3.27) and (3.23) and (3.24), as well as (3.25) and (3.26), provide the $4(k+1)b_{lm}^{ij}$'s and c_{lm}^{ij}'s as soon as we know the a_{lm}^{ij}'s. Trivial algebraic manipulations lead to

$$b_{mn}^{ij} = -\tfrac{1}{2} \delta_{jn} [1 + (-1)^{i-k-m}], \qquad m = 1, 2, \quad n = 0, \cdots, k,$$

and

$$c_{mn}^{ij} = -\tfrac{1}{2} \delta_{im} [1 + (-1)^{j-k-n}], \qquad m = 0, \cdots, k, \quad n = 1, 2. \tag{3.28}$$

After some other algebraic manipulations, we finally get:

$$u_C^{ij}(x, y) = P_i(x) P_j(x) - P_{k+m(i)}(x) P_j(y) - P_i(x) P_{k+m(j)}(y), \tag{3.29}$$

where $m(i)$ is equal to 1 or 2 and such that $m(i)+k$ and i have the same parity (odd or even).

Before leaving this section, we would like to mention that the general family of elements developed here can be extended to include elements with different degrees in x and y. Hereabove, the degree in x and y of the monomials present in the basis was at most $k+2$ for both x and y, but there may exist situations with thin or thick rectangles where it would be more natural to use different maximal degrees in x and y, say $k+2$ and $l+2$. Possible applications will be described in [19]. Appendix A gives the expressions for the basis functions, in the (k, l) general case: these expressions

reduce of course to the expressions derived in this section when $k = l$. Some particular examples with $k = l$ and $k \neq l$ are given in Appendix B, while trivial reductions to 1D and extensions to 3D are given in Appendix C.

3.2. General properties of the corresponding nonconforming schemes.

a. *Basis functions associated to boundary values.* Let us consider for instance the basis functions $u_L^i(x, y)$ associated to the left ($x = -1$) boundary values m_L^i, $i = 0, \cdots, k$. From (3.16), it is clear that

$$u_L^i \propto [P_{k+1}(x) - P_{k+2}(x)] \cdot P_i(y), \qquad i = 0, \cdots, k, \tag{3.30}$$

so that $u_L^i = 0$ as soon as x is one of the $(k+2)$ Right Radau points x_l^{RR}, $l = 1, \cdots, k+2$, including $x = +1$, or y is one of the i (assuming $i > 0$) Gauss points y_m^G, $m = 1, \cdots, i$. The zeros of u_L^i are shown in Fig. 3 for $k = 0$, 1 and 2.

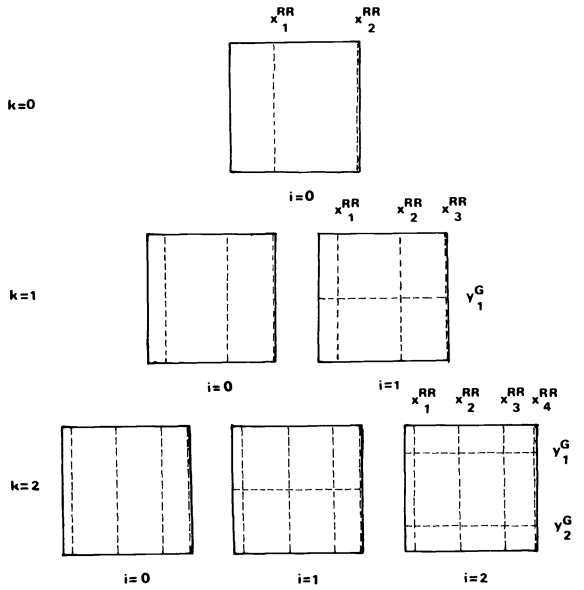


FIG. 3. *Zeros of* u_L^i, $i = 0, \cdots, k$ *for* $k = 0$, 1 *and* 2.

b. *Basis functions associated to cell values.* Using the general expression (3.29), it is easy to see that

$$u_c^{ij}(x, \pm 1) \propto P_{k+m(i)}(x), \tag{3.31a}$$

and

$$u_c^{ij}(\pm 1, y) \propto P_{k+m(j)}(y), \tag{3.31b}$$

so that $u_c^{ij} = 0$ on the boundary of the reference square when x (resp. y) is one of the $l = k + m(i)$ (resp. $k + m(j)$) Gauss points x_n^G (resp. y_n^G), $n = 1, \cdots, l$ for y (resp. x) $= \pm 1$. The zeros of some of the u_c^{ij}'s are shown in Fig. 4 for $k = 1$.

c. *Relationships between the boundary and cell basis functions.* Let us define $P_{lm}(x, y)$ as follows

$$P_{lm}(x, y) \equiv P_l(x) P_m(y). \tag{3.32}$$

Using the definition of the boundary and cell basis functions, it is easy to get the

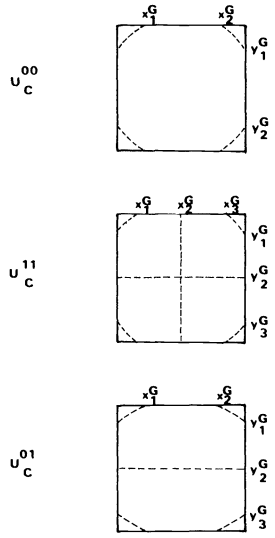


FIG. 4. *Zeros of* u_C^{00}, u_C^{11} *and* u_C^{01} *for* $k=1$.

following relationships

$$P_{lm}(x,y) = (-1)^l u_L^m(x,y) + u_R^m(x,y) + (-1)^m u_D^l(x,y) + u_U^l(x,y) + u_c^{lm}(x,y). \tag{3.33}$$

d. *Elementary mass and stiffness matrices.* Let us introduce the elementary mass and stiffness matrices M^e and K^e of order N as

$$M^e = (m_{ij}^e), \qquad K^e = (k_{ij}^e), \tag{3.34}$$

where

$$m_{ij}^e = \int_{-1}^{+1}\int_{-1}^{+1} u_i(x,y)\cdot u_j(x,y)\,dx\,dy, \qquad i,j=1,\cdots,N,$$
$$k_{ij}^e = \int_{-1}^{+1}\int_{-1}^{+1} \nabla u_i^T(x,y)\cdot \nabla u_j(x,y)\,dx\,dy, \qquad i,j=1,\cdots,N. \tag{3.35}$$

Assume now that the vector ${}^t\mathbf{u} = [u_1,\cdots,u_N]$ of basis functions is partitioned following

$$\mathbf{u} = \begin{bmatrix} \mathbf{u}_H \\ \hline \mathbf{u}_C \\ \hline \mathbf{u}_V \end{bmatrix}, \tag{3.36}$$

where

$${}^t\mathbf{u}_H = [u_L^0,\cdots,u_L^k,u_R^0,\cdots,u_R^k],$$
$${}^t\mathbf{u}_C = [u_C^{00},u_C^{10},\cdots,u_C^{kk}], \tag{3.37}$$
$${}^t\mathbf{u}_V = [u_D^0,\cdots,u_D^k,u_U^0,\cdots,u_U^k].$$

As M^e and K^e are partitioned accordingly, we have for instance

$$M^e = \begin{bmatrix} M_{HH}^e & M_{HC}^e & M_{HV}^e \\ M_{CH}^e & M_{CC}^e & M_{CV}^e \\ M_{VH}^e & M_{VC}^e & M_{VV}^e \end{bmatrix}. \tag{3.38}$$

If the matrix elements are evaluated exactly, it is easy to realize that

$$M^e_{HV} = {}^tM^e_{VH} = K^e_{HV} = {}^tK^e_{VH} = 0. \tag{3.39}$$

so that the coupling between the H (for horizontal) and V (for vertical) components is only via the cell parameters.

It is also interesting to note that the couplings between horizontal components due to mass and horizontal stiffness (corresponding to $\partial_x(p\,\partial_x \cdot)$) are *almost* only between those horizontal components belonging to a given vertical moment, for instance the unknowns ${}^t\mathbf{m}^l_H = [m^l_L;\ m^{il}_C,\ i = 0, \cdots, k;\ m^l_R]$, $l = 0, \cdots, k$, are coupled together through mass and horizontal stiffness but they are not directly coupled to ${}^t\mathbf{m}^m_H$, $m = 0, \cdots, k$ where $m \neq l$, *unless m and l have the same parity.* Thus for $k = 0$ and 1, there is no coupling at all, but for $k \geqq 2$ some coupling appears, which can in fact be eliminated by an almost consistent numerical integration of the Gauss tensor product type with $k+1$ or $k+2$ Gauss points in each direction.

The mass matrix can be greatly simplified if reduced integration is used in a somewhat nonstandard way. The keypoint here is to use a product Radau $(k+2)$ points rule corresponding to the matrix element we want to determine. Each time a boundary basis function u_B where B is L, R, D or U appears, the corresponding Radau rule should have sampling abscissae or ordinates x^{RR}, x^{RL}, y^{RU} or y^{RD} respectively. As a result the elementary mass matrix M^e only has nonzero entries in M^e_{CC}. For the elements of M^e_{CC}, we have thanks to (3.33)

$$(u^{ij}_c,\ u^{kl}_c) = (P_{ij},\ P_{kl}) = \frac{4}{(2i+1)(2j+1)}\,\delta_{ik}\delta_{jl}, \tag{3.40}$$

since the Radau rule is correct for $x^\alpha y^\beta$, $\alpha, \beta \leqq 2k+2$. The simplest elementary mass and stiffness matrices are sketched in Fig. 5, without and with reduced integration.

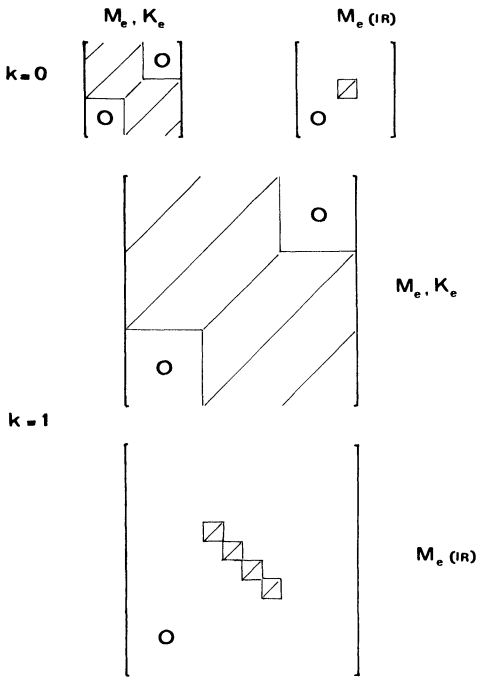


FIG. 5. *Elementary mass and stiffness matrices without and with reduced integration for* $k = 0$ *and* 1.

4. The physical nodal method. By P(hysical) N(odal) M(ethod) as opposed to the M(athematical) N(odal) M(ethod) presented in § 3, we mean a nodal method of index k using the same basis functions as above, but where the equations are derived through more physical arguments, namely:

1. *Cell balance equations.* The P_{lm}, $l, m = 0, \cdots, k$ first moments of the equation over a cell are zero:

$$\int_{K \in \tau_h} P_{lm}(x, y)[Lu_h - f]\, dr = 0, \qquad l, m = 0, \cdots, k, \tag{4.1}$$

2. *Current continuity conditions.* With the basis functions defined hereabove, the first $(k+1)$ moments of the function u are automatically continuous from one cell to its neighbors. The current continuity conditions simply ensure that the corresponding moments of $p\nabla u$ are continuous through cell interfaces:

$$\int_{\Gamma_{12} = K_1 \cap K_2} P_l(s)[p\nabla u_h|_{K_1} - p\nabla u_h|_{K_2}] \cdot 1_n\, ds = 0, \qquad l = 0, \cdots, k, \tag{4.2}$$

where 1_n is some (arbitrary) unit normal to Γ_{12}.

The following general theorem can be proved.

THEOREM. *Assuming that p and q are cellwise constant, the* PNM *is equivalent to the* MNM *with Radau reduced integration.*

Proof. The MNM consists of finding $u_h \in V_h$ such that

$$\forall v_h \in V_h, \qquad a_h(u_h, v_h) = f(v_h), \tag{4.3}$$

where

$$a_h(u_h, v_h) = \sum_{K \in \tau_h} \int_K (p\nabla u_h^T \cdot \nabla v_h + q u_h v_h)\, dr, \tag{4.4}$$

and

$$f(v_h) = \int_\Omega f v_h\, dr. \tag{4.5}$$

a. Let us take as v_h the cell basis functions u_C^{ij}, $i, j = 0, \cdots, k$ associated to some element K. Equation (4.3) becomes

$$\int_K (p\nabla u_h^T \cdot \nabla u_C^{ij} + q u_h u_C^{ij} - f u_C^{ij})\, dr = 0, \qquad i, j = 0, \cdots, k, \tag{4.6}$$

or equivalently

$$\int_{\partial K} u_C^{ij}(p\nabla u_h \cdot 1_n)\, ds + \int_K u_C^{ij}\{-\nabla \cdot p\nabla u_h + q u_h - f\}\, dr = 0, \qquad i, j = 0, \cdots, k. \tag{4.7}$$

Let us first show that the boundary term $\int_{\partial K}$ —— is zero: with (3.33), it is easy to rewrite $u_h \in V_h$ as

$$\begin{aligned} u_h = &\sum_{l=0}^{k} \sum_{m=0}^{k} m_C^{lm} P_{lm}(x, y) \\ &+ \sum_{l=0}^{k} [m_L^{l*} u_L^l(x, y) + m_R^{l*} u_R^l(x, y) + m_D^{l*} u_D^l(x, y) + m_U^{l*} u_U^l(x, y)], \end{aligned} \tag{4.8}$$

so that the contribution of the P_{lm} terms to the x and y components of ∇u_h is proportional to $P_m(y)$ and $P_l(x)$ respectively. With p a constant over a given cell, the corresponding boundary terms are zero as u_C^{ij} is orthogonal on the boundary to $P_m(y)$ and $P_l(x)$, $m, l = 0, \cdots, k$ (see § 3.2b).

The contributions of the boundary basis functions to ∇u_h are easily seen to be proportional to $P_l(x \text{ or } y)$, $l = 0, \cdots, k$ or to $P_{k+1} \pm P_{k+2}(x \text{ or } y)$. In the first case, the corresponding boundary terms are zero for the same reason as above while in the second case, they disappear if Radau reduced integration is used. Equation (4.7) under Radau reduced integration finally becomes

$$\int_K P_{ij}\{-\nabla \cdot p\nabla u_h + qu_h - f\}\, dr = 0, \qquad i,j = 0, \cdots, k, \tag{4.9}$$

and is therefore equivalent to the cell balance equations (4.1).

b. Let us now take as v_h the boundary basis functions u_B^i, $i = 0, \cdots, k$ where $B = L, R, B$ or T. For the sake of simplicity, let us consider u_B^i reducing to u_R^i on K_1 and to u_L^i on K_2 with $\Gamma_{12} = K_1 \cap K_2$ as shown in Fig. 6. For this particular example, (4.3) becomes

$$\begin{aligned} &\int_{K_1} (p\nabla u_h^T \cdot \nabla u_R^i + qu_h u_R^i - fu_R^i)\, dr \\ &\quad + \int_{K_2} (p\nabla u_h^T \cdot \nabla u_L^i + qu_h u_L^i - fu_L^i)\, dr = 0, \qquad i = 0, \cdots, k, \end{aligned} \tag{4.10}$$

or equivalently (after Radau reduced integration)

$$\int_{\partial K_1} u_R^i p\nabla u_h \cdot 1_{n_1} ds + \int_{\partial K_2} u_L^i p\nabla u_h \cdot 1_{n_2} ds = 0, \qquad i = 0, \cdots, k. \tag{4.11}$$

Since u^i is either zero (u_R^i on L_1, u_L^i on R_2) or because of Radau reduced integration on D_1, U_1, D_2 and U_2, (4.11) reduces to

$$\int_{\Gamma_{12}} P_i(y)[p\nabla u_h|_{K_1} - p\nabla u_h|_{K_2}] \cdot 1_n\, dy = 0, \qquad i = 0, \cdots, k, \tag{4.12}$$

and is therefore equivalent to the current continuity conditions on Γ_{12}. QED.

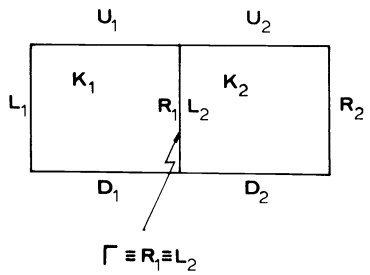


FIG. 6. *Two adjacent nodes K_1 and K_2.*

5. Dimensionally reduced nodal methods. By D(imensionally) R(educed) N(odal) M(ethods), as opposed to the F(ully) D(imensional) N(odal) M(ethods) presented in §§ 3 and 4, we mean mathematical or physical nodal methods of index k with the same basis functions over a given cell as above, but where the equations are derived

in such a way that an efficient solution of the global system of algebraic equations is directly suggested.

The basic idea here is to proceed to some transverse averaging of the given equation. In 2D, we shall follow the notation of Fig. 7. The original equation

$$-\nabla \cdot p\nabla u + qu = f \qquad \forall r \in \Omega, \tag{5.1}$$

is successively multiplied by $p_{jl}(y)$, $l=0,\cdots,k$ and integrated in the y direction between y_{Dj} and y_{Uj}, $j=1,\cdots,J$, $p_{jl}(y)$ being the modified lth Legendre polynomial

$$p_{jl}(y) = P_l\left[\frac{2(y-y_j)}{y_{Uj}-y_{Dj}}\right]. \tag{5.2}$$

The same is done in the x direction between x_{Li} and x_{Ri}, $i=1,\cdots,I$ after multiplication by $p_{il}(x)$, $l=0,\cdots,k$. Let us introduce the following transverse moments of u:

$$u^l_{xj}(x) = \frac{\int_{y_{Dj}}^{y_{Uj}} p_{jl}(y)u(x,y)\,dy}{\int_{y_{Dj}}^{y_{Uj}} p^2_{jl}(y)\,dy}, \qquad j=1,\cdots,J, \quad l=0,\cdots,k, \tag{5.3a}$$

and

$$u^l_{iy}(y) = \frac{\int_{x_{Li}}^{x_{Ri}} p_{il}(x)u(x,y)\,dx}{\int_{x_{Li}}^{x_{Ri}} p^2_{il}(x)\,dx}, \qquad i=1,\cdots,I, \quad l=0,\cdots k. \tag{5.3b}$$

The result of the transverse averaging procedure is a set of $2(k+1)$ one-dimensional equations for the $2(k+1)$ moments u^l_{xj} and u^l_{iy}, namely

$$-p\frac{d^2u^l_{xj}(x)}{dx^2} + qu^l_{xj}(x) = \hat{f}^l_{xj}(x), \qquad l=0,\cdots,k, \tag{5.4a}$$

$$-p\frac{d^2u^l_{iy}(y)}{dy^2} + qu^l_{iy}(y) = \hat{f}^l_{iy}(y), \qquad l=0,\cdots,k, \tag{5.4b}$$

where we have assumed p and q constant for the sake of simplicity. In (5.4a), $\hat{f}^l_{xj}(x)$ for instance is an effective source term including the lth 1D moment of f, $f^l_{xj}(x)$ defined analogously to $u^l_{xj}(x)$, as well as a transverse leakage term, $l^l_{xj}(x)$, arising from the cell boundary values of the y integral of the differential operator with respect to y. For

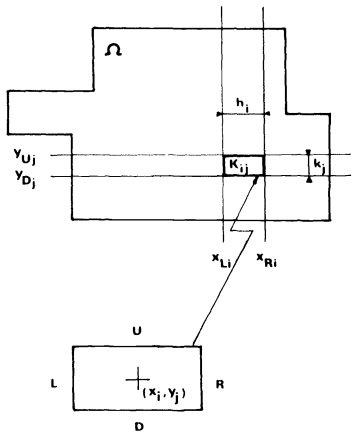


FIG. 7. *Dimensionally reduced nodal methods.*

instance

$$\hat{f}^0_{xj}(x) = f^0_{xj}(x) + (1/N^0_{xj})\{k_j(-q_{yU}(x) + q_{yD}(x))\},$$
$$\hat{f}^1_{xj}(x) = f^1_{xj}(x) + (1/N^1_{xj})\{-p(u_U(x) - u_D(x)) - k_j(q_{yU}(x) + q_{yD}(x))/2\}$$

and

$$f^2_{xj}(x) = f^2_{xj}(x) + (1/N^2_{xj})\{k_j(-q_{yU}(x) + q_{yD}(x))/2 - 3p(u_U(x) + u_D(x)) + 6pu^0_{xj}(x)\}, \tag{5.5}$$

where

$$N^l_{xj} = \frac{k_j}{2l+1}, \qquad k_j = y_{Uj} - y_{Dj},$$

$$q \equiv -p\nabla u = q_x 1_x + q_y 1_y,$$

and

$$u_U(x) = u(x, y_{Uj}), \qquad q_{yU}(x) = q_y(x, y_{Uj}), \text{ etc.} \tag{5.6}$$

Clearly over the cell $K_{ij} \equiv [x_{Li}, x_{Ri}] \times [y_{Dj}, y_{Uj}]$, we must have the compatibility conditions

$$\frac{1}{N^l_{iy}} \int_{x_{Li}}^{x_{Ri}} p_{il}(x) u^m_{xj}(x)\, dx = \frac{1}{N^m_{xj}} \int_{y_{Dj}}^{y_{Uj}} p_{jm}(y) u^l_{iy}(y)\, dy \equiv m^{lm}_C, \qquad l, m = 0, \cdots, k. \tag{5.7}$$

In fact, these conditions are automatically satisfied as soon as a given unique representation of $u_h(x, y)$ over each cell is looked for as in our case. Let $u_h(x, y)$ be given as in (3.2)

$$u_h(x, y) = \sum_{l=0}^{k} (m^l_{Lij} u^l_{Lij}(x, y) + m^l_{Rij} u^l_{Rij}(x, y) + m^l_{D_{ij}} u^l_{Dij}(x, y) + m^l_{Uij} u^l_{Uij}(x, y)) + \sum_{l=0}^{k} \sum_{m=0}^{k} m^{lm}_{Cij} u^{lm}_{Cij}(x, y) \quad \forall (x, y) \in K_{ij}, \tag{5.8}$$

where the basis functions and the corresponding moments are now defined with respect to K_{ij}, so that for instance

$$u^l_{Lij}(x, y) = u^l_L\left(\frac{2(x - x_i)}{h_i}, \frac{2(y - y_j)}{k_j}\right),$$
$$m^l_{Lij} = \frac{1}{N^l_{xj}} \int_{y_{Dj}}^{y_{Uj}} P_{jl}(y) u_h(x_{Li}, y)\, dy \equiv u^l_{xj}(x_{Li}), \tag{5.9}$$

$$u^l_{xj}(x)|_{K_{ij}} = m^l_{Li} u^l_{Li}(x) + m^l_{Ri} u^l_{Ri}(x) + \sum_{m=0}^{k} m^{ml}_{Ci} u^{ml}_{Ci}(x), \tag{5.10}$$

where

$$m^l_{Li} \equiv m^l_{Lij}, \qquad m^l_{Ri} \equiv m^l_{Rij}, \qquad m^l_{Ci} \equiv m^l_{Cij},$$

etc. Therefore it is easy to see that the restriction $u^l_{xj}(x)|_{K_{ij}}$ of $u^l_{xj}(x)$ to K_{ij} is given by

$$u_{Li}(x) \equiv \frac{1}{N^l_{xj}} \int_{y_{Dj}}^{y_{Uj}} p_{lj}(y) u^l_{Lij}(x, y)\, dy,$$

$$u_{Ri}(x) \equiv \frac{1}{N^l_{xj}} \int_{y_{Dj}}^{y_{Uj}} p_{lj}(y) u^l_{Rij}(x, y)\, dy,$$

while

$$u_{Ci}^{m}(x) \equiv \frac{1}{N_{xj}^{l}} \int_{y_{Dj}}^{y_{Uj}} p_{ij}(y) u_{Cij}^{ml}(x, y)\, dy, \qquad m = 0, \cdots, k. \tag{5.11}$$

After a standard mapping to the reference element $[-1, +1]$, the above basis functions are clearly the nodal basis functions in 1D:

$$\begin{aligned} u_{Li}^{l}\left(\frac{x - x_{Li}}{2(x_{Ri} - x_{Li})}\right) &\equiv u_L(x), \\ u_{Ri}^{l}\left(\frac{x - x_{Li}}{2(x_{Ri} - x_{Li})}\right) &\equiv u_R(x), \\ u_{Ci}^{ml}\left(\frac{x - x_{Li}}{2(x_{Ri} - x_{Li})}\right) &\equiv u_C^m(x), \qquad m = 0, \cdots, k, \end{aligned} \tag{5.12}$$

where we have dropped the lth moment superindex as the 1D basis functions are the same for any lth moment, $l = 0, \cdots, k$. Expressions similar to (5.10) are clearly valid for the restriction $u_{iy}^{l}(y)|_{K_{ij}}$ and we realize that the restriction to K_{ij} of each transverse moments depends on $(k+3)$ parameters which are its values at the end points and its first $(k+1)$ moments. The $(k+1)^2$ moments over the cell ensure the 2D coupling while the values at the end points only belong to vertical or horizontal transverse moments and realize the coupling in the other (i.e. horizontal or vertical) direction.

A numerical solution of each of the equations (5.4) is quite naturally obtained by a standard Galerkin approach using the same set of parameters as above, namely end-points values and in-cell moments: in other words, a nodal method in 1D is used, in either the mathematical version with consistent integration or the physical version with reduced integration. Since the end points values are common to adjacent elements, the corresponding approximation in 1D will now be conforming ($V_h \in H_0^1(\bar{\Omega})$).

In any case, the final scheme is an iterative one: at iteration (n), the x one-dimensional equations (5.4a) are solved assuming that the transverse leakage terms in the y direction are known from iteration $(n-1)$. The same is then done for the y one-dimensional equations (5.4b) assuming that the transverse leakage terms in the x direction are known from iteration (n) and so on. The corresponding iteration will eventually converge and several acceleration procedures are available for these alternating direction like methods. We shall not pursue in that direction and leave those details for a subsequent paper on the practical implementation of these nodal schemes.

Before leaving this section, we would like to make some comments with respect to the numerical solution of equations (5.4): first of all, the x or the y one-dimensional equations are all similar since only their second members change, which means that increasing the order k is not very expensive, since assembly and factorization are done once for all. Moreover, we are not limited to polynomial nodal methods and analytical nodal methods can easily be implemented in 1D as in [20], especially if the coefficients p and q are piecewise constants. In this case, the general solution of any of the equations (5.4) over a given interval I is a linear combination of $\cosh x/L$ and $\sinh x/L$, where $L \equiv (p/q)^{1/2}$ is the characteristic length known as the diffusion length by the nuclear engineers, plus a particular solution of the given equation: if $f(x)$ is a generic second member, this particular solution is given by $\int_I G(x|x_0) f(x)\, dx$ where $G(x|x_0)$ is the one-dimensional Green's function. In practice, $f(x)$ is truncated to the first few terms of its Taylor expansion and the corresponding general solution is therefore correct only for second members which are polynomials up to some degree: this method is

closely related to the $\mathscr{L}_1$-finite element method of [21] where a detailed analysis is given. Finally, we should say that in n-dimensions the transverse integration procedure can still be used leading to sets of one-dimensional differential equation, provided the transverse moments are taken over all the variables minus one.

6. Conclusions. In this section, we would like to summarize some of the results we have obtained. Starting from early nodal methods of the Σ type [5], [6], we developed in § 3 a general family of nodal methods, which are not affected by the limitations inherent to the early nodal schemes: in other words, the convergence order in L_2 norm is of $O(h^{k+2})$ for the nodal scheme of index or type (k), while the early nodal schemes had convergence orders of $O(h^2)$ for any k. A version of these schemes based on more physical considerations was presented in § 4 and was related to the original (or mathematical) nodal schemes of § 3 through the use of properly chosen quadrature rules. In § 5, finally, dimensionally reduced versions were introduced which directly lead to iterative strategies of alternating direction type.

In Fig. 8, we sketch the spaces P of index $k=0, 1, 2$, and show their relationship with the Raviart-Thomas spaces [22] used in the mixed finite element approximation of the given elliptic problem. Corresponding to $P \equiv \mathscr{Q}_{k+2,k} \cap \mathscr{Q}_{k,k+2}$, Raviart and Thomas pick up q in $V = V_1 \times V_2 \equiv \mathscr{Q}_{k+1,k} \times \mathscr{Q}_{k,k+1}$ and u in $W \equiv \mathscr{Q}_{k,k}$, and they show that

$$\text{div } V = W. \tag{6.1}$$

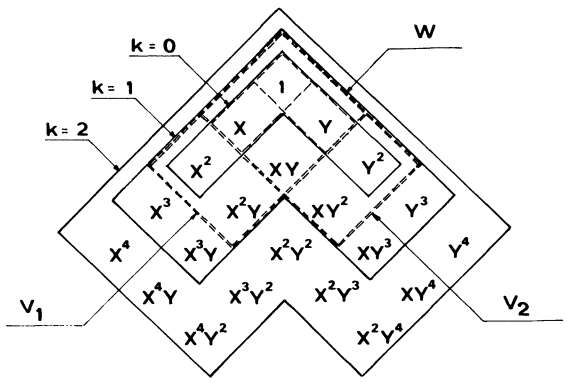


FIG. 8. *Relationship between nodal spaces of index* $k=0, 1,$ *and* 2 *and the lowest order Raviart-Thomas spaces* ($k=0$ – – – – *and* $k=1$ = = = =). V_1, V_2 *and* W (*see text*) *are shown for* $k=1$.

In the particular case, $k=0$, if u is taken in $\mathscr{Q}_{2,0} \cup \mathscr{Q}_{0,2}$, clearly q is in V and the relationship of the corresponding nodal scheme with a mixed-hybrid finite element one can be shown [23]. In the general case, $k>0$, q will be in $(\mathscr{Q}_{k+1,k} \cup \mathscr{Q}_{k-1,k+2}) \times (\mathscr{Q}_{k+2,k-1} \cup \mathscr{Q}_{k,k+1}) \supset V$ and the relationship with mixed-hybrid finite elements is less clear, although the dimensionally-reduced forms seem closely related as the L^2 projection provided by the transverse integration forces q in V again. In a future paper, we plan to further explore this interesting relationship.

Note. In a recent work [24], it is shown how the general nodal schemes proposed here do in fact provide a quite natural enhancement of the Raviart-Thomas mixed finite element approximations, when a mixed-hybrid implementation is proposed as in [25]. By "enhancement," it is meant that one order of convergence is gained by a post-processing operation performed element by element, which turns out to be trivial if the parameters chosen are the edge and cell moments used hereabove.

Appendix A. The general case (k, l) in two dimensions. Assuming that the maximal degrees in x and y are $k+2$ and $l+2$ respectively with $k, l \geqq 0$, the set Σ of basic parameters will be

(A1) $\Sigma \equiv \{m_D^i, m_U^i, i=0, \cdots, k; m_L^j, m_R^j, j=0, \cdots, l; m_c^{ij}, i=0, \cdots, k; j=0, \cdots, l\}$,

while the corresponding basis functions will belong to the space

(A2) $$P \equiv \mathcal{Q}_{k+2,l} \cup \mathcal{Q}_{k,l+2}.$$

Clearly

(A3) $$\operatorname{card} \Sigma = \dim P = (k+1)(l+1)+2(k+l+2).$$

With the same kind of arguments as in § 3.1, one easily obtains the following expressions for the basis functions

$$u_D^i(x, y) = \tfrac{1}{2}(-1)^{l+1} P_i(x)(P_{l+1}(y) - P_{l+2}(y)), \qquad i=0, \cdots, k,$$
$$u_U^i(x, y) = \tfrac{1}{2} P_i(x)(P_{l+1}(y) + P_{l+2}(y)), \qquad i=0, \cdots, k,$$
$$u_L^j(x, y) = \tfrac{1}{2}(-1)^{k+1}(P_{k+1}(x) - P_{k+2}(x)) P_j(y), \qquad j=0, \cdots, l,$$
$$u_R^j(x, y) = \tfrac{1}{2}(P_{k+1}(x) + P_{k+2}(x)) P_j(y), \qquad j=0, \cdots, l,$$

and

(A4) $$u_C^{ij}(x.y) = P_i(x)P_j(y) - P_{k+m(i)}(x)P_j(y) - P_i(x)P_{l+m(j)}(y), \qquad i=0, \cdots, k, \quad j=0, \cdots, l,$$

where $m(i)$ and $m(j)$ have the same meaning as in (3.24).

Appendix B. Some particular examples.

a. $\underline{k=l=0}$ (the "quadratic" nodal scheme [5]).

$$u_L^0(x, y) = -\tfrac{1}{2}(P_1(x) - P_2(x)),$$
$$u_R^0(x, y) = +\tfrac{1}{2}(P_1(x) + P_2(x)),$$
$$u_D^0(x, y) = u_L^0(y, x),$$
$$u_U^0(x, y) = u_R^0(y, x),$$
$$u_C^{00}(x, y) = 1 - P_2(x) - P_2(y).$$

b. $\underline{k=l=1}$

$$u_L^0(x, y) = \tfrac{1}{2}(P_2(x) - P_3(x)),$$
$$u_R^0(x, y) = \tfrac{1}{2}(P_2(x) + P_3(x)),$$
$$u_D^0(x, y) = u_L^0(y, x),$$
$$u_U^0(x, y) = u_R^0(y, x),$$
$$u_L^1(x, y) = \tfrac{1}{2}(P_2(x) - P_3(x)) P_1(y),$$
$$u_R^1(x, y) = \tfrac{1}{2}(P_2(x) + P_3(x)) P_1(y),$$
$$u_D^1(x, y) = u_L^1(y, x),$$
$$u_U^1(x, y) = u_R^1(y, x),$$
$$u_C^{00}(x, y) = 1 - P_2(x) - P_2(y),$$

$$u_C^{10}(x, y) = P_1(x) - P_3(x) - P_1(x)P_2(y),$$
$$u_C^{01}(x, y) = P_1(y) - P_3(y) - P_2(x)P_1(y),$$
$$u_C^{11}(x, y) = P_1(x)P_1(y) - P_3(x)P_1(y) - P_1(x)P_3(y).$$

c. $\underline{k=2, l=0}$.

$$u_L^0(x, y) = -\tfrac{1}{2}(P_3(x) - P_4(x)),$$
$$u_R^0(x, y) = \tfrac{1}{2}(P_3(x) + P_4(x)),$$
$$u_D^0(x, y) = -\tfrac{1}{2}(P_1(y) - P_2(y)),$$
$$u_U^0(x, y) = \tfrac{1}{2}(P_1(y) + P_2(y)),$$
$$u_D^1(x, y) = -\tfrac{1}{2}P_1(x)(P_1(y) - P_2(y)),$$
$$u_U^1(x, y) = \tfrac{1}{2}P_1(x)(P_1(y) + P_2(y)),$$
$$u_D^2(x, y) = -\tfrac{1}{2}P_2(x)(P_1(y) - P_2(y)),$$
$$u_U^2(x, y) = \tfrac{1}{2}P_2(x)(P_1(y) + P_2(y)),$$
$$u_C^{00}(x, y) = 1 - P_4(x) - P_2(y),$$
$$u_C^{10}(x, y) = P_1(x) - P_3(x) - P_1(x)P_2(y),$$
$$u_C^{20}(x, y) = P_2(x) - P_4(x) - P_2(x)P_2(y).$$

Appendix C. Summary of the nodal basis functions in $\mathbb{R}^n$, $n = 1, 2$ and 3.

a. *Nodal basis functions in* 1D.

$$\Sigma \equiv \{m_L, m_R, m_c^i, i = 0, \cdots, l\},$$
$$P \equiv \mathcal{P}_{l+2},$$
$$\text{card } \Sigma = \dim P = l + 3,$$
$$u_L(x) = \tfrac{1}{2}(-1)^{l+1}[P_{l+1}(x) - P_{l+2}(x)],$$
$$u_R(x) = \tfrac{1}{2}[P_{l+1}(x) + P_{l+2}(x)],$$
$$u_c^i(x) = P_i(x) - P_{l+m(i)}(x), \qquad i = 0, \cdots, l,$$

where $m(i) = 1$ or 2 and such that i and $l + m(i)$ have the same parity.

b. *Nodal basis functions in* 2D.

$$\Sigma \equiv \{m_L^i, m_R^i, m_D^i, m_U^i, i = 0, \cdots, l; m_c^{ij}, i, j = 0, \cdots, l\},$$
$$P \equiv \mathcal{Q}_{l+2,l} \cup \mathcal{Q}_{l,l+2}, \text{ card } \Sigma = \dim P = (l+1)(l+5),$$
$$u_L^i(x, y) = \tfrac{1}{2}(-1)^{l+1}[P_{l+1}(x) - P_{l+2}(x)]P_i(y), \qquad i = 0, \cdots, l,$$
$$u_R^i(x, y) = \tfrac{1}{2}[P_{l+1}(x) + P_{l+2}(x)]P_i(y), \qquad i = 0, \cdots, l,$$
$$u_D^i(x, y) = \tfrac{1}{2}(-1)^{l+1}P_i(x)[P_{l+1}(y) - P_{l+2}(y)], \qquad i = 0, \cdots, l,$$
$$u_U^i(x, y) = \tfrac{1}{2}P_i(x)[P_{l+1}(y) + P_{l+2}(y)], \qquad i = 0, \cdots, l,$$
$$u_c^{ij}(x, y) = P_i(x)P_j(y) - P_{l+m(i)}(x)P_j(y) - P_i(x)P_{l+m(j)}(y), \qquad i, j = 0, \cdots, l$$

where $m(i)$ (resp. $m(j)$) $= 1$ or 2 and such that i and $l + m(i)$ (resp. j and $l + m(j)$) have the same parity.

c. *Nodal basis functions in* 3D.

$$\Sigma \equiv \{m_L^{ij}, m_R^{ij}, m_D^{ij}, m_U^{ij}, m_F^{ij}, m_B^{ij}, i,j=0,\cdots,l;\ m_C^{ijk}, i,j,k,=0,\cdots,l\},$$

$$P \equiv \mathcal{Q}_{l+2,l,l} \cup \mathcal{Q}_{l,l+2,l} \cup \mathcal{Q}_{l,l,l+2},$$

$$\text{card}\,\Sigma = \dim P = (l+1)^2(l+7),$$

$$u_L^{ij}(x,y,z) = \tfrac{1}{2}(-1)^{l+1}[P_{l+1}(x) - P_{l+2}(x)]P_i(y)P_j(z), \qquad i,j=0,\cdots,l,$$

$$u_R^{ij}(x,y,z) = \tfrac{1}{2}[P_{l+1}(x) + P_{l+2}(x)]P_i(y)P_j(z), \qquad i,j=0,\cdots,l,$$

$$u_D^{ij}(x,y,z) = \tfrac{1}{2}(-1)^{l+1}P_i(x)[P_{l+1}(y) - P_{l+2}(y)]P_j(z), \qquad i,j=0,\cdots,l,$$

$$u_U^{ij}(x,y,z) = \tfrac{1}{2}P_i(x)[P_{l+1}(y) + P_{l+2}(y)]P_j(z), \qquad i,j=0,\cdots,l,$$

$$u_F^{ij}(x,y,z) = \tfrac{1}{2}(-1)^{l+1}P_i(x)P_j(y)[P_{l+1}(z) - P_{l+2}(z)], \qquad i,j=0,\cdots,l,$$

$$u_B^{ij}(x,y,z) = \tfrac{1}{2}P_i(x)P_j(y)[P_{l+1}(z) + P_{l+2}(z)], \qquad i,j=0,\cdots,l,$$

$$u_C^{ijk}(x,y,z) = P_i(z)P_j(y)P_k(z) - P_{l+m(i)}(x)P_j(y)P_k(z) - P_i(x)P_{l+m(j)}(y)P_k(z)$$
$$- P_i(x)P_j(y)P_{l+m(k)}(z), \qquad i,j,k=0,\cdots,l$$

where $m(i)$, $m(j)$ and $m(k)$ have the same meaning as above.

REFERENCES

[1] J. J. DORNING, *Modern coarse-mesh methods. A development of the '70's* in Computational Methods in Nuclear Engineering, Vol. 1, Williamsburg, Virginia, 1979, pp. 3-1-3-31.

[2] A. F. HENRY, *Nuclear-Reactor Analysis*, MIT Press, Cambridge, MA, 1975.

[3] J. R. FANCHI, K. J. HARPOLE AND S. W. BUJNOWSKI, BOAST: *A Three-dimensional, three-phase black oil applied simulation tool (Version* 1.1), Report DOE/BC/10033-3, U.S. Dept of Energy, Washington, DC, 1982.

[4] C. FEDON-MAGNAUD, J. P. HENNART AND J. J. LAUTARD, *On the relationship between some nodal schemes and the finite element method in static diffusion calculations*, in Advances in Reactor Computations, Vol. 2, Salt Lake City, UT, 1983, pp. 987-1000.

[5] S. LANGENBUCH, W. MAURER AND W. WERNER, *Coarse-mesh flux-expansion method for the analysis of space-time effects in large light water reactor cores*, Nucl. Sci. Engng., 63 (1977), pp. 437-456.

[6] ———, *High-order schemes for neutron kinetics calculations, based on a local polynomial approximation*, Nucl. Sci. Engng, 64 (1977), pp. 508-516.

[7] E. L. WACHSPRESS, *Iterative Solution of Elliptic Systems and Applications to the Neutron Diffusion Equations of Reactor Physics*, Prentice-Hall, Englewood Cliffs, NJ, 1966.

[8] G. STRANG AND G. J. FIX, *An Analysis of the Finite Element Method*, Prentice-Hall, Englewood Cliffs, NJ, 1973.

[9] I. GLADWELL AND R. WAIT, eds., *A Survey of Numerical Methods for Partial Differential Equations*, Clarendon Press, Oxford, 1979.

[10] P. G. CIARLET, *The Finite Element Method for Elliptic Problems*, North-Holland, Amsterdam, 1978.

[11] C. FEDON-MAGNAUD, *Etude théorique de quelques méthodes nodales de résolution de l'équation de diffusion—Tests numériques*, Thesis, Univ. Paris VI, 1983.

[12] I. BABUŠKA AND R. B. KELLOGG, *Numerical solution of the neutron diffusion equation in the presence of corners and interfaces*, in Numerical Reactor Calculations, International Atomic Energy Agency, Vienna, 1972, pp. 473-486.

[13] J. P. HENNART AND E. H. MUND, *Singularities in the finite element approximation of two-dimensional diffusion problems*, Nucl. Sci. Engng, 62 (1977), pp. 55-68.

[14] O. C. ZIENKIEWICZ, *The Finite Element Method*, third edition, McGraw-Hill, London, 1977.

[15] J. P. HENNART, *A general finite element framework for nodal schemes*, to appear in MAFELAP V, The Mathematics of Finite Elements and Applications, J. R. Whiteman, ed., Academic Press, London.

[16] J. P. HENNART, E. SAINZ AND M. VILLEGAS, *On the efficient use of the finite element method in static neutron diffusion calculations*, in Computational Methods in Nuclear Engineering, Vol. 1, Williamsburg, VA, 1979, pp. 3-87-3-103.

[17] J. J. LAUTARD, *New finite element representation for* 3D *reactor calculations*, in Advances in Mathematical Methods for the Solution of Nuclear Engineering Problems, Vol. 1, Munich, Federal Republic of Germany, 1981, pp. 349–366.

[18] A. KAVENOKY AND J. J. LAUTARD, *State of the art in using finite element method for neutron diffusion calculation* in Advances in Reactor Computations, Vol. 2, Salt Lake City, UT 1983, pp. 963–986.

[19] J. P. HENNART AND E. MUND, in preparation.

[20] Y. Y. AZMY AND J. J. DORNING, *A nodal integral approach to the numerical solution of partial differential equations* in Advances in Reactor Computations, Vol. 2, Salt Lake City, 1983, pp. 893–909.

[21] I. BABUŠKA AND J. E. OSBORN, *Generalized finite element methods: Their performance and their relation to mixed methods*, SIAM J. Numer. Anal., 20 (1983), pp. 510–536.

[22] P. A. RAVIART AND J. M. THOMAS, *A mixed finite element method for second order elliptic problems* in Lecture Notes in Mathematics 606, 1977, Springer-Verlag, Heidelberg, pp. 292–315.

[23] M. FORTIN, personal communication, 1983. M. Fortin also knew independently how to pick up the higher degree spaces P.

[24] J. P. HENNART, *Nodal schemes, mixed-hybrid finite elements and block-centered finite differences*, in preparation.

[25] D. N. ARNOLD AND F. BREZZI, *Mixed and nonconforming finite methods: implementation, post-processing and error estimates*, RAIRO, to appear.

SIAM J. SCI. STAT. COMPUT.
Vol. 7, No. 1, January, 1986

019

A FINITE DIFFERENCE SCHEME FOR THE EQUILIBRIUM EQUATIONS OF ELASTIC BODIES*

T. N. PHILLIPS† AND M. E. ROSE†

Abstract. A compact difference scheme is described for treating the first order system of PDE's which describe the equilibrium equations of an elastic body. An algebraic simplification enables the solution to be obtained by standard direct or iterative techniques.

Key words. elasticity, finite difference, iterative methods

Introduction. The conditions for the static equilibrium of an elastic body are described by an elliptic system of partial differential equations for the displacements and stresses. This paper describes a finite difference scheme which can be solved by standard direct or iterative methods and yields a solution which approximates smooth displacements with second-order accuracy. Iterative techniques can be attractive as a means of solving three-dimensional problems because they minimize storage requirements and present an algorithmic structure well-suited to advanced computer architectures. For material problems, these features can be useful for solving layered composite materials as well as materials with nonlinear properties.

As described here, a serious limitation of this method vis-a-vis finite element methods is that it is applicable only to bodies which can be subdivided into cube-like volume cells. However, a means of removing this restriction will be described in another paper.

Part I describes an algebraic approximation to the equilibrium conditions in a cell as expressed by tractions and displacements on the surface of the cell. The condition that traction forces balance across cell faces leads to an algebraic condition for equilibrium between any neighboring cells expressed solely in terms of displacements. A finite-sum approximation to the work due to tractions leads to an energy estimate and to a variational description of the algebraic equilibrium equations.

Part II illustrates this development for an isotropic material using a plane stress assumption to reduce the problem to two dimensions. Several simple iteration schemes are used to investigate the numerical convergence of the method when a singularity is present.

The methods described in this paper are closely related to those described by the authors in the context of a simpler problem [1].

I. General development.

I.1. The equilibrium problem. In this section we describe the equilibrium equations for an elastic body and attempt to motivate the origin of the finite difference scheme which will be described in the following section.

We consider a material body occupying a domain Ω on whose boundary, Γ, $\mathbf{n}$ is an outward unit normal; $\mathbf{u} = (u_1, u_2, u_3)^T$ denotes the displacement vector, $\tau = (\tau_1, \tau_2, \tau_3)$ the symmetric stress tensor, and $\varepsilon = (\varepsilon_1, \varepsilon_2, \varepsilon_3)$ the symmetric strain tensor.

* Received by the editors February 28, 1984, and in final revised form January 22, 1985. Research was supported by the National Aeronautics and Space Administration under NASA contract no. NAS1-17070 while the authors were in residence at ICASE, NASA Langley Research Center, Hampton, Virginia 23665.

† Institute for Computer Applications in Science and Engineering, NASA Langley Research Center, Hampton, Virginia 23665.

In the absence of body forces, the equations of equilibrium are described by the following system of first-order partial differential equations:

$$
\begin{aligned}
&\text{(a)} && \partial_{x_1}\tau_1+\partial_{x_2}\tau_2+\partial_{x_3}\tau_3=0,\\
(1)\quad &\text{(b)} && \varepsilon(\mathbf{u})=(\operatorname{grad}(\mathbf{u}+\mathbf{u}^T))/2 \quad \text{in } \Omega,\\
&\text{(c)} && \tau_i(\mathbf{u})=\sum_{j=1}^{3} c_{ij}\varepsilon_j(\mathbf{u}), \qquad i=1,2,3.
\end{aligned}
$$

Here (a) states the conditions for the equilibrium of forces, (b) defines the strain tensor in terms of the displacement, and (c) is the constitutive relationship between stress and strain (Hooke's law). In (c), (c_{ij}) involves 21 parameters. By assumption

$$\varepsilon^T\tau=\sum \varepsilon_{ij}\tau_{ij}\geqq 0 \tag{2}$$

with equality holding iff $\varepsilon=0$.

On the boundary surface Γ, we let $\mathbf{p}\equiv\tau\cdot\mathbf{n}$ denote the traction due to the stress. Then boundary conditions associated with (1) are

$$
\begin{aligned}
&\mathbf{u}=\mathbf{u}^0 \quad \text{on } \Gamma_1,\\
&\mathbf{p}=\mathbf{p}^0 \quad \text{on } \Gamma_2,
\end{aligned}
\tag{3}
$$

where Γ_1 and Γ_2 form a disjoint partition of Γ; if $\Gamma_1=\phi$ then the solution of (1) will be determined to within a rigid body displacement.

As a result of solving this boundary value problem, the tractions $\mathbf{p}$ will be determined on Γ_1 and the displacements $\mathbf{u}$ on Γ_2. Thus the tractions on Γ, say $\mathbf{p}(\Gamma)$, will be determined by the displacements $\mathbf{u}(\Gamma)$ on Γ, a fact we indicate symbolically by

$$\mathbf{p}(\Gamma)=R_\Omega\mathbf{u}(\Gamma). \tag{4}$$

We call the boundary operator R_Ω a transmission operator.

Let $\Pi(\Omega)\equiv\{\omega\}$ denote a partition of Ω into volume cells. Standard integration arguments show that $\mathbf{u}$ must be continuous and the surface tractions must balance across cell faces. Clearly, a necessary and sufficient condition for equilibrium in Ω is that any individual cell be in equilibrium with any neighboring cell.

Consider a cell ω whose volume is $O(h^3)$, where h is a representative length scale, and whose boundary surface γ consists of m faces. In equilibrium, the tractions $\mathbf{p}(\gamma)$ on γ will be related to the displacements $\mathbf{u}(\gamma)$ on γ by means of a transmission operator R_ω, i.e., $\mathbf{p}(\gamma)=R_\omega\mathbf{u}(\gamma)$. Let $\langle\phi\rangle_\gamma$ indicate a vector whose m components represent the average values of ϕ on the faces of ω. We may then relate the average tractions $\langle\mathbf{p}\rangle_\gamma$ on the faces of ω to the average displacements on the faces by

$$\langle\mathbf{p}\rangle_\gamma=R_\omega^h\langle\mathbf{u}\rangle_\gamma, \tag{5}$$

where R_ω^h is an $m\times m$ transmission matrix which is related to R_ω. When conditions for the balance of average surface tractions across cell faces are adjoined to (5), as well as boundary conditions for average tractions and displacements on Γ, we may expect that the resulting system of algebraic equations for average displacement values will yield an approximation to the equilibrium problem for (1)-(3) as $h\to 0$.

This approach can be made practical only if the transmission matrices R_ω^h in (5) can be approximated without an a priori knowledge of the transmission operators R_ω related to the continuous problem. A general method for constructing R_ω^h on arbitrary cells will be described in a separate paper.

In this paper we describe a simple construction for R_ω^h which results when cubical cells are employed. In this case, the equilibrium conditions (5) and the balance of traction conditions can be given a particularly simple form using finite difference notations and which we then call a compact finite difference scheme. Finite difference methods can then be used to obtain a finite sum energy estimate and also to characterize the average displacement values as the solution of a quadratic variational problem. As a result, the algebraic solution of the compact scheme can be obtained by either direct or iterative methods using a priori facts about the structure of the algebraic system. This development is explained below.

I.2. A compact difference scheme. In this section we describe a compact difference scheme (10) which approximately describes the equilibrium of a cell due to displacements and tractions on the cell faces. We then develop an energy estimate and present a variational principle for the difference scheme. An appropriate method for solving the difference scheme is given in the following section.

We suppose that Ω can be partitioned into regular cubical cells whose faces are parallel to the coordinate axes; $\omega(\mathbf{x})$ indicates a cell with centerpoint at $\mathbf{x}=(x_1, x_2, x_3)$. We denote the average value of a function ϕ on a face whose centerpoint is $\boldsymbol{\xi}$ by $\phi(\boldsymbol{\xi})$. If $h_i \equiv \Delta x_i/2$, $i=1,2,3$, the volume of ω is $\Delta\omega = 8h_1h_2h_3$.

Next, define central average and central divided difference operators μ_1 and δ_1 by

$$\begin{aligned} \mu_1\phi(\mathbf{x}) &\equiv (\phi(x_1+h_1, x_2, x_3)+\phi(x_1-h_1, x_2, x_3))/2, \\ \delta_1\phi(\mathbf{x}) &\equiv (\phi(x_1+h_1, x_2, x_3)-\phi(x_1-h_1, x_2, x_3))/2h_1. \end{aligned} \tag{6}$$

The operators μ_2, δ_2, μ_3 and δ_3 are defined similarly. Also define

$$\begin{aligned} \operatorname{grad}_h \phi &\equiv (\delta_1\phi, \delta_2\phi, \delta_3\phi)^T, \\ \operatorname{div}_h \mathbf{u} &\equiv \delta_1u_1+\delta_2u_2+\delta_3u_3. \end{aligned} \tag{7}$$

Finally, we write

$$\begin{aligned} \boldsymbol{\varepsilon}^h(\mathbf{u}) &\equiv \operatorname{grad}_h (\mathbf{u}+\mathbf{u}^T)/2, \\ \hat{\boldsymbol{\varepsilon}}^h(\mathbf{u}) &\equiv \operatorname{grad}_h (\mathbf{u}-\mathbf{u}^T)/2, \\ \boldsymbol{\tau}_i^h(\mathbf{u}) &= \sum_{j=1}^{3} c_{ij}\boldsymbol{\varepsilon}_j^h(\mathbf{u}), \qquad i=1,2,3. \end{aligned} \tag{8}$$

Thus, $\varepsilon^h(\mathbf{u})$, $\tau^h(\mathbf{u})$ and $\hat{\varepsilon}^h(\mathbf{u})$ are finite difference approximations to the strain, stress, and rotation tensors respectively.

The restriction to cube-like cells simplifies the evaluation of surface tractions. Let $\xi_i^{\pm}$, $i=1,2,3$, denote the centerpoints of the opposite faces of a cell ω. The outward normals $\mathbf{n}(\xi_i^{\pm})$ satisfy $\mathbf{n}(\xi_i^{+}) = -\mathbf{n}(\xi_i^{-})$ so that the average surface tractions $\mathbf{p}(\xi_i^{\pm})$ are given by

$$\mathbf{p}(\xi_i^{\pm}) = \pm\boldsymbol{\tau}_i(\xi_i^{\pm}), \qquad i=1,2,3. \tag{9}$$

In the following discussion the stress components arise as traction forces on the faces of cells. An important consequence is that then the conditions for the balance of average traction forces across cell faces simply reduce to the conditions that the jump in value of $\boldsymbol{\tau}_i$ vanish across a face $x_i = \text{const.}$, $i=1,2,3$.

Corresponding to the equilibrium equations (1), we propose to consider the following compact scheme: in each cell ω

$$
\begin{aligned}
&\text{(a)} && \delta_1\tau_1+\delta_2\tau_2+\delta_3\tau_3=0, && \\
&\text{(b)} && \mu_i\tau_i=\tau_i^h(\mathbf{u})+\gamma^2h_i^2\hat{\varepsilon}_i^h(\mathbf{u}), && i=1,2,3, \\
&\text{(c)} && \mu_i\mathbf{u}-\kappa^2h_i^2\delta_i\tau_i=\boldsymbol{\lambda}, && i=1,2,3.
\end{aligned} \tag{10}
$$

Here γ, κ are parameters, $\kappa\neq 0$, and $\boldsymbol{\lambda}$ is a vector determined such that the values $\delta_i\tau_i$ in (c) satisfy (a), i.e.,

$$
\boldsymbol{\lambda}=\left(\sum_{i=1}^{3}\rho_i\mu_i\mathbf{u}\right)\Big/\sum_{i=1}^{3}\rho_i\equiv\boldsymbol{\lambda}(\mathbf{u}) \tag{10d}
$$

where $\rho_i=(\kappa h_i)^{-2}$.

Clearly, (10) is consistent with (1) for $h\to 0$. Since $\kappa\neq 0$ by assumption, (10) can be written

$$
\begin{aligned}
&\mu_i\tau_i=\tau_i^h(\mathbf{u})+\gamma^2h_i^2\hat{\varepsilon}_i^h(\mathbf{u}), \\
&\delta_i\tau_i=\rho_i(\mu_i\mathbf{u}-\boldsymbol{\lambda}(\mathbf{u})),
\end{aligned} \qquad i=1,2,3 \tag{10}'
$$

where $\boldsymbol{\lambda}(\mathbf{u})$ is given by (10d). The assumption $\gamma\neq 0$ insures that the only solution $\mathbf{u}$ of (10)′ for which $\mu_i\tau_i=\delta_i\tau_i=0$, $i=1,2,3$, is $\mathbf{u}=$ const. and is the discrete analogue of the fact that the only displacements which produce zero stresses in the continuous problem are $\mathbf{u}=$ const.

Figure 1 indicates, with reference to a rectangular cell, the variables associated with the sides of the cell by the scheme (10). These equations thus result in a system of 18 algebraic equations for the 18 components of $\mathbf{u}$ and the 18 components of the tractions on the faces of a cell.

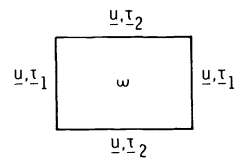


FIG. 1. *Variables associated with cell sides by the difference equations in two dimensions.*

We now indicate how (10) leads to a finite-sum energy estimate. Recall that the work W done on the body can be evaluated from (1) by the use of integration by parts and the use of Gauss' theorem with the result

$$
2W=\sum_{i,j}\int_\Omega\varepsilon_{ij}\tau_{ij}\,d\omega=\int_{\Gamma_1}\mathbf{p}^T\mathbf{u}^0\alpha\gamma+\int_{\Gamma_2}\mathbf{u}^T\mathbf{p}^0\,d\gamma-\int_\Omega\mathbf{u}^T\operatorname{div}\tau\,d\omega. \tag{11}
$$

For finite differences summation by parts results from the identity

$$
\delta_i(\phi\psi)=(\mu_i\phi)(\delta_i\psi)+(\mu_i\psi)(\delta_i\phi),\qquad i=1,2,3. \tag{12}
$$

using (6). Also, using (7), Gauss' theorem holds in the form

$$
\sum_{\omega\in\Omega}\operatorname{div}_h\mathbf{v}\Delta\omega=\sum_{\Gamma}\mathbf{v}^T\mathbf{n}\Delta\gamma, \tag{13}
$$

in which $\Delta\gamma$ is the area of a face of a cell on which $\mathbf{n}$ is the outward normal.

Using (12),

$$\operatorname{div}_h \mathbf{u}^T\tau = \sum_{i=1}^{3} \mu_i \mathbf{u}^T \cdot \delta_i\tau_i + (\varepsilon, \tau)_h, \tag{14}$$

where, from (8),

$$(\varepsilon, \tau)_h \equiv \sum_{i=1}^{3} [(\varepsilon_i^h(\mathbf{u}))^T \tau_i^h(\mathbf{u}) + \gamma^2 h_i^2 (\hat{\varepsilon}_i^h(\mathbf{u}))^T \hat{\varepsilon}_i^h(\mathbf{u})]. \tag{15}$$

Next, using (10c),

$$\sum_i (\mu_i \mathbf{u}^T)(\delta_i\tau_i) = \boldsymbol{\lambda}^T \sum_i \delta_i\tau_i + \sum_i \kappa^2 h_i^2 (\delta_i\tau_i)^2$$

where $\boldsymbol{\lambda}$ is determined by (10d). Thus, if we define

$$[\varepsilon, \tau]_h \equiv (\varepsilon, \tau)_h + \sum_i \kappa^2 h_i^2 (\delta_i\tau_i)^2, \tag{16}$$

then (14) assumes the form

$$\operatorname{div}_h \mathbf{u}^T\tau = [\varepsilon, \tau]_h + \boldsymbol{\lambda}^T \operatorname{div}_h \tau. \tag{17}$$

Recalling earlier remarks we see that $[\varepsilon, \tau]_h \geqq 0$ with equality holding iff $\mathbf{u} = \text{const}$. Also, recalling (9), $\operatorname{div}_h \mathbf{u}^T\tau$ is seen to represent the work per unit volume done by traction on the faces of a cell ω.

Summing (17) over cells in Ω

$$\sum_{\omega\in\Omega} \operatorname{div}_h \mathbf{u}^T\tau\Delta\omega = \sum_{\omega\in\Omega} [\varepsilon, \tau]_h \Delta\omega + \sum_{\omega\in\Omega} \boldsymbol{\lambda}^T \operatorname{div}_h \tau\Delta\omega, \tag{18}$$

so that for a solution of the compact scheme (10)

$$\sum\!\!\!\!\!\!\sum_{\omega\in\Omega} \operatorname{div}_h \mathbf{u}^T\tau\Delta\omega = \sum\!\!\!\!\!\!\sum_{\omega\in\Omega} [\varepsilon, \tau]_h \Delta\omega \geqq 0. \tag{19}$$

Also, using Gauss' theorem in the form (13), (18) can be written (compare (11))

$$\sum_{\omega\in\Omega} [\varepsilon, \tau]_h \Delta\omega = \sum_{\Gamma_1} \mathbf{p}^T\mathbf{u}^0\Delta\gamma + \sum_{\Gamma_2} \mathbf{u}^T\mathbf{p}^0\Delta\gamma - \sum_{\omega\in\Omega} \boldsymbol{\lambda}^T \operatorname{div}_h \tau\Delta\omega, \tag{20}$$

where use has been made of (9). In this equation, $\mathbf{u}^0$, $\mathbf{p}^0$ are to be interpreted as the average values on cell faces on Γ of the data given by (3).

Consider a problem in which $\mathbf{u}$ is prescribed everywhere on Γ. The preceding discussion can be used to verify that the average values of displacements on cell faces which solve the compact scheme (10) also solve the variational problem

$$\min_{\mathbf{u}} \sum_{\omega\in\Omega} [\varepsilon, \tau]_h \Delta\omega = \sum\!\!\!\!\!\!\sum_{\Gamma} \mathbf{p}^T\mathbf{u}^0\Delta\gamma \tag{21}$$

for $\mathbf{u}$ satisfying the boundary conditions on Γ. The Euler conditions for this problem simply express the balance of traction forces, expressed in terms of $\mathbf{u}$ by the use of (10), across cell faces.

Having obtained an energy estimate in the form (20), it is not a major step to establish convergence. As mentioned earlier, we will describe elsewhere how these ideas may be adopted to treat cells having general shapes. For this reason we shall not present here the details of the convergence argument as it applies to (10) except to cite the result: for any fixed values of the parameters κ, γ, the solution $\mathbf{u}^h$ of (10) converges to the solution $\mathbf{u}$ of (1) with accuracy $O(h^2)$ in an l_2-norm while $\mathbf{p}^h(u^h)$

converges to $\mathbf{p}(u)$ with an accuracy $O(h)$. These remarks apply, of course, only to sufficiently smooth solutions of (1).

An important feature of the compact scheme (10) is that it only employs values of the displacements and tractions which arise as average values on cell sides. This is in contrast to many finite element methods which employ edge and vertex values of $\mathbf{u}$.

I.3. Solution method. The compact scheme (10) may be solved by direct algebraic techniques such as Gauss elimination, considering the displacements and tractions as unknown variables. A preferable approach, which we shall now describe, is to eliminate the traction variables so as to obtain an algebraic system involving only the displacement variables. We first indicate, in general form, the steps which lead to this elimination. The specific result upon which numerical calculations can be performed is given by (32).

Let $\gamma_i(\omega)$, $i=1,2,\cdots,6$ indicate a face of a cell ω and write

$$\begin{aligned}[\mathbf{u}]_\gamma &\equiv [\mathbf{u}(\gamma_1), \mathbf{u}(\gamma_2), \cdots, \mathbf{u}(\gamma_6)]^T,\\ [\mathbf{p}]_\gamma &\equiv [\mathbf{p}(\gamma_1), \mathbf{p}(\gamma_2), \cdots, \mathbf{p}(\gamma_6)]^T,\end{aligned} \tag{22}$$

where, for brevity, reference to ω has been omitted.

Using (10)′ we may write the tractions $[\mathbf{p}]_\gamma$ in terms of the displacements $[\mathbf{u}]_\gamma$ in the form

$$[\mathbf{p}]_\gamma = R_\omega^h [\mathbf{u}]_\gamma, \tag{23}$$

where R_ω^h is a block 6×6 transmission matrix associated with ω which we write in terms of its rows as

$$R_\omega^h \equiv \begin{bmatrix} \mathbf{r}_\omega^h(\gamma_1) \\ \mathbf{r}_\omega^h(\gamma_2) \\ \vdots \\ \mathbf{r}_\omega^h(\gamma_6) \end{bmatrix}. \tag{24}$$

A direct evaluation using (10)′ shows that this matrix is symmetric. Thus (23) states

$$\mathbf{p}(\gamma_i(\omega)) = \mathbf{r}_\omega^h(\gamma_i(\omega))[\mathbf{u}]_{\gamma(\omega)}, \qquad i=1,2,\cdots,6. \tag{25}$$

As noted earlier, $\operatorname{div}_h \mathbf{u}^T\tau$ represents the work per unit volume due to the tractions arising from the displacements on the faces of ω. Let

$$\Delta\gamma = \operatorname{diag}(\Delta\gamma_1, \Delta\gamma_2, \cdots, \Delta\gamma_6)^T.$$

In terms of the notations just described and recalling (19) we then have

$$0 \leqq (\operatorname{div}_h \mathbf{u}^T\tau)\Delta\omega = [\mathbf{u}]_\gamma^T \Delta\gamma [\mathbf{p}]_\gamma = [\mathbf{u}]_\gamma^T \Delta\gamma R_\omega^h [u]_\gamma. \tag{26}$$

This shows that R_ω^h is positive semidefinite since the equality in (26) holds, according to (19), iff $[\varepsilon, \tau]_h \geqq 0$ and this was seen to hold iff $\mathbf{u}(\gamma_i) = \text{constant}$, $i=1,2,\cdots,6$.

Since $[\varepsilon, \tau]_h = \operatorname{div}_h \mathbf{u}^T\tau$, the variational principle (21) also applies in the form

$$\min_{\mathbf{u}} \sum_{\omega\in\Omega} [\mathbf{u}]_{\gamma(\omega)}^T \Delta\gamma R_\omega^h [\mathbf{u}]_{\gamma(\omega)}. \tag{27}$$

The conditions for a minimum resulting from this problem are, as remarked earlier, simply the balance of average traction forces across any face $\hat{\gamma}$ common to any neighboring cells which, using (25), may be written as

$$\mathbf{r}_\omega^h(\hat{\gamma})[\mathbf{u}]_{\gamma(\omega)} + \mathbf{r}_{\omega'}^h(\hat{\gamma})[\mathbf{u}]_{\gamma(\omega')} = 0. \tag{28}$$

If $\hat{\gamma}$ lies on Γ_1, then also

$$\mathbf{u}(\hat{\gamma}) = \mathbf{u}^0(\hat{\gamma}) \tag{29a}$$

while if $\hat{\gamma}$ lies on Γ_2, then

$$\mathbf{r}_\omega^h(\hat{\gamma})[\mathbf{u}]_{\gamma(\omega)} = \mathbf{p}^0(\hat{\gamma}). \tag{29b}$$

Earlier remarks show that this system has a unique solution for $\mathbf{u}(\hat{\gamma})$ unless $\Gamma_1 = \phi$.

We now give the explicit result of applying (28) on cell faces. Indicating by $\mathbf{p}(\xi_i^{\pm})$ the tractions on cell sides, as in (9), and using the definitions of the operators μ_i, δ_i given by (6) we have

$$\mathbf{p}(\xi_i^{\pm}) = h_i \delta_i \boldsymbol{\tau}_i \pm \mu_i \boldsymbol{\tau}_i. \tag{30}$$

Let $\omega_{l,m,n}$ indicate a cell whose centerpoint is $\mathbf{x} = (l\Delta x_1, m\Delta x_2, n\Delta x_3)$. Then across the face of $\omega_{l,m,n}$ incident with $\omega_{l+1,m,n}$ (28) may be expressed, using (30), as

$$(h_1\delta_1\boldsymbol{\tau}_1 + \mu_1\boldsymbol{\tau}_1)|_{\omega_{l,m,n}} + (h_1\delta_1\boldsymbol{\tau}_1 - \mu_1\boldsymbol{\tau}_1)|_{\omega_{l+1,m,n}} = 0 \tag{31}$$

or, more simply,

$$\mu_1(\delta_1\boldsymbol{\tau}_1) - \delta_1(\mu_1\boldsymbol{\tau}_1) = 0.$$

Using (10)′, it then follows that

$$\rho_1\mu_1(\mu_1\mathbf{u} - \boldsymbol{\lambda}(\mathbf{u})) = \delta_1(\boldsymbol{\tau}_1^h(\mathbf{u}) + \gamma^2 h_1^2 \hat{\boldsymbol{\varepsilon}}_1^h(\mathbf{u})).$$

More generally the balance of traction conditions across each of the sides of $\omega_{l,m,n}$ may be expressed as

$$\rho_i\mu_i(\mu_i\mathbf{u} - \boldsymbol{\lambda}(\mathbf{u})) = \delta_i(\boldsymbol{\tau}_i^h(\mathbf{u}) + \gamma^2 h_i^2 \hat{\boldsymbol{\varepsilon}}_i^h(\mathbf{u})), \qquad i = 1, 2, 3 \tag{32}$$

in which $\boldsymbol{\lambda}(\mathbf{u})$ is given by (10d) and $\boldsymbol{\tau}_i^h(\mathbf{u})$ is given by (8).

These equations are supplemented by the boundary conditions (3). At points of the boundary on which tractions are prescribed the use of (30), in which $\delta_i\boldsymbol{\tau}_i$, $\mu_i\boldsymbol{\tau}_i$ are given by (10)′, leads to an effective and simple means of imposing traction boundary conditions and will be employed in the example discussed below.

In these equations we have left unspecified the parameters γ, κ. Provided that $\kappa \neq 0$ (cf. [1]), these may, as indicated in the next section, be chosen for convenience since, as indicated earlier, their particular choice will not affect the asymptotic behavior of the solution as $h \to 0$.

We call (32) the stress-eliminated form of the compact scheme (10).

In summary, then, the system of equilibrium equations (28), (29), in which R_ω^h is the transmission matrix for the compact scheme (10), arises as the Euler equations for a related positive definite quadratic variational problem. For three-dimensional problems, in particular, it is natural to consider iterative solution methods.

II. Example of an isotropic material.

II.1. Equilibrium equations. We consider an isotropic material characterized by Young's modulus E and Poisson's ratio ν. We use a plane stress assumption to formulate the problem in two dimensions, the x_1-x_2 plane say. This involves setting $\tau_3 = 0$ and assuming that the stress components τ_{11}, τ_{22}, τ_{21} are independent of x_3 (Timoshenko and Goodier [2]). In two dimensions, the stress-strain relationship (1c) may be written

in component form as follows:

$$\tau_{11} = \zeta\varepsilon_{11} + \eta\varepsilon_{22},$$
$$\tau_{22} = \eta\varepsilon_{11} + \zeta\varepsilon_{22},$$
$$\tau_{12} = \tau_{21} = \sigma\varepsilon_{12}$$

or, in terms of the displacement $\mathbf{u}$,

$$\begin{aligned} \tau_{11} &= \zeta\,\partial_{x_1}u_1 + \eta\,\partial_{x_2}u_2, \\ \tau_{22} &= \eta\,\partial_{x_1}u_1 + \zeta\,\partial_{x_2}u_2, \\ \tau_{12} &= \tau_{21} = \tfrac{1}{2}\sigma(\partial_{x_2}u_1 + \partial_{x_1}u_2). \end{aligned} \tag{33}$$

The parameters ζ, η and σ are given in terms of Young's modulus and Poisson's ratio as follows;

$$\zeta = \frac{E}{(1-\nu^2)}, \quad \eta = \frac{E\nu}{(1-\nu^2)}, \quad \sigma = \frac{E}{(1+\nu)}.$$

The quantity $\frac{1}{2}\sigma$ is known as the shear modulus.

II.2. Method of solution. In this section we write down the compact scheme for the two-dimensional case when square cells are employed and obtain the transmission matrix which relates the tractions and displacements in each cell. The properties of the resulting system are then discussed in the context of its iterative solution.

Analogous with (10), and upon elimination of the strains, we have the following compact scheme for the components of displacement and stress:

$$\begin{aligned} &\delta_{x_1}\tau_{11} + \delta_{x_2}\tau_{12} = 0, \\ &\delta_{x_1}\tau_{21} + \delta_{x_2}\tau_{22} = 0, \end{aligned} \tag{34a}$$

$$\begin{aligned} &\mu_{x_1}\tau_{11} = \zeta\delta_{x_1}u_1 + \eta\delta_{x_2}u_2, \\ &\mu_{x_2}\tau_{22} = \eta\delta_{x_1}u_1 + \zeta\delta_{x_2}u_2, \\ &\mu_{x_1}\tau_{21} = \tfrac{1}{2}\sigma(\delta_{x_2}u_1 + \delta_{x_1}u_2) + \tfrac{1}{2}\gamma^2h^2(\delta_{x_1}u_2 - \delta_{x_2}u_1), \\ &\mu_{x_2}\tau_{12} = \tfrac{1}{2}\sigma(\delta_{x_2}u_1 + \delta_{x_1}u_2) + \tfrac{1}{2}\gamma^2h^2(\delta_{x_2}u_1 - \delta_{x_1}u_2), \end{aligned} \tag{34b}$$

$$\begin{aligned} &\mu_{x_1}u_1 - \kappa^2h^2\delta_{x_1}\tau_{11} = \mu_{x_2}u_1 - \kappa^2h^2\delta_{x_2}\tau_{21}, \\ &\mu_{x_2}u_2 - \kappa^2h^2\delta_{x_2}\tau_{21} = \mu_{x_2}u_2 - \kappa^2h^2\delta_{x_2}\tau_{22}. \end{aligned} \tag{34c}$$

Recalling (9) we may solve for the tractions $[\mathbf{p}]_\gamma$ in terms of the displacements $[\mathbf{u}]_\gamma$ in each cell ω to obtain

$$[\mathbf{p}]_\gamma = R_\omega^h[\mathbf{u}]_\gamma, \tag{35}$$

where

$$[\mathbf{p}]_\gamma = (\mathbf{p}(\xi_1^+), \mathbf{p}(\xi_2^+), \mathbf{p}(\xi_1^-), \mathbf{p}(\xi_2^-))^T,$$
$$[\mathbf{u}]_\gamma = (\mathbf{u}(\xi_1^+), \mathbf{u}(\xi_2^+), \mathbf{u}(\xi_1^-), \mathbf{u}(\xi_2^-))^T,$$

and in which the transmission matrix R_ω^h is given by

$$R_\omega^h = \frac{\theta^{-1}}{2h}\begin{bmatrix} I+\theta c_{11} & -(I-\theta c_{12}) & I-\theta c_{11} & -(I+\theta c_{12}) \\ -(I-\theta c_{21}) & I+\theta c_{22} & -(I+\theta c_{21}) & I-\theta c_{22} \\ I-\theta c_{11} & -(I+\theta c_{12}) & I+\theta c_{11} & -(I-\theta c_{12}) \\ -(I+\theta c_{21}) & I-\theta c_{22} & -(I-\theta c_{21}) & I+\theta c_{22} \end{bmatrix}, \tag{36}$$

in which $\theta = 2\kappa^2$ and

$$c_{11} = \begin{bmatrix} \zeta & 0 \\ 0 & (\sigma + \gamma^2 h^2)/2 \end{bmatrix}, \qquad c_{22} = \begin{bmatrix} (\sigma + \gamma^2 h^2)/2 & 0 \\ 0 & \zeta \end{bmatrix},$$

$$c_{12} = c_{21}^T = \begin{bmatrix} 0 & \eta \\ (\sigma - \gamma^2 h^2)/2 & 0 \end{bmatrix}.$$

The matrix R_ω^h is obviously symmetric and is easily shown to be positive semi-definite for positive θ. Balancing the tractions across vertical and horizontal faces common to neighboring cells, we obtain, with reference to Fig. 2,

$$\begin{aligned} &2(I + \theta c_{11})\mathbf{u}(P_0) + (I - \theta c_{11})(\mathbf{u}(P_1) + \mathbf{u}(P_2)) \\ &\quad = (I + \theta c_{12})(\mathbf{u}(Q_1) + \mathbf{u}(Q_4)) + (I - \theta c_{12})(\mathbf{u}(Q_0) + \mathbf{u}(Q_3)), \end{aligned} \tag{37a}$$

$$\begin{aligned} &2(I + \theta c_{22})\mathbf{u}(Q_0) + (I - \theta c_{22})(\mathbf{u}(Q_1) + \mathbf{u}(Q_2)) \\ &\quad = (I + \theta c_{21})(\mathbf{u}(P_1) + \mathbf{u}(P_4)) + (I - \theta c_{21})(\mathbf{u}(P_0) + \mathbf{u}(P_3)). \end{aligned} \tag{37b}$$

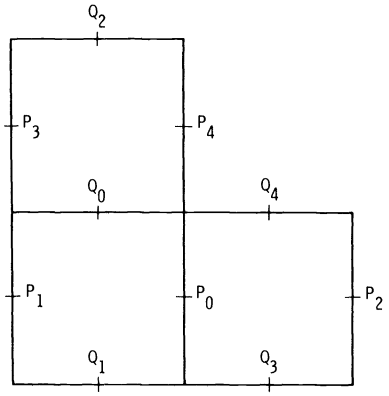


FIG. 2. *Points associated with the stress-eliminated equations* (37).

These equations correspond to the stress-eliminated form (32). If $\Gamma_2 = \phi$, i.e., the values of the displacements are prescribed on the boundary, then the coefficient matrix of the above problem is symmetric and positive definite. A natural iterative method for solving (37) is line SOR. The method involves solving block tridiagonal systems firstly along all horizontal lines and then along vertical lines. In the next section we will compare this and point relaxation methods on a simple problem.

II.3. Numerical example. Consider a square domain on whose vertical edges the displacements are fixed and which experiences a uniform load along its top horizontal surface. This situation is illustrated in Fig. 3.

The boundary conditions are

$$\begin{aligned} &\mathbf{u} = (0, 0)^T \quad \text{on } x = 0, 1, \\ &\tau_2 = (0, 0)^T \quad \text{on } y = 0, \\ &\tau_2 = (0, -1)^T \quad \text{on } y = 1. \end{aligned}$$

Our experiments were performed using the plane stress approximation with values of Young's modulus and Poisson's ratio given by $E = 10^7$, $\nu = 0.3$. The equilibrium

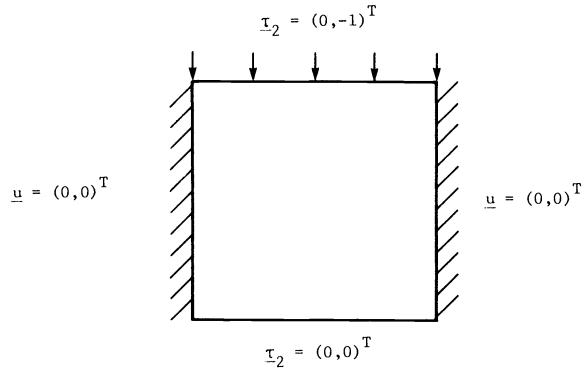


FIG. 3. *Description of the plane stress test problem.*

displacement u_1 is anti-symmetric and u_2 is symmetric about the line $x = \frac{1}{2}$. However, in the computations that were performed, no advantage was taken of this symmetry.

We have remarked earlier that a fixed choice of the parameters γ, κ will not affect the asymptotic convergence rate of the scheme. For $\gamma \neq 0$, $\kappa \neq 0$ the system of equations for the displacements is positive definite and leads to a unique solution for the displacements. Reference to (10)′ shows that the condition $0 \neq \kappa^2 = (\sigma + \gamma^2 h^2)^{-1}$ is required if the tractions are to be determined from the displacements in any cell. The condition $\gamma \neq 0$ simply insures that any two displacement distributions which produce the same traction forces in a cell differ by a constant, while if $\gamma = 0$ the same stress distribution will be produced by more general displacement fields. Since our primary interest in the following calculations was to determine the stress field we chose the values of $\gamma = 0$, $\kappa^2 = 1/\sigma$.

In our experiments we used the point Gauss-Seidel, point SOR, and line SOR methods. The parameters for the SOR methods were chosen to be the optimum ones for Laplace's equation. Initially, the value of **u** was taken to be zero at all the grid points. Keeping in mind that the method can be expected to yield only second-order accuracy, the iterations were terminated when the l_2-norm of the residuals was less than 10^{-3}.

In Table 1 we show the dependence of the number of iterations required to attain the convergence criterion on the mesh size for various iterative schemes.

TABLE 1
Dependence of number of iterations on mesh size.

h	pt. Gauss-Seidel	pt. SOR	line SOR
1/2	42	34	—
1/4	100	54	30
1/8	325	97	51
1/16	1240	191	99

These results indicate that the rates of convergence of the Gauss-Seidel and SOR methods on this problem are $O(h^2)$ and $O(h)$ respectively. Table 2 contains the values of the l_2-norms of the solutions to the stress-eliminated equations (37) for different grid sizes.

We note that the convergence of $\|u_1\|_2$ is $O(h^2)$ while that of $\|u_2\|_2$ is $O(h^{3/2})$ as $h \to 0$. This degradation in behavior is due to the singularities which are located at the top corners.

TABLE 2
l_2-norms of numerical solution.

h	$\|u_2\|_2$	$\|u_2\|_2$
1/2	0.0221	0.0979
1/4	0.0086	0.0710
1/8	0.0055	0.0603
1/16	0.0048	0.0568
1/32	0.0046	0.0556

The values of the tractions are calculated by substituting the values of $\mathbf{u}$ in (35). The components of displacement and traction display the relevant symmetric or anti-symmetric properties to four decimal places. There are two integral checks that can be carried out to verify the computations, namely

$$-\int_0^1 \tau_{11}(0, y)\, dy + \int_0^1 \tau_{11}(1, y)\, dy = 0,$$
$$\int_0^1 \tau_{21}(1, y)\, dy = -\tfrac{1}{2}. \tag{38}$$

These conditions express the integral condition

$$\int_\Gamma \tau \cdot \mathbf{n}\, d\gamma = 0$$

using the assumed symmetry of τ_{12} about $x = \frac{1}{2}$. These integrals were computed numerically using the midpoint rule. The first integral check held exactly while the second one was found to be correct to the number of decimal places specified in the displacement calculation.

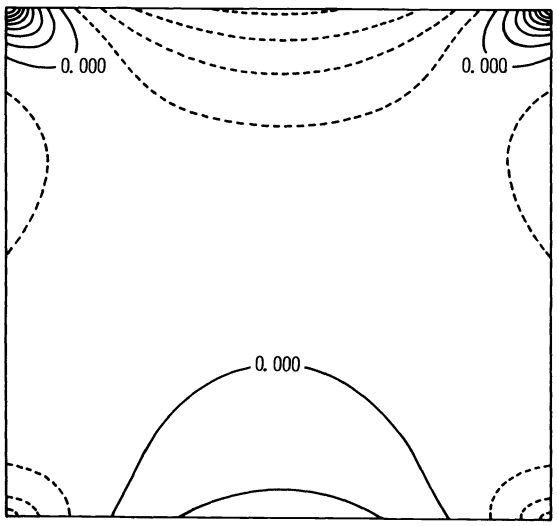
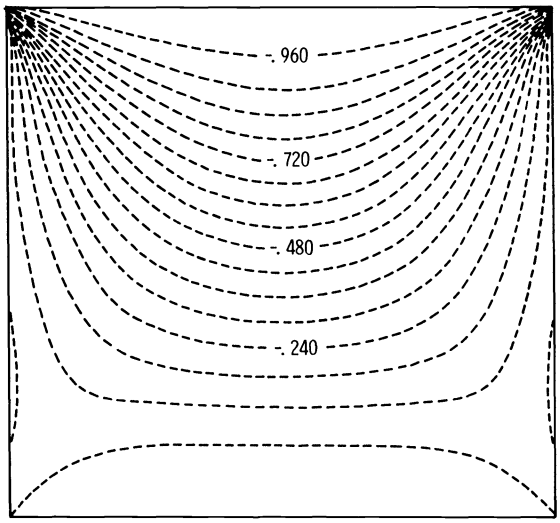


FIG. 4. *Plot of the stress component τ_{11}.*

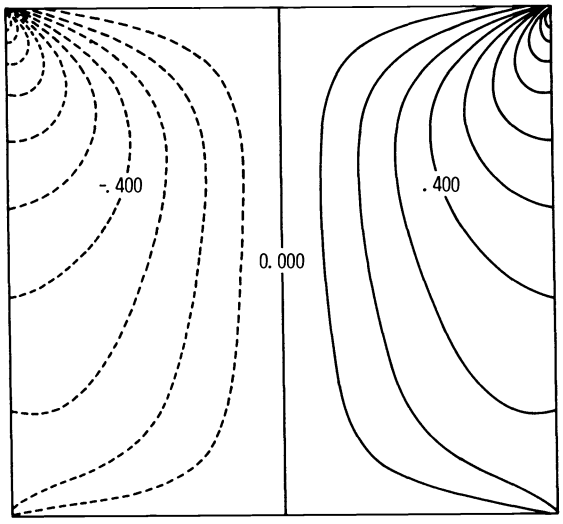
Various plots are given of the solution. Figures 4, 5, 6 are contour plots of the stress components τ_{11}, τ_{22}, τ_{12} respectively. These were obtained using 64 cells in each direction. Figures 7a and 7b show the principal stress vectors within each computational cell. The principal stress directions are defined to be those vectors $\mathbf{x} \neq 0$ which satisfy the eigenproblem

$$(\tau - \lambda I)\mathbf{x} = \mathbf{0}.$$



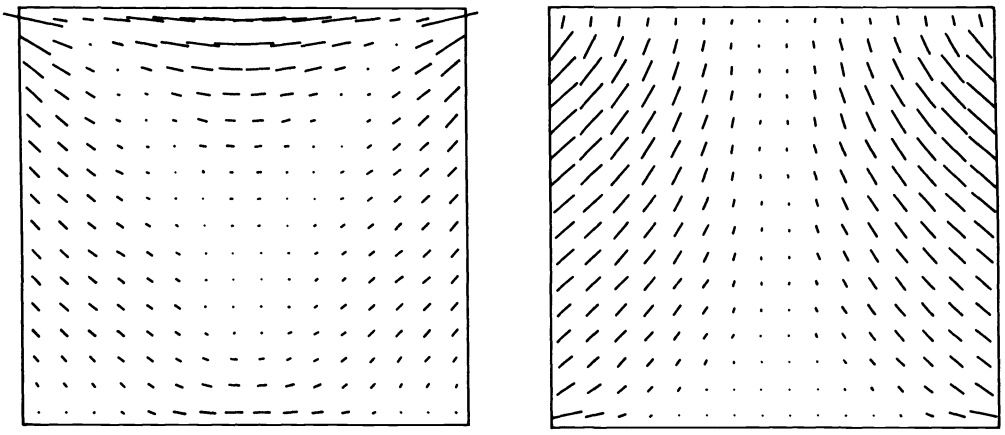
CONTOUR FROM -. 96000 TO 0
CONTOUR INTERVAL IS . 60000E-01
τ_{22}

FIG. 5. *Plot of the stress component* τ_{22}.



CONTOUR FROM -1. 1000 TO 1. 1000
CONTOUR INTERVAL IS . 10000E+00
τ_{12}

FIG. 6. *Plot of the stress component* τ_{12}.



(a) (b)

FIG. 7. *Principal stress vectors.*

Concluding remarks. The treatment of equilibrium with a volume force $\mathbf{f}$ requires no essential modifications to the method. In this case (1a) has the form $\sum_i \partial_{x_i} \boldsymbol{\tau}_i = \mathbf{f}$; correspondingly (10a) is modified to

$$\sum_i \delta_{x_i} \boldsymbol{\tau}_i = \mathbf{f}. \tag{10a'}$$

We leave the details of the consequent developments to the reader.

Acknowledgments. We wish to thank A. Noor for encouragement and helpful discussions as well as M. Dunn and P. Spence for computational assistance. We also thank S. Ta'asan for some critical observations.

REFERENCES

[1] T. N. PHILLIPS AND M. E. ROSE, *A finite difference scheme for a class of first-order elliptic partial differential equations*, Comput. Math. Appl., to appear.

[2] S. P. TIMOSHENKO AND J. N. GOODIER, *Theory of Elasticity*, McGraw-Hill, New York, 1970.

[3] R. S. VARGA, *Matrix Iterative Analysis*, Prentice-Hall, Englewood Cliffs, NJ, 1962.

[4] D. M. YOUNG, *Iterative Solution of Large Linear Systems*, Academic Press, London and New York, 1971.

SIAM J. SCI. STAT. COMPUT.
Vol. 7, No. 1, January 1986

020

FULLY ADAPTIVE SOLUTIONS OF ONE-DIMENSIONAL MIXED INITIAL-BOUNDARY VALUE PROBLEMS WITH APPLICATIONS TO UNSTABLE PROBLEMS IN COMBUSTION*

M. D. SMOOKE† AND M. L. KOSZYKOWSKI‡

Abstract. We develop a fully adaptive method for solving one-dimensional mixed initial-boundary value problems. The method adaptively adjusts the number of grid points needed to equidistribute a positive weight function over a given mesh interval at each time level. In addition, when the solution can be described accurately by a fixed number of points, the mesh is moved by extrapolating the nodal positions from earlier time levels. We monitor the solution from one time level to another to insure that the local error per unit step associated with the time differencing method is below some specified tolerance. The method is applied to several unstable problems in combustion.

Key words. adaptive gridding, unstable combustion, initial-boundary value problems

1. Introduction. Many problems ≷n the physical sciences can be reduced to the solution of a set of coupled time-dependent partial differential equations of the form

$$u_t = f(x, t, u, u_x, u_{xx}), \qquad a < x < b, \quad t > 0, \tag{1.1a}$$

$$g_1(a, t, u(a), u_x(a)) = 0, \qquad t > 0, \tag{1.1b}$$

$$g_2(b, t, u(b), u_x(b)) = 0, \qquad t > 0, \tag{1.1c}$$

$$u(x, 0) = r(x), \qquad a < x < b, \tag{1.1d}$$

where u, f, g_1, g_2 and r are N vectors. Problems of this type can arise in heat and mass transfer—e.g., combustion and flow through porous media. Most of these problems are sufficiently complex that analytical solutions cannot be obtained. As a result, numerical methods must be used.

For problems in which the solution is smoothly varying with respect to the spatial variable (i.e., no sharp peaks or steep fronts), an equispaced or mildly nonuniform grid is ordinarily chosen a priori and then used for the entire calculation. However, for problems that exhibit a high degree of spatial activity, a uniform grid held fixed for the entire calculation can be computationally inefficient. Ideally we want to concentrate grid points in regions where the spatial variation of the solution is large. If the solution can be described accurately with a fixed number of points (e.g., a moving flame front), we want to move the points along with the region of high spatial activity, but if the time evolution of the solution is such that a variable number of points is needed to resolve these regions, we want to change the number of grid points as the calculation progresses.

In this paper we develop an algorithm that obtains fully adaptive (space and time) solutions to mixed initial-boundary value problems of the form (1.1). By discretizing the time derivative in (1.1) we obtain a nonlinear two-point boundary value problem at each time level. We solve the boundary value problems with a finite difference method in which the grid points are chosen such that a positive weight function is equidistributed over each subinterval. This procedure has the advantage of automati-

* Received by the editors June 23, 1983, and in revised form September 11, 1984. This work was supported by U.S. Department of Energy Office of Basic Energy Sciences.

† Department of Mechanical Engineering, Yale University, New Haven, Connecticut 06520.

‡ Sandia National Laboratories, Livermore, California 94550.

cally adjusting the number of grid points at each time level (a variable node, static-rezone approach). However, when the number of points does not change, then for reasons of computational efficiency, we move a fixed number of nodes by extrapolating their positions from previous time levels. In addition, we monitor the solutions of the boundary value problems from one time level to another to insure that the local error per unit step associated with the time differencing method is below some specified tolerance.

In the next section we present some background material on other approaches used to solve mixed initial-boundary value problems with moving adaptive grids, and in § 3 we develop our fully adaptive algorithm. The method is applied to several unstable combustion problems in § 4.

2. Background. Several approaches for solving one-dimensional mixed initial-boundary value problems with moving adaptive grids have recently appeared in the literature. For example, Miller and Miller [12] (see also Gelinas, Doss, and Miller [9]) have implemented a moving finite element method in which the solution is expanded in a series of piecewise linear polynomials where both the time-dependent coefficients of the series and the grid points are unknown functions of time. These quantities are determined by minimizing the partial differential equation's residual in a least squares sense. Davis and Flaherty [4] have also developed an adaptive finite element method. In their implementation the calculation of the solution and the grid points are not coupled together. White [20] has generalized the idea of equidistributing the arc-length of the solution of a two-point boundary value problem to one-dimensional mixed initial-boundary value problems. In his approach the solution and the grid are calculated together by a finite difference method. Other approaches have been considered, for example, by Dwyer, Kee, and Sanders [6], Tscharnuter and Winkler [18], and Bolstad [2].

With the exception of the work by Bolstad on hyperbolic systems, all of these methods move a fixed number of grid points so they are concentrated in regions where the spatial activity of the solution is highest. At each time level, the approaches can be viewed as equidistributing a positive weight function over a fixed number of spatial intervals. Aside from the choice between finite difference and finite element spatial discretizations, the methods vary primarily in the choice of the weight function, and its implicit or explicit dependence upon the solution. In the implicit case the solution components and the grid are calculated together, while in the explicit case the grid is determined by using a previously calculated solution—the solution and grid calculations are not coupled.

For problems in which a fixed number of grid points are to be used throughout the entire calculation, the coupling of the grid and the solution has the potential of accurately placing the grid points in regions of high spatial activity. However, the extra differential equations needed to locate the points can increase significantly the size of the problem as well as the time required to solve it. It is our experience that one does not have to couple the calculation of the grid and the solution components in order to obtain accurate placement of the mesh. In particular, one can obtain accurate grid placement if a static-rezone approach is combined with a moving grid method based upon extrapolation. If the extrapolated grid points have moved out of the high activity regions, we can rezone. In addition, for problems in which the high spatial activity of the solution either increases or decreases, the variable node, static-rezone approach has the ability to either insert or take away grid points to maintain a specified degree of accuracy.

3. Method of solution. The applications we consider occur in the realm of heat and mass transfer—in particular, combustion. The time evolution of these problems ordinarily requires that implicit time discretization methods be employed. For the discussion that follows, we consider a backward-Euler approximation to the time derivative in (1.1). We point out that, in principle, a higher order backward differentiation formula or an implicit Runge Kutta method could be used as well.

Starting from the initial point $t^0 = 0$, we obtain a numerical solution of (1.1) at time levels $0 = t^0 < t^1 < t^2 < \cdots < t^J = \mathcal{T}$, for some finite time $\mathcal{T}$. If for a continuous mapping $g:[0, \mathcal{T}] \to \mathcal{L}^2(a, b)$, we define $g^n(x) = g(x, t^n)$, $n = 0, 1, 2, \cdots, J$, then the system in (1.1) can be written in the form

$$\frac{u^{n+1}(x) - u^n(x)}{\tau^{n+1}} = f(x, t^{n+1}, u^{n+1}, u_x^{n+1}, u_{xx}^{n+1}) + \rho^{n+1}(x), \tag{3.1a}$$

$$g_1(a, t^{n+1}, u^{n+1}(a), u_x^{n+1}(a)) = 0, \tag{3.1b}$$

$$g_2(b, t^{n+1}, u^{n+1}(b), u_x^{n+1}(b)) = 0, \tag{3.1c}$$

$$u^0(x) = r(x), \tag{3.1d}$$

for $n = 0, 1, 2, \cdots, J-1$, where the time step $\tau^{n+1} = t^{n+1} - t^n$ and the discretization error $\rho^{n+1}(x) = [\partial^2 u(x, \xi)/\partial t^2]\tau^{n+1}/2$, $\xi \in [t^n, t^{n+1}]$.

If we rearrange (3.1a) and neglect the discretization error, we have the following semidiscrete approximation to (1.1)

$$f(x, t^{n+1}, \tilde{v}^{n+1}, \tilde{v}_x^{n+1}, \tilde{v}_{xx}^{n+1}) - \frac{\tilde{v}^{n+1}}{\tau^{n+1}} = -\frac{\tilde{v}^n}{\tau^{n+1}}, \tag{3.2a}$$

$$g_1(a, t^{n+1}, \tilde{v}^{n+1}(a), \tilde{v}_x^{n+1}(a)) = 0, \tag{3.2b}$$

$$g_2(b, t^{n+1}, \tilde{v}^{n+1}(b), \tilde{v}_x^{n+1}(b)) = 0, \tag{3.2c}$$

$$\tilde{v}^0(x) = r(x), \tag{3.2d}$$

where $n = 0, 1, 2, \cdots, J-1$, and where the change of notation from u to $\tilde{v}$ recognizes the fact that the solution of (3.1) is in general different from the solution of (3.2). We observe that solution of the original mixed initial-boundary value problem has been reduced to solving a nonlinear two-point boundary value problem at each time level. We solve the problems in (3.2) by applying a finite difference method (though another two-point boundary value solution method could be employed as well).

Newton's method. Our goal is to obtain a discrete solution of (3.2) on the mesh $\mathcal{M}^{n+1}$,

$$\mathcal{M}^{n+1} = \{a = x_0^{n+1} < x_1^{n+1} < \cdots < x_{M^{n+1}}^{n+1} = b\}, \tag{3.3}$$

where $h_j^{n+1} = x_j^{n+1} - x_{j-1}^{n+1}$, $j = 1, 2, \cdots, M^{n+1}$, and $h^{n+1} = \max_{1 \le j \le M^{n+1}} h_j^{n+1}$. The time level superscript on the mesh is used to account for the fact that, in general, the number and/or the location of the grid points can differ from one time level to another.

We approximate the spatial derivatives in (3.2) using finite difference expressions. In particular, omitting the discretization error and the time level superscript, we write

$$\frac{\partial}{\partial x}\left(a(x)\frac{\partial g}{\partial x}\right)_{x_j} \approx \partial^2(a_j g_j) = \left(\frac{2}{x_{j+1} - x_{j-1}}\right)(a_{j+1/2}\,\partial g_{j+1} - a_{j-1/2}\,\partial g_j), \tag{3.4}$$

and

$$\left(\frac{\partial g}{\partial x}\right)_{x_j} \approx \partial g_j, \qquad j = 1, 2, \cdots, M, \tag{3.5}$$

where for a continuous mapping $g:[a, b]\to R^1$, we define $g_j = g(x_j), j = 0, 1, \cdots, M$, and

$$g_{j+1/2}=\frac{(g_{j+1}+g_j)}{2}, \qquad j=0, 1, \cdots, M-1, \tag{3.6}$$

$$\partial g_{j+1}=\frac{(g_{j+1}-g_j)}{h_{j+1}}, \qquad j=0, 1, \cdots, M-1. \tag{3.7}$$

By replacing the continuous differential operators in (3.2) by expressions similar to those in (3.4)-(3.7), we convert the problem of finding an analytic solution of (3.2) to one of finding an approximation to this solution at each point x_j^{n+1} of the mesh $\mathcal{M}^{n+1}$. Upon denoting this approximation by v_j^{n+1}, $j=0, 1, \cdots, M^{n+1}$, we seek the solution $V^{n+1}=(v_0^{n+1}, v_1^{n+1}, \cdots, v_{M^{n+1}}^{n+1})^T$ of the system of nonlinear difference equations

$$\mathcal{F}(V^{n+1}) = F(V^{n+1})-\left(\frac{V^{n+1}-V^n}{\tau^{n+1}}\right)=0, \qquad n\geqq 0, \tag{3.8}$$

where the $(N)(M^{n+1}+1)$ vector F corresponds to the discretization of f, g_1, and g_2 at each point of the mesh $\mathcal{M}^{n+1}$.

For an initial solution estimate that is sufficiently close to V^{n+1}, we can, in principle, solve the nonlinear equations in (3.8) by Newton's method. We write

$$\left(J(V_k^{n+1})-\frac{I}{\tau^{n+1}}\right)(V_{k+1}^{n+1}-V_k^{n+1})=-\lambda_k\mathcal{F}(V_k^{n+1}), \qquad n\geqq 0, \quad k=0, 1, \cdots, \tag{3.9}$$

where V_k^{n+1} denotes the kth solution iterate, λ_k the kth damping parameter $(0<\lambda\leqq 1)$, I the identity matrix, and $J(V_k^{n+1})=\partial F(V_k^{n+1})/\partial V^{n+1}$ the $(N)(M^{n+1}+1)\times(N)(M^{n+1}+1)$ "steady-state" Jacobian matrix. At each iteration a system of linear equations is solved for corrections to the previous solution vector.

For problems in which the cost of forming and then factoring the Jacobian matrix $(J-I/\tau^{n+1})$ is a significant part of the cost of a full Newton step, it is natural to apply the modified version of Newton's method in which the Jacobian is re-evaluated periodically. In such cases we implement the modified Newton method

$$\left(J(V_0^{n+1})-\frac{I}{\tau^{n+1}}\right)(V_{k+1}^{n+1}-V_k^{n+1})=-\mathcal{F}(V_k^{n+1}), \qquad n\geqq 0, \quad k=0, 1, \cdots, \tag{3.10}$$

where the Jacobian matrix is evaluated at the initial solution estimate. The immediate problem one faces when applying the modified method is to determine whether the sequence of successive modified Newton iterates is converging at a fast enough rate. If the rate of convergence is too slow, we want to revert to a full Newton method, make use of new Jacobian information, and possibly employ a damping strategy. An estimate that enables us to determine an upper bound for the size of the sequence of modified Newton iterates, assuming the Kantorovich convergence conditions are satisfied, has been derived in [16]. As a result, if after solving (3.8) we determine the size of $\Delta V_k^{n+1}=V_{k+1}^{n+1}-V_k^{n+1}$ to be larger than the value the estimate in [16] predicts, we form a new Jacobian and restart the modified Newton algorithm with a new initial solution estimate given by V_k^{n+1}.

Adaptive gridding. If the boundary value problems in (3.2) admit solutions that exhibit regions of high spatial activity, it is crucial that the grid points be placed adaptively in these regions to obtain an efficient solution algorithm. The use of an equispaced or mildly nonuniform grid can require a large number of points to obtain

the solution to the same accuracy. One of the reasons initial value (e.g., shooting) methods have been used to solve two-point boundary value problems has been the adaptive mesh capability of the initial value solver. In such methods the solution is monitored and the spatial step size appropriately chosen as the integration proceeds. Global solution procedures such as finite differences and collocation require that a mesh be determined before the calculation is begun.

Many of the methods that have been used to obtain adaptive grid spacings for two-point boundary value problems can be interpreted as equidistributing a positive weight function on a given interval (e.g., see Kautsky and Nichols [10]). Essentially one attempts to determine a mesh $\mathcal{M}$ such that the weight function achieves the same variation over each subinterval. Among the various approaches, White [19] has equidistributed the arclength of the solution, and Pereyra and Sewell [13] have equidistributed the local truncation error. For an excellent discussion of various equidistribution methods, see Russell [15].

The mesh $\mathcal{M}$ is equidistributed on $[a, b]$ with respect to the nonnegative function w and the constant C if

$$\int_{x_j}^{x_{j+1}} w\,dx = C, \qquad j = 0, 1, \cdots, M-1, \tag{3.11}$$

where for convenience, we have omitted the time level superscript. For our purposes, however, we will employ the convention that $\mathcal{M}$ is equidistributed if the integral in (3.11) is less than or equal to C. Our experience with the various equidistribution techniques indicates that while they may all be viable in theory, some are to be preferred over others in practice. We attempt to equidistribute the difference in the components of the solution and its gradient between adjacent mesh points. Upon denoting the vector $\tilde{v} = [\tilde{v}_1, \tilde{v}_2, \cdots, \tilde{v}_N]^T$, we seek to obtain a mesh $\mathcal{M}$ such that

$$\int_{x_j}^{x_{j+1}} \left|\frac{d\tilde{v}_i}{dx}\right| dx \leqq \delta \left| \max_{a \leqq x \leqq b} \tilde{v}_i - \min_{a \leqq x \leqq b} \tilde{v}_i \right|, \tag{3.12}$$

$j = 0, 1, \cdots, M-1$, $i = 1, 2, \cdots, N$, and

$$\int_{x_j}^{x_{j+1}} \left|\frac{d^2\tilde{v}_i}{dx^2}\right| dx \leqq \gamma \left| \max_{a \leqq x \leqq b} \frac{d\tilde{v}_i}{dx} - \min_{a \leqq x \leqq b} \frac{d\tilde{v}_i}{dx} \right|, \tag{3.13}$$

$j = 1, 2, \cdots, M-1$, $i = 1, 2, \cdots, N$, where δ and γ are small numbers less than one and the values of $\max \tilde{v}_i$, $\min \tilde{v}_i$, $\max d\tilde{v}_i/dx$, and $\min d\tilde{v}_i/dx$ are estimated from a numerical solution on a previously determined mesh.

A potential problem of such an equidistribution procedure is the formation of a mesh that may not be smoothly varying. For example, the ratio of consecutive mesh intervals may differ by several orders of magnitude. This can affect the accuracy of the method as well as the convergence properties of the Newton iteration. As a result, we impose the added constraint that the mesh produced by employing (3.12) and (3.13) be locally bounded, i.e., the ratio of adjacent mesh intervals must be bounded above and below by constants (see also [10]). We require

$$\frac{1}{A} \leqq \frac{h_j}{h_{j-1}} \leqq A, \qquad j = 2, 3, \cdots, M, \tag{3.14}$$

where A is a constant $\geqq 1$. This smooths out rapid changes in the size of the mesh intervals.

In the implementation of the adaptive mesh algorithm, we first solve the boundary value problems in (3.2) on a given mesh. The maximum and minimum values of $\tilde{v}_i$ and $d\tilde{v}_i/dx$ are then obtained. We test the inequalities in (3.12) and (3.13) for each of the N dependent solution components at the nodes of the mesh. If either of the inequalities is not satisfied, a grid point is inserted at the midpoint of the interval in question. Once a new mesh has been obtained, we check to see whether it is locally bounded. If it is not, a grid point is inserted at the midpoint of the intervals in which (3.14) is not satisfied. The previously converged numerical solution is interpolated onto this new mesh and the result serves as an initial solution estimate for Newton's method on this finer grid. The process continues until the inequalities in (3.12), (3.13), and (3.14) are satisfied.

We remark that nothing prevents the use of other weight functions in (3.11). In particular, we could choose to equidistribute the local truncation error associated with the spatial differencing. For very "nonlinear" problems, we have found that the grids determined with the weight functions in (3.12) and (3.13) are similar to the grids determined with the local truncation error. In addition, for high order spatial methods, they eliminate the need to evaluate high order derivatives numerically.

Interpolation. We observe that solution of the boundary value problem in (3.2) at time level $n+1$ requires a knowledge of the solution at time level n at the grid points x_j^{n+1}, $j=0,1,\cdots,M^{n+1}$. If the same spatial grid were used for the entire calculation—at time levels $t^0, t^1, \cdots, t^J$—we would have solution values available at the proper mesh points. The physical locations of the grid points at one time level would be identical to the locations of the points at any other time level. However, since we may have to determine adaptively the proper location of the grid every time the boundary value problem is solved, it is likely that the number as well as the location of the grid points will change from one time level to another. In such cases we interpolate solution values from the previous time level to obtain solution information corresponding to the location of the grid points at the current time. (We point out that for finite element boundary value solvers, e.g., COLSYS [1], there is no interpolation problem since a continuous representation of the solution is available.)

In our finite difference method we must be able to relate a given grid point at level $n+1$ with the same physical point at level n. We have chosen to use linear interpolation. For example, suppose the grid points at time levels $n+1$ and n are given as in Fig. 1. We see that the point at level $n+1$ does not coincide with any point from

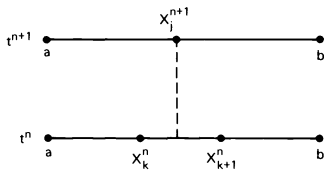


FIG. 1. *Illustration of grid points at time levels n and n+1. The physical location of x_j^{n+1} does not coincide with either x_k^n or x_{k+1}^n.*

level n. We assume, however, that we can bound the point x_j^{n+1} above and below by two points from the nth level—say x_k^n and x_{k+1}^n. To obtain v_j^n we form

$$v_j^n = v_k^n + \left(\frac{v_{k+1}^n - v_k^n}{x_{k+1}^n - x_k^n}\right)(x_j^{n+1} - x_k^n). \tag{3.15}$$

(An equivalent expression can be derived if v_j^n is expanded about x_{k+1}^n.) As a result,

the fully discrete form of (3.2) becomes

$$f(x_j^{n+1}, t^{n+1}, v_j^{n+1}, \partial v_j^{n+1}, \partial^2 v_j^{n+1}) - \frac{v_j^{n+1}}{\tau^{n+1}}$$

$$= -\frac{1}{\tau^{n+1}}\left(v_k^n + \left(\frac{v_{k+1}^n - v_k^n}{x_{k+1}^n - x_k^n}\right)(x_j^{n+1} - x_k^n)\right), \tag{3.16}$$

with appropriate initial and boundary conditions. We point out that an additional spatial discretization error is introduced into our finite difference algorithm as a result of the interpolation procedure used in (3.15). In particular, if we let $\Omega = \{(x, t) | (x, t) \in (a, b) \times (0, \mathcal{T}]\}$, then for a function $g(x, t)$ where $g_{tt} \in C^2(\Omega)$ and $g_{xxx} \in C^3(\Omega)$ we can show

$$\frac{g_j^{n+1} - \left(g_k^n + \left(\frac{g_{k+1}^n - g_k^n}{x_{k+1}^n - x_k^n}\right)(x_j^{n+1} - x_k^n)\right)}{\tau^{n+1}} = \frac{\partial g_j^{n+1}}{\partial t} + O\left(\frac{\tau^{n+1}}{2} + \frac{h_{k+1}^n (x_j^{n+1} - x_k^n)}{\tau^{n+1}}\right). \tag{3.17}$$

We see that the discretization error associated with the time derivative term has an additional error $O(h_{k+1}^n (x_j^{n+1} - x_k^n)/\tau^{n+1})$. As we will see shortly, this error influences the way the adaptive time steps are chosen. In addition, since $|x_j^{n+1} - x_k^n| \leqq h_{k+1}^n/2$ or $|x_j^{n+1} - x_{k+1}^n| \leqq h_{k+1}^n/2$, the finite difference equations are consistent providing $(h_{k+1}^n)^2$ goes to zero faster than $2\tau^{n+1}$.

Grid propagation. One of the advantages in adaptively solving the boundary value problems in (3.2) is the ability to adjust the number of grid points at every time level. This is particularly useful for problems in which the spatial activity of the solution changes as a function of time (e.g., an unstable flame front). To be assured that the spatial portion of the discretization error is uniformly bounded over all time steps, we must be able to vary the number of points as the solution evolves. The finite difference method we employ is implicit with respect to the time discretization. As a result, we must solve systems of nonlinear equations. Since we employ Newton's method, a Jacobian matrix must be formed and factored at each time level. However, if at some time during the calculation the solution can be determined accurately with a fixed number of moving grid points (e.g., a stable flame front), then we can keep the Jacobian fixed; its factorization can be stored and the cost of propagating the solution will, in general, be less than if the variable node, static-rezone approach had been used. For purposes of the discussion that follows we introduce the following definitions.

DEFINITION 1. We say a problem is of Class 1, if the number of (possibly moving) nodes required to satisfy (3.11) varies from one time level to another, i.e., $M^{n+1} \neq M^n$.

DEFINITION 2. We say a problem is of Class 2 if the number of (possibly moving) nodes required to satisfy (3.11) remains fixed from one time level to another, i.e., $M^{n+1} = M^n$.

We point out that a problem exhibiting Class 1 characteristics can be solved as if it were a Class 2 problem if enough (possibly moving) grid points are used. For example, if the nodes of a mesh that contained the largest number of points needed to equidistribute the weight function at each of the J time levels were moved appropriately, then a Class 1 problem could be solved as if it were Class 2. However, in some problems we anticipate a loss of efficiency if such a procedure were employed. Instead, it may be preferable to determine both the number and the location of the nodes at each time level (see § 4).

The moving finite element method of Miller and Miller [12] and Gelinas et al. [9], the adaptive finite element method of Davis and Flaherty [4], and the arclength

approach of White [20], for example, all attempt to move a fixed number of grid points. In our implementation we begin the calculation by applying the variable node, static-rezone approach—we assume initially the problem is of Class 1. As soon as the number of nodes required to equidistribute the weight function remains constant for several time levels, we assume the problem has shifted to a Class 2 type problem. Given M grid points at time levels n and $n-1$, we predict the location of the points at level $n+1$ by solving

$$\frac{dx_j}{dt}=\frac{x_j^n-x_j^{n-1}}{\tau^n}, \qquad j=1,2,\cdots,M-1, \tag{3.18a}$$

$$x_j(0)=x_j^{n-1}, \tag{3.18b}$$

which can be integrated to yield

$$x_j(t)=\left(\frac{x_j^n-x_j^{n-1}}{\tau^n}\right)t+x_j^{n-1}, \qquad j=1,2,\cdots,M-1. \tag{3.19}$$

In particular, for $t=t^{n+1}-t^{n-1}$, we have

$$x_j^{n+1}=\left(\frac{x_j^n-x_j^{n-1}}{\tau^n}\right)\tau^{n+1}+x_j^n, \qquad j=1,2,\cdots,M-1. \tag{3.20}$$

We see that the grid locations are linearly extrapolated from their positions at time levels n and $n-1$. We have found this approach to work well though higher order extrapolation methods are less reliable in tracking regions of high spatial activity (see also [4]).

Two problems can occur if an extrapolation method such as (3.20) is used to move the grid. First, suppose that two grid points at time levels n and $n-1$ are given as in Fig. 2. As the dotted lines indicate, the trajectories have crossed so grid point

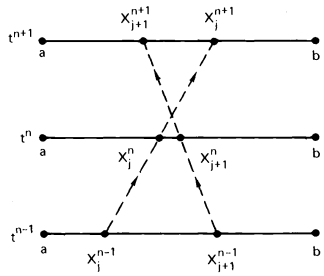


FIG. 2. *Illustration of interior grid crossing as a result of extrapolation.*

x_{j+1}^{n+1} is now positioned before point x_j^{n+1}. The difficulty is easily rectified by reordering the nodes so $x_j^{n+1}<x_{j+1}^{n+1}$ etc. The second difficulty occurs when grid points near the boundary are extrapolated out of the spatial domain as in Fig. 3. In such cases we bring the node in question back inside the region $[a, b]$. Since we anticipate its location to be close to $x_0^{n+1}=a$, we typically position the node midway between a and the next interior point. If the extrapolation procedure tends to move slightly the nodes out of the region of high spatial activity, the buffering of the mesh produced by (3.14) absorbs some of the inaccuracy of the grid points' locations. Most importantly, however, once a solution to (3.2) is obtained after a grid extrapolation, we recheck the equidistribution conditions in (3.12)-(3.13) as well as the local boundedness criteria in (3.14) to determine whether or not new nodes have to be added.

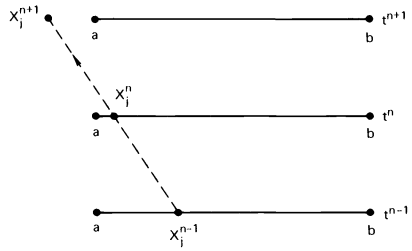


FIG. 3. *Illustration of extrapolation out of the computational domain.*

Time stepping. We choose the time steps τ^n, $n=1,2,\cdots,J$, such that the local error per unit step associated with the time differencing scheme in (3.16) is below some given value ε. The local error at the point x_j^{n+1} can be estimated by

$$\text{l.e.} \approx \alpha\frac{(\tau^{n+1})^2}{2}+2\beta h_{k+1}^n(x_j^{n+1}-x_k^n), \tag{3.21}$$

where $\alpha=\max_{t^n\leqq t\leqq t^{n+1}}|\partial^2u/\partial t^2|$ and $\beta=\max_{x_k^n\leqq x\leqq x_{n+1}^k}|\partial^2u/\partial x^2|$. The partial derivatives in α and β are approximated by finite difference expressions. For the point x_j^{n+1}, we want to determine the value of τ^{n+1} such that

$$\text{l.e.}\leqq\theta\varepsilon\tau^{n+1}, \tag{3.22}$$

where $0<\theta\leqq 1$ is a safety factor designed to account for the fact that the expression for the local error in (3.21) is only approximate. For purposes of the discussion that follows, we assume that we have an estimated value of τ^{n+1} (e.g., the last accepted time step τ^n) which we denote by τ_{old}^{n+1}. The value of τ^{n+1} that we want to calculate we denote by τ_{new}^{n+1}. Hence, if the contribution to the local error arising from the interpolation process is denoted by ε_{int}, we can write

$$\alpha\frac{(\tau_{\text{new}}^{n+1})^2}{2}\leqq\theta\varepsilon\tau_{\text{new}}^{n+1}-\varepsilon_{\text{int}}, \tag{3.23}$$

and similarly

$$\alpha\frac{(\tau_{\text{old}}^{n+1})^2}{2}\leqq\text{l.e.}-\varepsilon_{\text{int}}. \tag{3.24}$$

If we eliminate α we find

$$\left(\frac{\tau_{\text{new}}^{n+1}}{\tau_{\text{old}}^{n+1}}\right)^2\leqq\frac{\theta\varepsilon\tau_{\text{new}}^{n+1}-\varepsilon_{\text{int}}}{\text{l.e.}-\varepsilon_{\text{int}}}. \tag{3.25}$$

The above expression can be solved for the two roots of τ_{new}^{n+1}. Choosing the largest positive root, we have

$$\tau_{\text{new}}^{n+1}\leqq\frac{(\tau_{\text{old}}^{n+1})^2\theta\varepsilon}{\text{l.e.}-\varepsilon_{\text{int}}}+\sqrt{\frac{(\tau_{\text{old}}^{n+1})^4\theta^2\varepsilon^2}{(\text{l.e.}-\varepsilon_{\text{int}})^2}-\frac{4(\tau_{\text{old}}^{n+1})^2\varepsilon_{\text{int}}}{(\text{l.e.}-\varepsilon_{\text{int}})}}\Big/2, \tag{3.26}$$

with the restriction that

$$\varepsilon\geqq\frac{\sqrt{4\varepsilon_{\text{int}}(\text{l.e.}-\varepsilon_{\text{int}})}}{\tau_{\text{old}}^{n+1}\theta}. \tag{3.27}$$

This procedure is carried out for each of the $M^{n+1}-2$ interior nodes. We choose τ^{n+1} as the smallest of all the calculated values of τ_{new}^{n+1}. If at any time in the calculation

we find that we are unable to obtain a converged numerical solution to the difference equations in (3.8), we halve the time step and restart the $(n+1)$st level calculation. If after a successful calculation at the $(n+1)$st level, we find that the time step determined from (3.26) should be smaller than the value we have used, we redo the calculation at the $(n+1)$st level with the new value of τ^{n+1}. In addition, to prevent the time steps from increasing too rapidly, we impose the criteria $\tau^{n+1}/\tau^n \leqq 2$.

Several points are worth mentioning. If no interpolation error were present, we see that the expression for τ_{new}^{n+1} in (3.26) reduces to

$$\tau_{\text{new}}^{n+1} \leqq \frac{(\tau_{\text{old}}^{n+1})^2 \theta \varepsilon}{\text{l.e.}}, \tag{3.28}$$

as one would expect for the backward-Euler method. The restriction on ε in (3.27) can be attributed to the fact that part of the local error arises from the interpolation process. For the inequality in (3.25) to make sense, we must have $\theta\varepsilon\tau_{\text{new}}^{n+1} - \varepsilon_{\text{int}} \geqq 0$, which implies that the interpolation errors cannot be too large. If the expression in (3.27) is not satisfied we increase the number of spatial nodes in the manner described earlier. In practice, for the problems and the tolerances we have considered, the restriction in (3.27) presents no difficulty. We point out that the above analysis can be generalized to accommodate higher order interpolation and time discretization methods.

Computational considerations. Before concluding this section, it is worthwhile to discuss a few points concerning the numerical implementation of the method.

Numerical Jacobian. With the difference approximations we have chosen, the Jacobian matrix in (3.9) can be written in block tridiagonal form. Although we considered evaluating the Jacobian analytically, we found that in combustion problems characterized by complicated transport coefficients such a procedure was not very efficient [17]. Since our interest centers around problems of this type, we evaluate all the Jacobians by a numerical finite difference procedure. The method implements the ideas outlined by Curtis, Powell, and Reid [3]. For our problem we can compute several columns of the Jacobian simultaneously by first evaluating $\mathcal{F}$ at some vector Z. We then perturb every $3N$th element of Z beginning with the first. The function is evaluated at this new point and the approximate difference quotient found. The procedure is repeated beginning with the second element of Z and the cycle is continued until the first $3N$ elements of Z have been perturbed. As a result, we can form the entire numerical Jacobian in $3N+1$ vector function evaluations. Once the Jacobian is formed, a block tridiagonal set of linear equations must be solved. We use the block tridiagonal subroutines DECBT and SOLBT written by Hindmarsh [8].

Damping, convergence, and rezoning. The choice of the damping parameter λ_k to ensure monotonic convergence of the full Newton iteration (3.9) has been studied in depth by Deuflhard [5]. We have implemented a variation of his method for nonsingular Jacobians; however, we have found that at a given time level we almost always take full Newton steps, i.e., $\lambda_k = 1$. We terminate the Newton iteration when

$$\|\Delta V_{k+1}^{n+1}\|_2 = \|V_{k+1}^{n+1} - V_k^{n+1}\|_2 \leqq \text{TOL}\sqrt{NM^{n+1}}, \qquad k = 0, 1, \cdots, \tag{3.29}$$

where we typically take $10^{-6} \leqq \text{TOL} \leqq 10^{-9}$.

The converged numerical solution at the nth time level is ordinarily an excellent initial approximation to the solution at the $(n+1)$st level. As a result, we employ this strategy directly when the number of nodes is not changing from one time level to another (Class 2). When the problem exhibits a numerical solution that is operating

in a Class 1 mode, the number of nodes can change from one time level to another. If the solution at the nth level were used as a starting guess for the solution at the $(n+1)$st level, the converged solution at the $(n+1)$st level would contain all the nodes from the nth level plus those added at the $(n+1)$st level. This can increase rapidly the number of grid points used in a calculation—especially for problems that exhibit unstable burn front type characteristics. To alleviate this difficulty, we ordinarily use one half the number of grid points from the nth level as our starting mesh for the $(n+1)$st level. This "skeleton" mesh then serves as the initial grid from which we can determine adaptively a solution to the problem at the $(n+1)$st level. This procedure restricts the unwanted growth in the number of nodes during the variable node, static-rezone part of the calculation.

4. Numerical results. In this section we apply the fully adaptive algorithm to two unstable problems from combustion. Both problems exhibit extended periods of Class 1 behavior. In the first problem we calculate the temperature and mass fraction profiles of an unsteady propagating flame. In the second problem we consider the modeling of solid-solid alloying (aluminum and palladium) reactions. Our goal is to compare the accuracy and efficiency of the method to three other solution procedures. The first of these is a variable node, static-rezone (VNSR) approach without any grid extrapolation. In the second procedure a fixed number of nodes is used to equidistribute the arclength of the temperature (ARC) at each time level. In this formulation the independent variable x and the temperature are scaled between zero and one, and the calculation of the temperature and the grid are not coupled together. In the last method (EQUI) a fixed number of equispaced points are used for the entire calculation. We anticipate that for these problems the variable node, static-rezone approach with grid extrapolation (VNSR-X) will be the most efficient of the four considered.

Problem 1. The first problem we consider is an unsteady propagating flame with one-step chemistry and Lewis number different from unity [14]. The governing equations are

$$\frac{\partial T}{\partial t}=\frac{\partial^2 T}{\partial x^2}+R, \tag{4.1}$$

$$\frac{\partial Y}{\partial t}=\frac{1}{\mathrm{Le}}\frac{\partial^2 Y}{\partial x^2}-R, \tag{4.2}$$

where T is the normalized temperature Le is the Lewis number and Y is the normalized mass fraction of the reactant. The normalized reaction rate R is given by

$$R=\frac{\beta^2}{2\mathrm{Le}}Y\exp\left(-\frac{\beta(1-T)}{1-\alpha(1-T)}\right), \tag{4.3}$$

where β is a nondimensional activation energy and α is a nondimensional heat release term. The initial conditions are given by

$$T=\exp(x), \qquad x\leqq 0, \tag{4.4a}$$

$$Y=1-\exp(\mathrm{Le}\,x), \qquad x\leqq 0, \tag{4.4b}$$

and

$$T=1, \qquad x>0, \tag{4.4c}$$

$$Y=0, \qquad x>0, \tag{4.4d}$$

and the boundary conditions by

(4.5a) $$T = 0 \quad \text{as } x \to -\infty,$$

(4.5b) $$Y = 1 \quad \text{as } x \to -\infty,$$

and

(4.5c) $$\frac{dT}{dx} = 0 \quad \text{as } x \to \infty,$$

(4.5d) $$\frac{dY}{dx} = 0 \quad \text{as } x \to \infty.$$

We solve the problem for $\alpha = .8$, $\beta = 20.0$, and $\text{Le} = 2.0$. All of the calculations for this problem were carried out on the region $-50 \leqq x \leqq 50$. For the VNSR and VNSR-X methods the values of δ and γ in (3.12) and (3.13) and ε in (3.22) were chosen such that the numerical solutions were accurate to two significant figures. In particular, we set $\delta = \gamma = .1$ and $\varepsilon = 10^{-3}$. The value of the local boundedness constant A was set equal to 2.5, and both calculations had initially five equispaced subintervals.

Our numerical results reveal that the temperature and species profiles and the velocity of propagation oscillate as a function of time. In Fig. 4 we illustrate the calculated temperature profile for the VNSR-X method during a typical oscillation (the VNSR calculation gave comparable results). We observe that the peak temperature changes 10–15% from its initial maximum value. It is during the period of temperature rise that the velocity of the flame increases substantially. In Fig. 5 we illustrate the velocity of propagation as a function of time for the VNSR-X method. We notice that the velocity of the flame oscillates with a period of about ten seconds and the ratio of the peak to minimum flame speed is approximately equal to six. During the period of

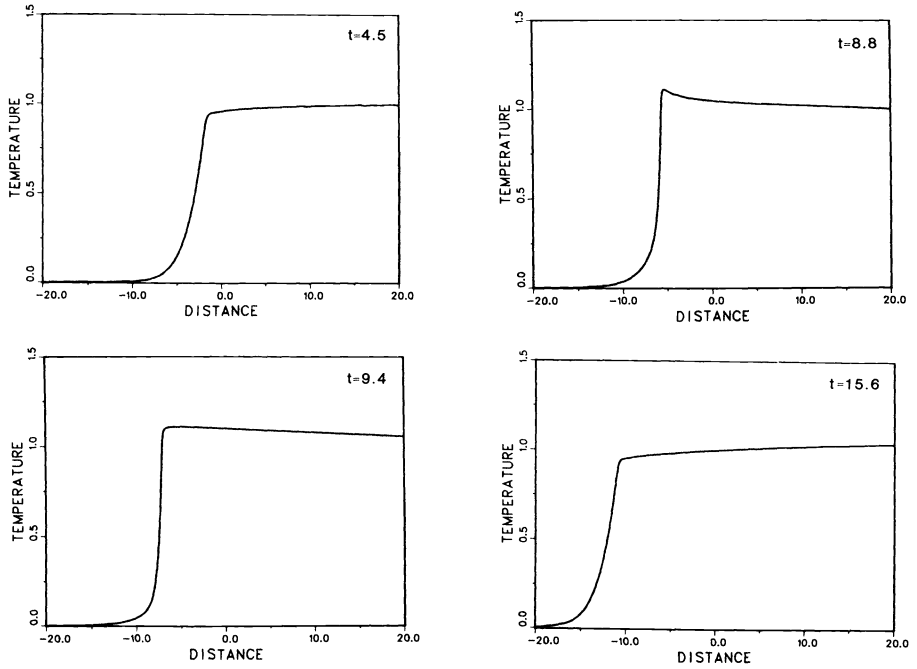


FIG. 4. *Calculated temperature profiles* (VNSR-X) *during an oscillation of the unsteady propagating flame.*

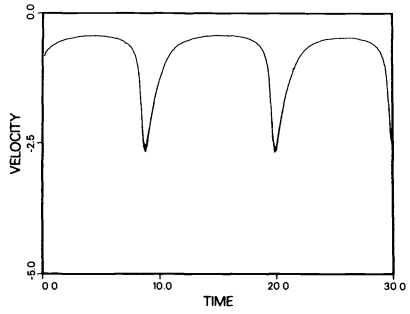


FIG. 5. *Calculated velocity of propagation* (VNSR-X) *for the unsteady propagating flame.*

velocity increase, the solution exhibits Class 1 behavior. Both the VNSR and VNSR-X methods increase the number of grid points needed to satisfy (3.12)–(3.14). As the solution propagates with a more constant velocity (Class 2) the number of grid points used in both calculations remains fairly constant. The VNSR-X calculation used a maximum of 206 grid points and a minimum of 103 while the VNSR calculation used a maximum of 150 points with a minimum of 101. The difference in the maximum number of points for the two methods stems from the fact that at some time during the calculation the extrapolation procedure moved the grid out of the flame zone and extra points had to be added to satisfy (3.12)–(3.14). To be assured that the profile in Fig. 5 represented a "reference" solution, we performed VNSR and VNSR-X calculations with up to three times the number of adaptive grid points used to produce the results in Fig. 5. No observable differences in the velocity profiles were found.

We next ran the calculation for the equispaced and arclength procedures. In both calculations we used 150 points—the maximum number required by the VNSR approach. In Figs. 6 and 7 we illustrate the temperature profiles for the EQUI and ARC calculations respectively. The profiles were taken at the same times as the profiles in the VNSR-X calculation. The resolution of the flame front is poorer, and we observe that the position of the flame is shifted from the position of the front in Fig. 4.

Although the temperature is an important measurable quantity, the velocity of propagation is also a quantity of physical significance. It enables one to determine the time for a flame to propagate a fixed distance. This is important in a variety of combustion applications. As a result, we performed equispaced and arclength calculations with an increasing number of grid points in an attempt to match the velocity profile of the VNSR-X calculation. In Fig. 8 we illustrate the velocity of propagation as a function of time for the EQUI calculation with 150, 250, 500 and 4000 points. We obtain a very "noisy" profile for 150 and 250 points. The velocity oscillates with short periods and small amplitude variations. It was not until we used 500 points that we started to see behavior that remotely resembled the results in Fig. 5. We continued the calculation on grids with 1000, 2000 and 3000 points. It was not until we used an equispaced grid with 4000 points that we were able to obtain results within 5% of both the period and peak amplitude of the VNSR-X calculation. In Fig. 9 we illustrate a similar set of calculations for the arclength method with 150, 250, 500 and 3000 points. In the calculations with 150 and 250 points we obtain velocity profiles that oscillate with large amplitude variations and much smaller periods than the VNSR-X calculation. However, as the number of nodes increases, we gradually obtain results similar to those in Fig. 5. It was not until we used 3000 arclength points that we were able to obtain velocity profiles in which both the period and peak amplitude of the oscillation were within 5% of the values of the profile in Fig. 5.

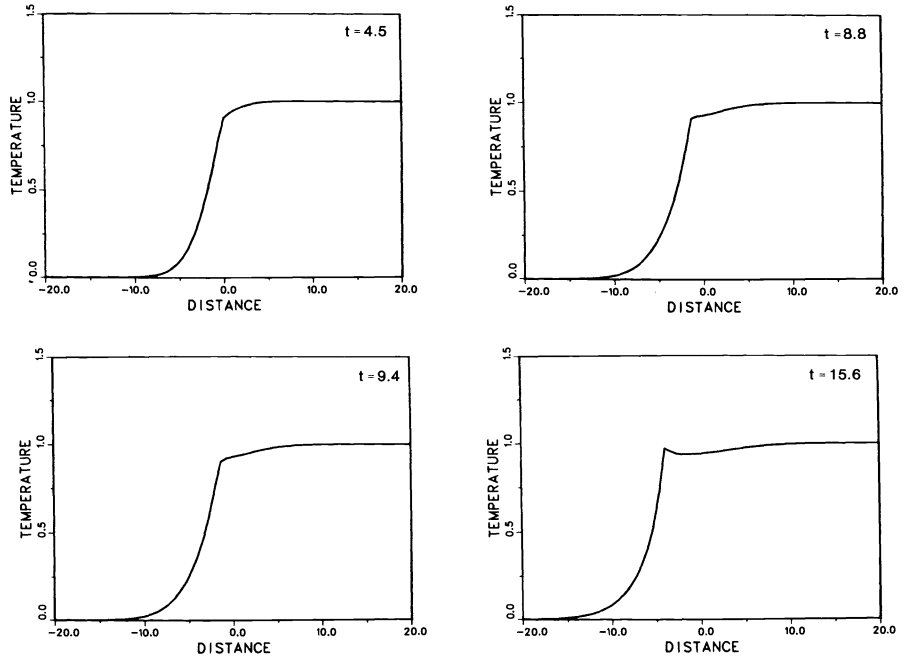


FIG. 6. *Calculated temperature profiles for the unsteady propagating flame. The calculation was performed with* 150 *equispaced points* (EQUI).

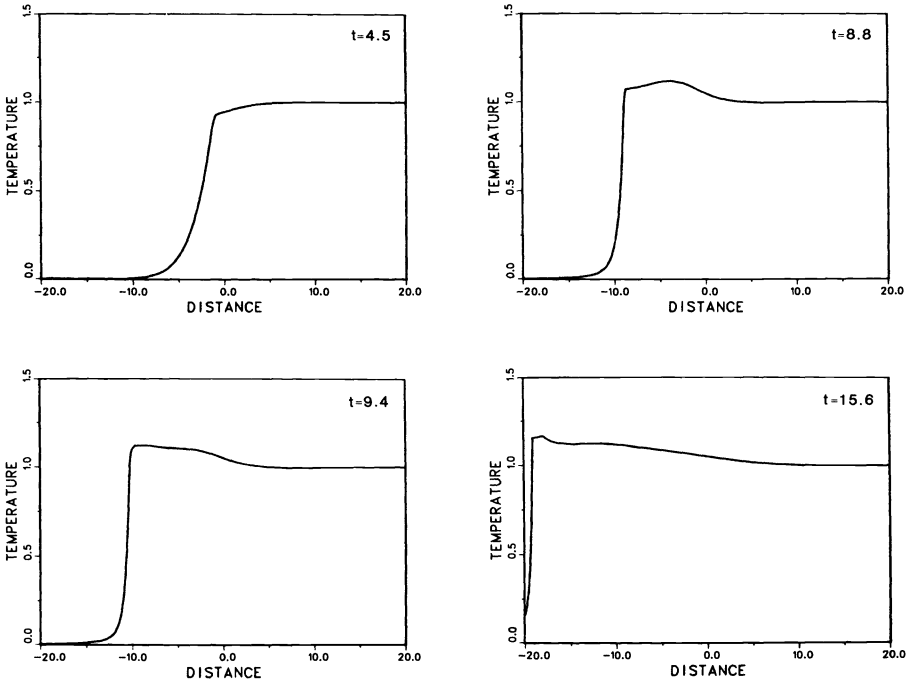


FIG. 7. *Calculated temperature profiles for the unsteady propagating flame. The calculation was performed with* 150 *equi-arclength points* (ARC).

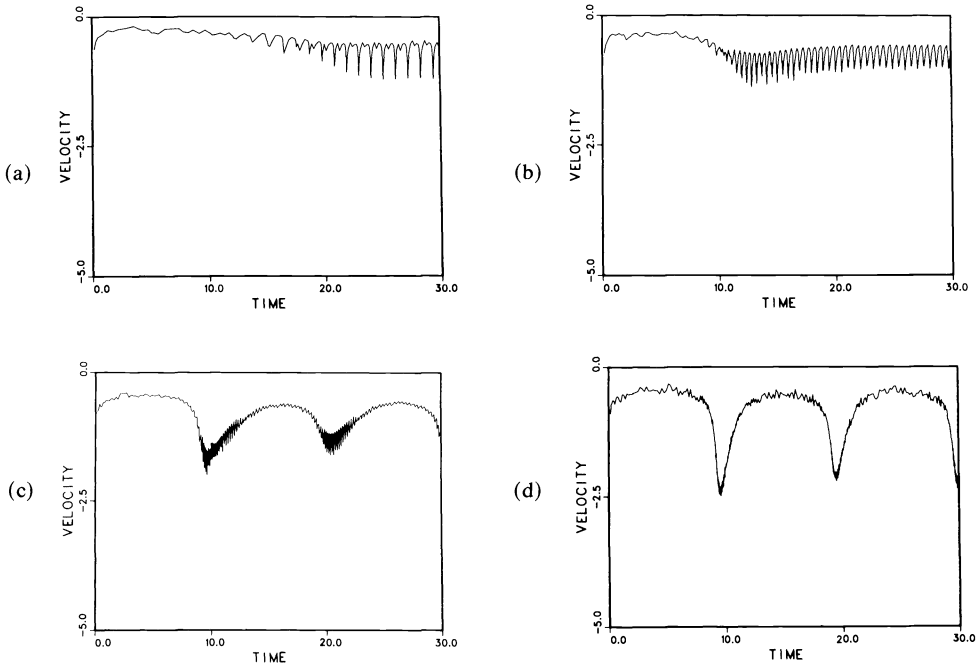


FIG. 8. *Calculated velocities of propagation for the unsteady propagating flame. The calculations were performed with* 150(a), 250(b), 500(c), *and* 4000(d) *equispaced points* (EQUI).

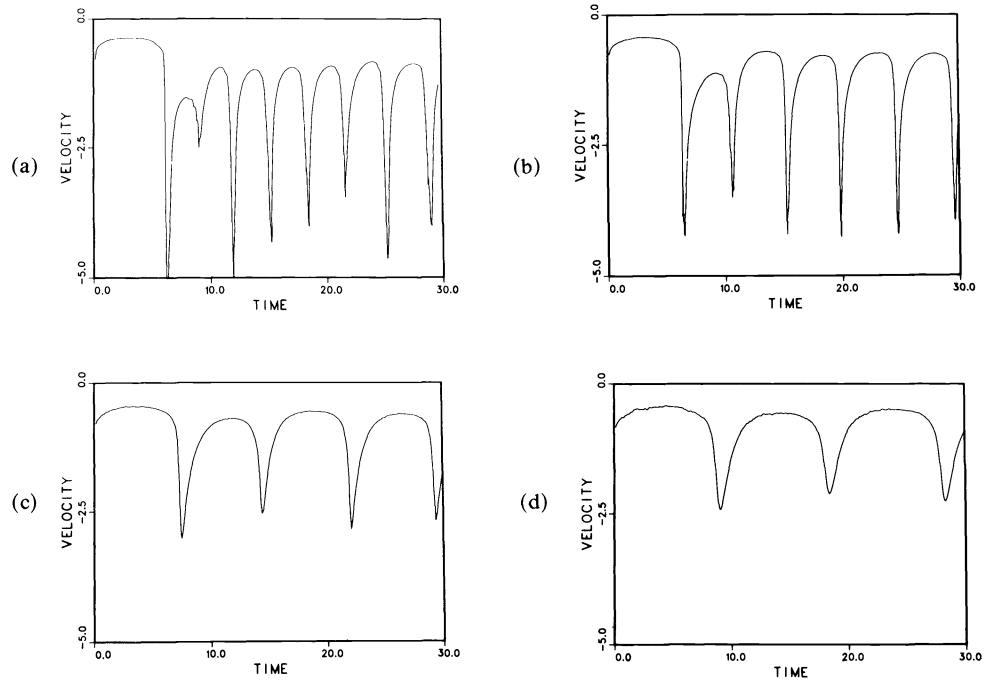


FIG. 9. *Calculated velocities of propagation for the unsteady propagating flame. The calculations were performed with* 150(a), 250(b), 500(c), *and* 3000(d) *equi-arclength points* (ARC).

If we normalize to one the CPU time for a velocity oscillation of the VNSR-X calculation, then a comparison of the cost of an oscillation for the other three methods (we use the 4000 EQUI and 3000 ARC calculations) is given in Table 1. We also include the maximum and minimum number of nodes used in each calculation. All four calculations were run with TOL $= 10^{-8}$. This problem exhibits both Class 1 and Class 2 behavior and we observe that the VNSR-X approach is the most efficient of the four considered.

TABLE 1
Computational statistics for the unstable freely propagating flame.

Method	CPU	Max	Min
VNSR-X	1.0	206	103
VNSR	2.3	150	101
ARC	3.6	3000	3000
EQUI	4.2	4000	4000

Several points are worth discussing. Based upon the difference in the number of grid points between the arclength, equispaced, and VNSR-X calculations, we might expect an increase in efficiency of the VNSR-X method over the ARC and EQUI methods by a factor of 10–20. However, when the solution exhibits Class 1 behavior, the VNSR and VNSR-X methods adjust the number of grid points at each time level. This requires building up a solution from a skeleton grid. As a result, we generally form one Jacobian on the skeleton grid and another one on the finest grid. When these results are combined with the facts that approximately 50% of the cost of the VNSR-X calculation occurred when the solution was propagating in a Class 1 mode, and that the Jacobian in the ARC and EQUI calculations was often used for as many as ten time steps, it is clear why we obtain a factor of 3–4 over the fixed node methods. If the solution to this problem were such that the increase in spatial activity occurred for only a short period of time, the cost of the variable node methods would decrease compared to the fixed node methods. The cost of the VNSR and VNSR-X methods would be lower since we would only have to propagate the maximum number of nodes (150 and 206 points respectively) for a small time interval. If the solution propagated with extended periods of Class 2 behavior (without any large changes in spatial activity), the VNSR method would still form Jacobians at each time level. Assuming that fewer than 3000 equi-arclength and 4000 equispaced points could be used, we would expect the efficiency of the fixed node methods to increase with respect to the VNSR approach. The VNSR-X method would still be competitive, however. Finally, if the solution continually propagated in a Class 1 mode, the efficiency of the variable node methods compared to the fixed node methods would be similar to the results in Table 1 for the VNSR and fixed node methods—the VNSR-X and VNSR methods would be the same.

Problem 2. The last problem we consider involves the modeling of solid-solid diffusion controlled alloying reactions (see e.g., [7]). Such problems are important in pyrotechnic applications. The model assumes we have two materials in powder form—A and B (e.g., aluminum and palladium)—having different radii. At a given temperature material A melts and it starts to coat B. As A diffuses into B chemical energy is released and a reaction front begins to propagate. We model this process by a nonlinear energy equation and a time-dependent source term. The governing equations are

$$\rho c \frac{\partial T}{\partial t} = \frac{\partial}{\partial x}\left(k \frac{\partial T}{\partial x}\right) + \Delta h \frac{\partial f}{\partial t}, \tag{4.6}$$

$$\frac{\partial f}{\partial t}=\frac{D\exp(-E/T)}{R^2}\,\frac{(1-f)^{1/3}}{1-(1-f)^{1/3}}, \tag{4.7}$$

where T is the temperature and f is the fraction of the product formed. The initial conditions are given by

$$T(x,0)=300, \tag{4.8a}$$

$$f(x,0)=10^{-3}, \tag{4.8b}$$

and the boundary conditions for $t\leqq .05$ by

$$T(0,t)=3.4\times 10^4 t+300, \tag{4.9a}$$

$$\frac{\partial T}{\partial x}(L,t)=0, \tag{4.9b}$$

and for $t>.05$ by

$$\frac{\partial T}{\partial x}(0,t)=0, \tag{4.10a}$$

$$\frac{\partial T}{\partial x}(L,t)=0. \tag{4.10b}$$

In addition to the quantities already defined, k, denotes the thermal conductivity; ρ, the density; c, the specific heat; D, a diffusion rate; E, the activation energy; R, the radius of the unmelted particle and Δh, the heat of reaction. In this problem heat is added at the left boundary until the temperature reaches 2000 K. The heat source is then turned off and the reaction begins to propagate. The length of the domain L was 5 cm., and we performed the calculation with the same values of the tolerances used in Problem 1.

Due to nonuniform packing densities, nonuniform particle sizes and material impurities, the peak temperature and the velocity of propagation can vary as a function of time. The velocity of propagation can also vary due to a pulsating instability similar to the one in the gaseous case considered in Problem 1. This has been investigated, for example, by Margolis [11]. In this problem, however, we consider variations in the temperature and velocity profiles due strictly to mixture variations. To simulate such behavior we pack a one-dimensional reactor with sections having different material properties. While we know a priori where these sections are located, in a realistic problem they can be randomly distributed throughout the reactor. In the sections between $.25\leqq x\leqq .375$ cm. the particle size decreases linearly by a factor of two, and the heat release term increases by 30%. Both quantities then increase linearly (until $x=.5$ cm.) to their original values. These striations are then repeated in the regions beginning at $x=.75$, 1.25, 1.75, $\cdots$, 4.25, and 4.75 cm.

As our calculations indicate, once the burn front enters these sections, the velocity of the front begins to increase. In addition, due to the larger heat release term, the peak temperature increases as well. As the additional heat pulse develops, the VNSR and VNSR-X methods increase rapidly the number of grid points needed to satisfy (3.12)-(3.14). We performed both calculations with an initial equispaced grid consisting of 50 mesh intervals. The VNSR calculation used a maximum of 362 and a minimum of 57 points. Similar results were observed for the VNSR-X calculation in which a maximum of 437 and a minimum of 65 points were used. In Fig. 10 we illustrate the temperature profile for the VNSR-X calculation as the front emerges from the first

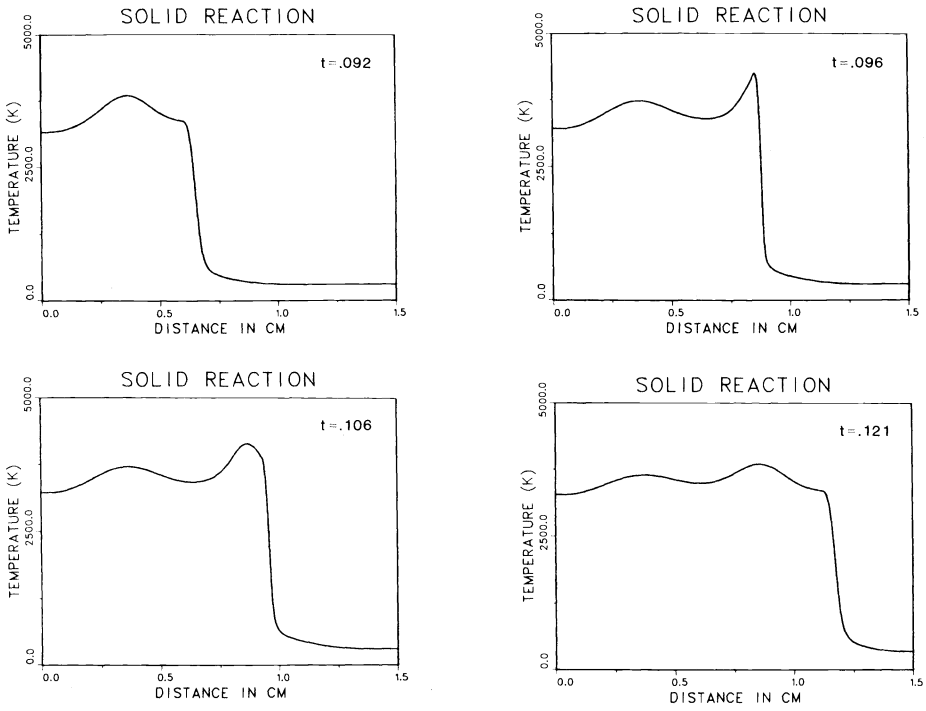


FIG. 10. *Calculated temperature profiles* (VNSR-X) *for the solid-solid alloying problem.*

striation and as it propagates through the second nonuniform zone. We see that the peak temperature increases by as much as 1500 K. In Fig. 11 we illustrate the velocity of the front from the time the heat source is turned off ($t = .05$ seconds) until it reaches the beginning of the second striation (approximately .04 seconds later). As the figure illustrates, the velocity increases by a factor of four over its uniform material value.

We next performed the same type of grid point study as in Problem 1. We ran the problem with the EQUI and ARC methods. For both the equispaced and arclength calculations with 150, 250, 500, 1000, and 1500 points, the reaction "went out" after the heat source was turned off. Heat diffused away faster than it was produced. It was not until we used 2000 arclength points and 4000 equispaced points that we were able to propagate a reaction with adequate resolution of the temperature profile. The velocities of propagation, however, (Figs. 12a and 12b) differ dramatically from the VNSR-X calculation. We notice immediately the very noisy profiles, and, for the equispaced calculation, the total lack of resolution of the velocity spike. It is worthwhile to point out that this same behavior was observed in our initial calculations with these methods for test Problem 1. By lowering the Newton tolerance from 10^{-6} to 10^{-8}, however, we were able to eliminate much of the noise in the velocity profiles (see Figs. 8d and 9d). The noisy profiles were observed not only on the finest grids but on the 150 and 250 point grids as well. The VNSR and VNSR-X calculations for Problem 1, however, were free from noise for TOL $= 10^{-6}$.

By requiring a stricter termination criteria for Newton's method, more modified Newton iterations are performed per time step than if a less severe criterion is used. As a result, since the CPU times listed in Table 1 were for TOL $= 10^{-8}$, the efficiency of the VNSR and VNSR-X methods compared to the ARC and EQUI methods is actually greater than that reported. In this test problem the calculations that produced

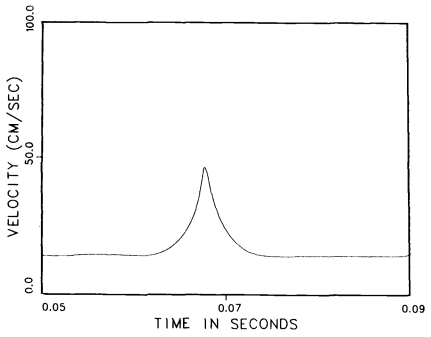


FIG. 11. *Calculated velocity of propagation* (VNSR-X) *for the solid-solid alloying problem.*

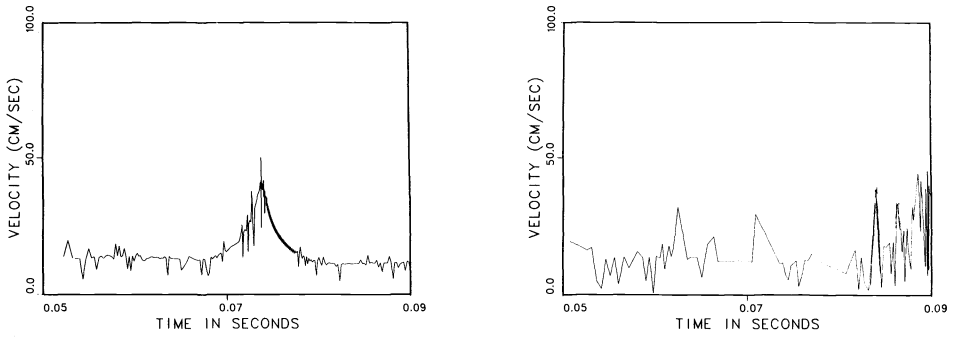


FIG. 12. *Calculated velocities of propagation for the solid-solid alloying problem with* TOL $= 10^{-6}$. *The calculations were performed with* (a) 2000 *equi-arclength points and* (b) 4000 *equispaced points.*

the results in Figs. 11 and 12 were performed with TOL $= 10^{-6}$. If we run the EQUI and ARC calculations with a stricter Newton tolerance, we can reduce the noise in the velocity profiles (see Figs. 13a and 13b). We found that a tolerance of 10^{-9} for both the ARC and EQUI methods was sufficient. The peak velocities and the time that the peaks occur are within 10% of the values for the profile in Fig. 11. As one would expect, however, the stricter Newton tolerance increases the CPU time over the 10^{-6} calculations.

If we normalize to one the average CPU time required for the VNSR-X method to propagate a solution from one striation to the next, then a comparison of the cost of the calculation for the other three methods is given in Table 2. All reported CPU

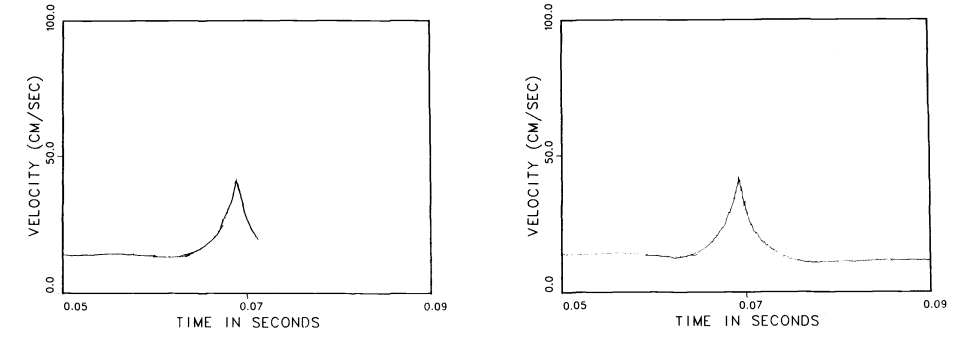


FIG. 13. *Calculated velocities of propagation for the solid-solid alloying problem with* TOL $= 10^{-9}$. *The calculations were performed with* (a) 2000 *equi-arclength points and* (b) 4000 *equispaced points.*

TABLE 2.
Computational statistics for the solid-solid alloying problem.

Method	CPU	Max	Min
VNSR-X	1.0	437	65
VNSR	1.9	362	57
ARC	1.8	2000	2000
EQUI	3.3	4000	4000

times are for TOL = 10^{-9}. Again, due to the difference in the required Newton tolerances, the efficiency of the VNSR-X and VNSR methods compared to the ARC and EQUI methods is actually greater than that reported. This problem exhibits Class 1 and Class 2 2 behavior, and we again observe that the VNSR-X approach is the most efficient of the four considered.

5. Summary. We have developed a fully adaptive method for solving one-dimensional mixed initial-boundary value problems with particular emphasis on unstable problems from combustion. The method adaptively adjusts the number of grid points needed to equidistribute a positive weight function over a given mesh interval at each time level. In addition, when the solution can be described accurately by a fixed number of points, the mesh is moved by extrapolating the nodal positions from earlier time levels. We monitor the solution from one time level to another to insure that the local error per unit step associated with the time differencing method is below some specified tolerance. We have applied the method to several problems of physical interest.

Both of the test problems considered exhibited extended periods of Class 1 behavior. As a result, the VNSR and VNSR-X methods frequently adjusted the number and the location of the grid points required to satisfy (3.12)-(3.14). In addition, when the solutions to both problems were propagating in a Class 2 mode, the VNSR-X method kept the number of grid points fairly constant. In both problems we found that a combination of static-rezoning and grid propagation was the most efficient of the four methods considered. Although we have not reported any results for problems that exhibit extended periods of Class 2 behavior (e.g. a stable burn front), we have performed a number of such calculations. As one would expect, the VNSR method is the least efficient. Formation and factorization of the Jacobian at each time step is far too costly to make the VNSR method competitive. However, the VNSR-X and ARC methods are competitive as a result of their ability to use a Jacobian for several time steps. Both methods are significantly more efficient than equispaced calculations.

Our results suggest that when a solution propagates with "large" changes in its spatial activity (steeper fronts, additional peaks etc.), methods that attempt to adjust the number and the location of the grid points (VNSR-X) can be more efficient than methods that employ a fixed number of moving nodes (ARC). However, when a solution propagates without any large change in the regions of high spatial activity, methods that attempt to move a fixed number of grid points at each time level (VNSR-X and ARC) are to be preferred over methods that employ a totally variable node approach (VNSR). In addition, any of the adaptive techniques considered in this paper are to be preferred over an equispaced calculation. Finally, while the method developed in this paper is important for one-dimensional unstable problems, it is essential in two-dimensional calculations. The savings in CPU time and central memory that can result from applying adaptive methods to two-dimensional problems can be the deciding

factors as to whether a problem is computationally feasible. In a future paper we will present the results of generalizing the VNSR-X method to two dimensions.

Acknowledgments. The authors would like to thank Drs. J. Grcar and S. Margolis for their helpful discussions and their important comments and suggestions concerning this material.

REFERENCES

[1] U. ASCHER, J. CHRISTIANSEN AND R. D. RUSSELL, COLSYS—*A collocation code for boundary value problems*, Proc. Conference for Working Codes for Boundary Value Problems in ODE's, B. Childs et al., eds., Lecture Notes in Computer Science 176, Springer-Verlag, New York, 1979.

[2] J. H. BOLSTAD, *An adaptive finite difference method for hyperbolic systems in one space dimension*, Ph.D. Thesis, Stanford Univ., Stanford, CA, 1982.

[3] A. R. CURTIS, M. J. POWELL AND J. K. REID, *On the estimation of sparse Jacobian matrices*, J. Inst. Math. Appl., 13 (1974), pp. 117–119.

[4] S. F. DAVIS AND J. E. FLAHERTY, *An adaptive finite element method for initial-boundary value problems for partial differential equations*, this Journal, 3 (1982), pp. 6–27.

[5] P. DEUFLHARD, *A modified Newton method for the solution of ill-conditioned systems of nonlinear equations with application to multiple shooting*, Numer. Math., 22 (1974), pp. 289–315.

[6] H. A. DWYER, R. J. KEE AND B. R. SANDERS, *Adaptive grid methods for problems in fluid mechanics and heat transfer*, AIAA J., 18 (1980), pp. 1205–1212.

[7] A. P. HARDT AND P. V. PHUNG, *Propagation of gasless reactions in Solids-I. Analytical study of exothermic intermetallic reaction rates*, Comb. and Flame, 21 (1973), pp. 77–89.

[8] A. C. HINDMARSH, *Solution of block tridiagonal systems of linear equations*, Lawrence Livermore Laboratory Report UCID-30150, Livermore, CA, 1977.

[9] R. J. GELINAS, S. K. DOSS AND K. MILLER, *The moving finite element method: application to general partial differential equations with multiple large gradients*, J. Comp. Phys., 40 (1981), pp. 202–249.

[10] J. KAUTSKY AND N. K. NICHOLS, *Equidistributing meshes with constraints*, this Journal, 1 (1980), pp. 499–511.

[11] S. B. MARGOLIS, *An asymptotic theory of condensed two-phase flame propagation*, SIAM J. Appl. Math., 43 (1983), to appear.

[12] K. MILLER AND R. MILLER, *Moving finite elements*, I, SIAM J. Numer. Anal., 18 (1981), pp. 1019–1032.

[13] V. PEREYRA AND E. G. SEWELL, *Mesh selection for discrete solution of boundary value problems in ordinary differential equations*, Numer. Math., 23 (1975), pp. 261–268.

[14] N. PETERS AND J. WARNATZ, eds., *Numerical Methods in Laminar Flame Propagation*, Vieweg, Wiesbaden, 1982.

[15] R. D. RUSSELL, *Mesh selection methods*, Proc. Conference for Working Codes for Boundary Value Problems in ODE's, B. Childs et al., eds., Lecture Notes in Computer Science 176, Springer-Verlag, New York, 1979.

[16] M. D. SMOOKE, *An error estimate for the modified Newton method with applications to the solution of nonlinear two-point boundary value problems*, J. Optim. Theory Appl., 39 (1983), pp. 489–511.

[17] ———, *Solution of burner-stabilized pre-mixed laminar flames by boundary value methods*, J. Comp. Phys., 48 (1982), pp. 72–105.

[18] W. M. TSCHARNUTER AND K. H. WINKLER, *A method for computing selfgravitating gas flows with radiation*, Comp. Phys. Comm., 18 (1979), pp. 171–199.

[19] A. B. WHITE, *On selection of equidistributing meshes for two-point boundary value problems*, SIAM J. Numer. Anal., 16 (1979), pp. 472–502.

[20] ———, *On the numerical solution of initial/boundary-value problems in one space dimension*, SIAM J. Numer. Anal., 19 (1982), pp. 683–697.

SIAM J. SCI. STAT. COMPUT.
Vol. 7, No. 1, January 1986

021

INTERPOLATION SCHEMES FOR COLLOCATION SOLUTIONS OF TWO POINT BOUNDARY VALUE PROBLEMS*

STEVEN PRUESS†

Abstract. Several interpolatory approximations for collocation solutions to systems of two point boundary value problems are studied. These interpolate superconvergent values are produced by the usual Gauss-Legendre collocation algorithm. Comparisons are made to the collocation solution in terms of accuracy, storage requirements and number of operations; numerical examples are given which illustrate typical behavior.

Key words. spline interpolation, collocation for mixed order systems, stability of spline interpolation, two-point boundary value problems

1. Introduction. Numerical approximations to solutions of two point boundary value problems are often computed only on a discrete set of mesh points; some type of interpolation is necessary if solutions are desired at points not in the mesh. This is not the case for collocation as implemented in COLSYS of Ascher, Christiansen and Russell [2], since closed form approximations in terms of B-splines are available. However, the findings of Ascher, Pruess and Russell [3] indicate that there are computational advantages in using local monomial representations which admit savings in time and space when only mesh point values are computed. In addition, the error estimates for collocation are asymptotically smaller (superconvergent) at the mesh points than at nonmesh points. This paper examines several interpolation schemes and compares them to each other and to the collocation solution in terms of efficiency (time and space) and accuracy (theoretical and actual). The paper is written with collocation in mind, but should be relevant to other discretization methods.

As in COLSYS [2], systems of mixed order ordinary differential equations are considered. For simplicity, only linear equations are treated, though one of the major needs for interpolation schemes (when only mesh values are available) is in solving nonlinear systems by linearization. We write (D denotes d/dx)

$$D^{m(l)}u_l = \sum_{j=1}^{p} \sum_{i=1}^{m(j)} C_{lij} D^{i-1} u_j + f_l, \qquad 1 \leqq l \leqq p \tag{1.1}$$

with $\{C_{lij}\}$ and $\{f_l\}$ the given coefficient functions over some interval $[a, b]$, and $\mathbf{u} = (u_l)$ to be determined. With

$$m^* = \sum_{j=1}^{p} m(j), \tag{1.2}$$

m^* boundary conditions also must be given. In the program used to generate the numerical results, the boundary conditions were assumed to be linear and separated [9, § 1.2], but the exact form is not critical for the discussions of §§ 2–4.

For the remainder of the paper, in order not to complicate mathematical statements, it is assumed that (1) the coefficient functions (hence, the solutions) are sufficiently smooth that all indicated derivatives exist and (2) there is a unique solution to (1.1), i.e., a Green's matrix exists for the problem with homogeneous boundary conditions.

* Received by the editors November 23, 1983, and in final form September 27, 1984.

† Department of Mathematics and Statistics, University of New Mexico, Albuquerque, New Mexico 87131.

Three types of interpolation schemes are considered, as well as the algorithm for computing the collocation solution (using a local monomial representation). All three use the superconvergent mesh point approximations and differ only in how remaining approximations (if needed) are generated. The first uses the differential equation and possibly its derivatives, while the second uses superconvergent values from neighboring intervals. The third differs from the second in that a different polynomial is used for each desired derivative approximation. In terms of efficiency, for a fixed mesh some of the alternative algorithms are superior to the full collocation solution. However, in terms of accuracy this may not be the case: the alternative algorithms may have difficulties on highly nonuniform meshes or on some singular perturbation problems. Details are given below.

A brief outline of the paper is: Section 2 contains known results about the error of the collocation solution and new results on stability of piecewise polynomial interpolants. Since the numerical algorithm for collocating (1.1) based on a local monomial representation has not appeared elsewhere, this is briefly derived in § 3. The final section presents numerical results and conclusions.

2. Mathematical preliminaries. Notation similar to that in [1], [3] is used in describing the collocation approximation. The space of all polynomials of order M (degree $< M$) is denoted by P_M. Set $\Delta = \{a = x_1 < x_2 < \cdots < x_{N+1} = b\}$, $h_j = x_{j+1} - x_j$, $h = \max h_j$. Sometimes it will be necessary to restrict partitions to that subset of all partitions having bounded local mesh ratios, i.e., there is a constant $R \geqq 1$ such that for any Δ in the subset

$$\frac{1}{R} \leqq \frac{h_j}{h_{j-1}} \leqq R \quad \text{for all } j. \tag{2.1}$$

By $P_{\Delta,M}$ is meant the space of piecewise polynomials of order M with breakpoints in Δ.

To estimate solutions of (1.1) for some choice of $k(l)$ we seek $\hat{u}_l$ in $P_{m(l)+k(l),\Delta} \cap C^{m(l)-1}[a, b]$ which satisfies the m^* boundary conditions and collocation conditions (the differential equation is satisfied exactly on some finite set of points called the collocation points). Since $\{\hat{u}_l\}_{l=1}^p$ together involve $N \sum_{j=1}^p k(j) + m^*$ free parameters, the total number of collocations per interval must be $\sum_{j=1}^p k(j)$. In order to use the same collocation points for each component we require $k(l) = k$ for all l. Various authors have established convergence results for collocation methods applied to various types of differential equations. The standard reference is de Boor and Swartz [7] where nonlinear scalar mth order problems are considered. Cerutti [5] treats nonlinear mixed order systems, but the rates of convergence are not sharp in the superconvergent case. Houstis [8] has the correct rates, but assumes the order m_j is the same for all components and requires some extra unnecessary assumptions. When the collocation points are Gauss-Legendre points, these results can be summarized in

THEOREM 1 [1]. *For any partition Δ with h sufficiently small, assume $k \geqq m(l)$ for all l, then there exist unique $\{\hat{u}_l\}_{l=1}^p$, $\hat{u}_l \in P_{k+m(l),\Delta} \cap C^{m(l)-1}[a, b]$, satisfying the kpN collocation conditions and the m^* boundary conditions. Moreover, for $1 \leqq l \leqq p$*

$$\text{for } x \text{ in } \Delta \quad |D^{i-1}u_l(x) - D^{i-1}\hat{u}_l(x)| \leqq Ch^{2k}, \qquad 1 \leqq i \leqq m(l), \tag{2.2}$$

$$\text{for } x \text{ not in } \Delta \quad |D^{i-1}\hat{u}_l(x) - D^{i-1}u_l(x)| \leqq Ch^{m_l+k-i+1}, \qquad 1 \leqq i \leqq m(l)+1. \tag{2.3}$$

Here, and in what follows, C is a generic constant which may depend on k, the solution u and its derivatives or $\{m(l)\}_{l=1}^{p}$, but not on the partition Δ. Henceforth, we also assume

$$k \geqq m(l), \qquad 1 \leqq l \leqq p, \tag{2.4}$$

and h is sufficiently small so that (2.2), (2.3) hold; clearly, superconvergence occurs only when strict inequality prevails in (2.4).

The piecewise polynomial approximations to $\hat{\mathbf{u}}$ are described in terms of linear projectors, i.e., linear idempotent maps. Interpolation polynomials will be defined on a standard interval $[-\nu, 1+\nu]$ where ν is a nonnegative integer which depends on the order of $\hat{\mathbf{u}}$ and/or the order of the differential equation, but not on the partition Δ. This standard interval is then mapped into the appropriate piece of Δ by a change of variables. In detail, let $S_{n\nu}$ be the piecewise linear change of variable map from $C[x_{n-\nu}, x_{n+1+\nu}]$ to $C[-\nu, 1+\nu]$, i.e.,

$$(S_{n\nu}g)(t) = g((t-j)h_{n+j} + x_{n+j}), \; t \in [j, j+1] \tag{2.5}$$

$j = -\nu, -\nu+1, \cdots, \nu$. For an interpolating set in $[-\nu, 1+\nu]$ containing the distinct points $\{\zeta_j\}_{j=1}^{d}$ with respective multiplicities $\{\mu_j\}_{j=1}^{d}$, set $M = \sum_{j=1}^{d} \mu_j$, and for any G in $C^M[-\nu, 1+\nu]$ define PG to be that polynomial in P_M which interpolates G over this set. Thus,

$$(D^{i-1}PG)(\zeta_j) = (D^{i-1}G)(\zeta_j), \qquad 1 \leqq i \leqq \mu_j, \quad 1 \leqq j \leqq d. \tag{2.6}$$

Finally, define the interpolatory map P_Δ from $C^M[a, b]$ onto $P_{\Delta,M}$ by

$$P_\Delta g = S_{n\nu}^{-1} P S_{n\nu} g, \qquad x \in [x_n, x_{n+1}], \qquad 1 \leqq n \leqq N. \tag{2.7}$$

Since $[0, 1]$ gets mapped into $[x_n, x_{n+1}]$ by $S_{n\nu}$, in order for $P_\Delta g$ to be well defined at points of Δ, it is assumed that 0 and 1 are in the interpolating set $\{\zeta_j\}$ and have the same multiplicity. For the applications considered here, 0 and 1 will have multiplicity at least m, the order of the differential equation for the component under consideration, so $P_\Delta g \in C^{m-1}[a, b]$ in all cases.

If $\zeta_j \in [i(j), i(j)+1]$ for some $i(j)$, then the interpolation conditions (2.6) can also be written

$$(D^{l-1}P_\Delta g)(\eta_{nj}) = (D^{l-1}g)(\eta_{nj}), \qquad 1 \leqq l \leqq \mu_j \tag{2.8}$$

with $\eta_{nj} = (\zeta_j - i_j)h_{n+i(j)} + x_{n+i(j)}$; $1 \leqq j \leqq d$, $1 \leqq n \leqq N$.

Error bounds for the case $\nu = 0$, are well known (for special cases see de Boor [6], Schultz [11], or, in general Pruess [10]); but for $\nu > 0$, there does not appear to be one source which treats errors in the generality given here.

THEOREM 2. *If $g \in C^M[a, b]$ then there exists a constant C such that for any Δ with bounded local mesh ratios* (2.1), $1 \leqq i \leqq M$

$$\max_{[x_n, x_{n+1}]} |D^{i-1}(g - P_\Delta g) \leqq C \cdot R^{M\nu} h_n^{M-i+1} \max_{[x_{n-\nu}, x_{n+\nu+1}]} |D^M g|. \tag{2.9}$$

Moreover, if $\nu = 0$ then the assumption of bounded local mesh ratios can be omitted.

Proof. For $x \in [x_n, x_{n+1}]$

$$\begin{aligned} |D^{i-1}(g - P_\Delta g)| &= |D^{i-1}S_{n\nu}^{-1}S_{n\nu}g - D^{i-1}S_{n\nu}^{-1}PS_{n\nu}g| \\ &= |S_{n\nu}^{-1}D^{i-1}(1-P)S_{n\nu}g|/h_n^{i-1} \leqq |D^{i-1}(1-P)S_{n\nu}g|/h_n^{i-1}. \end{aligned}$$

But $D^{i-1}P$ induces a projector on P_{M-i+1}, as described in [10], so by Lebesgue's

inequality

$$\begin{aligned}|D^{i-1}(g-P_\Delta g)| &\leqq C \operatorname{dist}(D^{i-1}(S_{n\nu}g), P_{M-i+1})/h_n^{i-1}\\ &\leqq C \sup_{[-\nu,1+\nu]} |D^M(S_{n\nu}g)|/h_n^{i-1}\\ &\leqq C \max_{n-\nu\leqq j\leqq n+\nu} (h_j^M \max_{[x_j,x_{j+1}]} |D^M g|)/h_n^{i-1}\\ &\leqq CR^{M\nu}h_n^{M-i+1} \max_{[x_{n-\nu},x_{n+\nu+1}]} |D^M g|.\end{aligned}$$

Also, note that if $\nu=0$ then $\max_{n-\nu\leqq j\leqq n+\nu} h_j^M = h_n^M$, so the factor R does not arise.

In this paper the main concern is in extending such bounds to the case when only approximate data are used for the interpolation conditions (2.8). Stability results have been established for smooth spline approximations (e.g., Swartz and Varga [12]) and some local ones (Birkhoff and de Boor [4]) but not for arbitrary meshes in the generality needed here.

Define a projector $\hat{P}_\Delta$ from $C[a,b]$ onto $P_{\Delta,M}$ which satisfies (instead of (2.8))

$$(D^{i-1}\hat{P}_\Delta g)(\eta_{nj}) = \alpha_{ij}^n, \qquad 1\leqq i\leqq \mu_j, \quad 1\leqq j\leqq d. \tag{2.10}$$

Then the following stability theorem can be established.

THEOREM 3. *If $g\in C^M[a,b]$ and for $\{\alpha_{ij}^n\}$ in* (2.10) *there exists a constant C independent of Δ and g such that*

$$|\alpha_{ij}^n - (D^{i-1}g)(\eta_{nj})| \leqq Ch_n^{M-i+1} \max_{[x_{n-\nu},x_{n+1+\nu}]} |D^M g| \tag{2.11}$$

for $1\leqq i\leqq \mu_j$, $1\leqq j\leqq d$, $1\leqq n\leqq N$; then for partitions having bounded local mesh ratios there exists a constant C independent of Δ and g such that for x in $[x_n, x_{n+1}]$

$$|D^{i-1}(g-\hat{P}_\Delta g)(x)| \leqq CR^{M\nu}h_n^{M-i+1} \max_{[x_{n-\nu},x_{n+1+\nu}]} |D^M g| \tag{2.12}$$

$1\leqq i\leqq M$. If (2.11) *is replaced by the weaker inequality*

$$|\alpha_{ij}^n - (D^{i-1}g)(\eta_{nj})| \leqq Ch^{M-i+1}\|D^M g\| \tag{2.13}$$

then (2.12) *can be replaced by*

$$\|D^{i-1}(g-\hat{P}_\Delta g)\| \leqq CR^{M\nu}h^{M-i+1}\|D^M g\|. \tag{2.14}$$

Moreover, if $\nu=0$ then the assumption of bounded local mesh ratios can be omitted.

Proof. Choose θ_{rs} in P_M so that

$$(D^{i-1}\theta_{rs})(\zeta_j) = \delta_{ir}\delta_{js}, \qquad 1\leqq r\leqq \mu_s, \qquad 1\leqq s\leqq d;$$

then for any $G\in C^M[-\nu, 1+\nu]$, $PG = \sum_{s=1}^d \sum_{r=1}^{\mu_s} (D^{r-1}G)(\zeta_s)\theta_{rs}$. Hence

$$\begin{aligned}P_\Delta g = S_{n\nu}^{-1}PS_{n\nu}g &= \sum_{s=1}^d \sum_{r=1}^{\mu_s} (D^{r-1}S_{n\nu}g)(\zeta_s)S_{n\nu}^{-1}\theta_{rs}\\ &= \sum_{s=1}^d \sum_{r=1}^{\mu_s} h_{i(s)}^{r-1}(D^{r-1}g)(\eta_{ns})S_{n\nu}^{-1}\theta_{rs}\end{aligned}$$

($i(s)$ as in the definition of η_{ns} in (2.8)). Now $D^{l-1}(g-\hat{P}_\Delta g) = D^{l-1}(g-P_\Delta g) + D^{l-1}(P_\Delta - \hat{P}_\Delta)g$ and the first term on the right can be bounded in the desired manner from Theorem 2. For the second, if $x\in[x_n, x_{n+1}]$

$$\begin{aligned}D^{l-1}(P_\Delta - \hat{P}_\Delta)g &= \sum_{s=1}^d \sum_{r=1}^{\mu_s} h_{i(s)}^{r-1}[(D^{r-1}g)(\eta_{ns}) - \alpha_{rs}^n]D^{l-1}S_{n\nu}^{-1}\theta_{rs}\\ &= \sum_{s=1}^d \sum_{r=1}^{\mu_s} h_{i(s)}^{r-1}[(D^{r-1}g)(\eta_{ns}) - \alpha_{rs}^n]S_{n\nu}^{-1}D^{l-1}\theta_{rs}\cdot h_n^{1-l}\end{aligned}$$

so

$$|D^{l-1}(P_\Delta - \hat{P}_\Delta)g| \leqq \sum_{s=1}^{d} \sum_{r=1}^{\mu_s} [h_{i(s)}^{r-1} C h_n^{M-r+1}] h_n^{1-l} \|D^{l-1}\theta_{rs}\| \max_{[x_{n-\nu}, x_{n+1+\nu}]} |D^m g|$$

$$\leqq \left(\sum_{s=1}^{d} \sum_{r=1}^{\mu_s} \|D^{l-1}\theta_{rs}\| \right) R^{(M-1)\nu} C h_n^{M-l+1} \max_{[x_{n-\nu}, x_{n+1+\nu}]} |D^M g|.$$

Result (2.14) follows by a similar argument using (2.12). The $\nu = 0$ case is self-evident since $i(s) = n$ for all s when $\nu = 0$.

One extension of Theorem 3 is needed for the algorithms of the next section. The interval $[x_{n-\nu}, x_{n+\nu+1}]$ containing the interpolating points $P_{M,\Delta}$ is symmetric about $[x_n, x_{n+1}]$ and difficulties arise near the ends of the partition. A study of the above proofs shows that this symmetry is merely for convenience (the actual value of the error constant C is also smaller in the symmetric case); the important thing is that the number of subintervals of Δ in the smallest interval containing the interpolating points be bounded by some fixed integer for all the partitions under consideration. This integer appears in place of ν in the exponent of R in (2.12) and (2.14). Secondly, the set of interpolating points $\{\zeta_j\}$ could vary from interval to interval. The bounds (2.12) and (2.14) would remain valid as long as $\sum_{s=1}^{d} \sum_{r=1}^{\mu_s} \|D^{l-1}\theta_{rs}\|$, θ_{rs} as in the proof of Theorem 3, is uniformly bounded over the various choices of interpolating points. For the examples to follow, the number of choices of interpolating points is always some small integer independent of N, so this uniform bound does exist.

These interpolation schemes were chosen because the analysis is straightforward; while Lagrangian interpolation is more natural, at the time this paper was written, stability bounds of the type in Theorem 3 for general meshes were unknown to the author (they have since been proven). The actual behavior of Lagrangian interpolation, however, is qualitatively similar to that of the schemes considered here.

3. Collocation with monomial bases for mixed order systems. In this section we briefly describe how $\hat{\mathbf{u}}$, the collocation approximation to $\mathbf{u}$, is computed. There are several ways of representing $\hat{\mathbf{u}}$; motivated by the findings of Ascher, Pruess and Russell [3], we use a local monomial representation. Thus, on each subinterval $[x_n, x_{n+1}]$ of Δ write

$$\hat{u}_l(x) = \sum_{j=1}^{m(l)} \frac{(D^{j-1}\hat{u}_l)(x_n)(x-x_n)^{j-1}}{(j-1)!} + h_n^{m(l)} \sum_{j=1}^{k} w_{nlj} \phi_{m(l),j}\left(\frac{x-x_n}{h_n}\right). \tag{3.1}$$

Here, each $\phi_{m(l),j} \in P_{k+m(l)}$ with $D^{i-1}\phi_{m(l),j}(0) = 0$, $1 \leqq i \leqq m_l$ for consistency of notation. The remaining degrees of freedom in $\phi_{m(l),j}$ are arbitrary (as long as the set remains linearly independent), but for computational reasons detailed below, it is convenient to require

$$(D^{i-1}\phi_{m(l),j})(1) = \delta_{ij}, \qquad 1 \leqq j \leqq k, \quad 1 \leqq i \leqq k. \tag{3.2}$$

From (3.1) it follows that for $1 \leqq i \leqq m(l)+1$

$$D^{i-1}\hat{u}_l(x) = \sum_{j=i}^{m(l)} \frac{D^{j-1}\hat{u}_l(x_n)(x-x_n)^{j-i}}{(j-i)!} + h_n^{m(l)-i+1} \sum_{j=1}^{k} w_{nlj} D^{i-1}\phi_{m(l),j}\left(\frac{(x-x_n)}{h_n}\right). \tag{3.3}$$

Collocation of the differential equation (1.1) at $\xi_{nr} = x_n + \rho_r h_n$, $\{\rho_r\}_{r=1}^{k}$ the roots of the

kth degree Legendre polynomial shifted to $[0, 1]$, yields

$$\sum_{j=1}^{p}\sum_{\mu=1}^{m(j)}\sum_{i=1}^{\mu}\frac{C_{lij}(\xi_{nr})(\rho_r h_n^{\mu-i})\hat{u}_j(x_n)}{(\mu-i)!}+f_l(\xi_{nr})$$
$$=\sum_{\mu=1}^{k}\left\{D^{m(l)}\phi_{m(l),\mu}(\rho_r)w_{nl\mu}-\sum_{j=1}^{p}w_{nj\mu}\left[\sum_{i=1}^{m(j)}C_{lij}(\xi_{nr})h_n^{m(j)-i+1}D^{i-1}\phi_{m(j),\mu}(\rho_r)\right]\right\}$$

for $1\leqq l\leqq p$, $1\leqq r\leqq k$.

Define

$$\mathbf{z}_j=(\hat{u}_j, D\hat{u}_j, \cdots, D^{m(j)-1}\hat{u}_j),\qquad 1\leqq j\leqq p,$$
$$\mathbf{z}=(\mathbf{z}_1, \cdots, \mathbf{z}_p), \tag{3.4}$$
$$\mathbf{w}_{n\mu}=(w_{n1\mu}, \cdots, w_{np\mu}),\qquad 1\leqq \mu\leqq k,\quad 1\leqq n\leqq N.$$

Then the collocation equations can be written in the form

$$H_n\mathbf{z}(x_n)+(f.(\xi_{nr}))_{r=1}^{k}=G_n(\mathbf{w}_{n\mu})_{\mu=1}^{k} \tag{3.5}$$

with $H_n kp$ by m^*, $G_n kp$ by kp. Since G_n is an $O(h_n)$ perturbation of a generalized Vandermonde matrix, for h_n sufficiently small G_n^{-1} exists and, hence, (3.5) can be rewritten as

$$(\mathbf{w}_{n\mu})_{\mu=1}^{k}=G_n^{-1}H_n\mathbf{z}(x_n)+G_n^{-1}(f.(\xi_{nr}))_{r=1}^{k}. \tag{3.6}$$

The requirement that $D^{m(l)-1}\hat{u}_l$ be continuous over $[a, b]$ is not built into the representation (3.1), so $\mathbf{z}(x_{n+1}^{+})=\mathbf{z}(x_{n+1}^{-})$ must be imposed explicitly. From (3.3) it follows that

$$\mathbf{z}(x_{n+1})=M_n\mathbf{z}(x_n)+N_n(\mathbf{w}_{n\mu})_{\mu=1}^{k} \tag{3.7}$$

where M_n is m^* by m^* block diagonal with $m(l)$ by $m(l)$ upper triangular diagonal blocks (entry $i-j$ is $h_n^{j-i}/(j-i)!$); N_n is m^* by kp. The nonzero entries of N_n have the form $h_n^{m(l)-i+1}D^{i-1}\phi_{m(l),j}(1)$; because of (3.2) each row of N_n contains exactly one nonzero entry, viz., in row i ($1\leqq i\leqq m(l)$) of block-row l ($1\leqq l\leqq p$) the nonzero entry $h_n^{m(l)-i+1}$ occurs in column l of block-column i.

Substitution of (3.6) into (3.7) yields

$$\mathbf{z}(x_{n+1})=\Gamma_n\mathbf{z}(x_n)+\mathbf{g}_n \tag{3.8a}$$

where

$$\mathbf{g}_n=N_nG_n^{-1}(f.(\xi_{nr}))_{r=1}^{k}, \tag{3.8b}$$

$$\Gamma_n=M_n+N_nG_n^{-1}H_n. \tag{3.8c}$$

The simplest procedure for computing Γ_n and $\mathbf{g}_n$ is to set up H_n, $f.$, G_n; then solve $G_n^{-1}H_n$, $G_n^{-1}(f.)$. Multiplication by N_n is just simple scaling, and addition of M_n only affects diagonal blocks.

Note that if the collocation solution $\hat{\mathbf{u}}(x)$ is to be recovered, i.e., $\mathbf{w}_{n\mu}$ as well as $\mathbf{z}(x_n)$ are to be computed, then the quantities $G_n^{-1}H_n$ and $G_n^{-1}(f.)$ must be saved globally. Then $w_{n\mu}$ can be evaluated from (3.6). This requires $Nkp(m^*+1)$ storage locations. In contrast, if only the superconvergent quantities ($\mathbf{z}(x_n)$, $1\leqq n\leqq N+1$) are desired, there is no need for this additional storage.

4. Interpolation algorithms. In this section the four algorithms for computing approximations $\mathbf{v}$ to $\mathbf{u}$ are described. It is assumed that the superconvergent quantities, $\mathbf{z}(x_n)$, $1\leqq n\leqq N+1$, have been computed, and since these are used by all four algorithms

the storage and operations needed for their calculation are not counted. The algorithms differ in how they utilize the superconvergent quantities to produce a v which can be evaluated for any x in $[a, b]$.

In what follows, we give detailed descriptions of the four algorithms, asymptotic errors, operation counts and storage requirements. The latter two are implementation dependent so must be viewed with some skepticism; however, order of magnitude comparisons should be useful. The only operations counted are * and / which are weighted equally. It is assumed that values of the independent variable used to produce divided difference representations of interpolating polynomials will be generated as needed and not stored for the entire partition Δ (so that their storage requirements are independent of N). The first algorithm computes the actual collocation solution $\hat{u}$. To calculate the quantities w_{nlj} in (3.1) it is necessary to save $G_n^{-1}H_n$ and $G_n^{-1}(f\cdot)$ as mentioned at the end of § 3. Calculation of w_{nlj} and scaling then produces $D^{i-1}\hat{u}_l(x_{n+1}^-) = w_{nli}/h_n^{i-m(l)+1}$, $m(l) < i \leqq k$, (cf. (3.2)). This computation requires $(m^*+1)(kp-m^*)$ operations. The collocation solution then has an interpolatory representation in terms of $\hat{u}_l, \cdots, D^{m(l)-1}\hat{u}_l$ at x_n and x_{n+1}, and $D^{m(l)}\hat{u}_l, \cdots, D^{m(l)+k-1}\hat{u}_l$ at x_{n+1}^- for each component l. If the Newton divided difference form is used and the required differences computed once and for all for the total partition, then $\frac{1}{2}N\sum_{l=1}^{p}(m(l)+k)\cdot(m(l)+k-1)$ operations and $(kp+m^*)N$ locations are needed. However, the storage used earlier to produce w_{nlj} can be overwritten so no additional storage is actually necessary. The error is given by (2.3) and to evaluate $D^{i-1}\hat{u}_l$, $1 \leqq i \leqq m(l)$, $1 \leqq l \leqq p$, at a single point is just the usual Newton divided difference algorithm. In summary, we have

(4.1)

error: $|D^{i-1}(u_l - \hat{u}_l)(x)| \leqq Ch^{m(l)+k-i+1}, \qquad 1 \leqq i \leqq m(l)+1$

storage: $Nkp(m^*+1)$

operation counts:

overhead $\left[(m^*+1)(kp-m^*)+\frac{1}{2}\sum_{l=1}^{p}(m(l)+k)(m(l)+k-1)\right]N$

evaluation of $\hat{u}^{(\cdot)}$ $(k-1)m^*+p+\sum_{l=1}^{p}m(l)^2.$

The reasons for considering alternate approximations were to possibly reduce the amount of storage and computation needed and to improve the (asymptotic) accuracy. The first alternative, Algorithm 2, constructs an approximation v which uses a local $2k$th order Hermite representation for each component. For this, estimates of $D^{i-1}u_l$ at mesh points are needed for $1 \leqq i \leqq k$. The superconvergent quantities $(D^{i-1}\hat{u}_l)(x_n)$, $1 \leqq i \leqq m(l)$ provide a start, and the differential equation can be used to generate the higher derivatives. Since

$$D^{m(l)+\mu}u_l = \sum_{j=1}^{p}\sum_{i=1}^{m(j)} D^{\mu}(C_{lij}D^{i-1}u) + D^{\mu}f_l, \qquad 1 \leqq l \leqq p$$

the higher derivative estimates can be computed recursively with the superconvergent approximations used to start the recursion. Errors can be bounded using the results in § 2. For this example $\nu = 0$, the interpolating set $\{\zeta_j\}$ consists of k zeros and k ones, and the perturbation inequality (2.11) holds with $M = 2k$ (by recursion). Thus, the error in the ith derivative is $O(h^{2k-i})$.

Of course, this algorithm suffers from the disadvantage that derivatives of the coefficient functions (or $O(h^{2k})$ approximations to them) are needed for $k > m(l)$. The numerical results in § 5 have used exact derivatives, but it would be more convenient (though more expensive) to use difference approximations. In either case extra function evaluations are needed; the summary below assumes exact derivatives are available. Once the mesh point values have been computed, the work to generate the divided difference tables and storage requirements is similar to Algorithm 1, except here the polynomials all have order $2k$ instead of $m(l)+k$.

(4.2)

error:	$\lvert D^{i-1}(u_l - v_l)(x)\rvert \leqq Ch^{2k-i+1}, \qquad 1 \leqq i \leqq m(l)+1$
storage:	$2Nkp$
operation counts:	
overhead	$\left[kp(2k-1) + m^* \sum_{l=1}^{p} (k-m(l))(k-m(l)+1) \right] N$
evaluation of $\mathbf{v}^{(\cdot)}$	$(2k-1)m^* + p$
fn. evaluations:	$(kp - m^*)(m^*+1)N.$

The remaining two algorithms avoid the extra coefficient evaluations of Algorithm 2, as they use only superconvergent quantities to generate interpolants. To do this, however, they need values from several intervals to produce approximations on one; consequently, factors from local mesh ratios appear in the error bounds and numerical difficulties may be anticipated for highly nonuniform meshes.

Algorithm 3, at least away from the ends of the partition, in the notation of § 2 has $\nu = \text{int}\,((k-1)/(m(l))$ for the lth component of $\mathbf{v}$. The integers $-\nu+1, \cdots, \nu$ are in $\{\zeta_j\}$, each having multiplicity $m(l)$; the remaining values $-\nu, 1+\nu$ occur with equal multiplicity such that there are $M = 2k$ total points. Hypothesis (2.11) is satisfied for this M so the error in each component is given by (2.12) for the stated ν. In short, each component of $\mathbf{v}$ on $[x_n, x_{n+1}]$ is an order $2k$ polynomial which interpolates superconvergent data from nearby symmetric intervals. Near the ends nonsymmetric superconvergent data must be used: this makes the effective ν in the error bound (2.12) larger, but the asymptotic behavior, $O(h^{2k})$, remains the same. The work and storage is the same as for Algorithm 2 once the latter's extra higher derivatives are available.

(4.3)

error:	$\lvert D^{i-1}(u_l - v_l)(x)\rvert \leqq CR^{2k\nu(l)}h^{2k-i+1}, \qquad 1 \leqq i \leqq m(l)+1$
	$\nu(l) = \text{int}\,((k-1)/m(l))$
storage:	$2Nkp$
operation counts:	
overhead	$kp(2k-1)N$
evaluation of $\mathbf{v}^{(\cdot)}$	$(2k-1)m^* + p.$

The final algorithm fits polynomials to each derivative independently using $2k$ superconvergent values chosen symmetrically. Thus, for this case $\nu = k-1$, $\{\zeta_j\} = \{-\nu, \cdots, 1+\nu\}$ with each interpolation point having multiplicity one. Again nonsymmetric points must be used near the ends of the partition. The work and storage counts

differ from the previous two algorithms, since a divided difference table is generated for each derivative of each component of v. The (apparent) advantage is that the asymptotic rate of convergence is better for the higher derivatives.

(4.4)

error:	$\lvert D^{i-1}(u_l - v_l)(x)\rvert \leqq CR^{2k(k-1)}h^{2k}, \quad 1 \leqq i \leqq m(l)+1$
storage:	$2Nkm^*$
operation counts:	
overhead	$km^*(2k-1)N$
evaluation of $\mathbf{v}^{(\cdot)}$	$2km^*$.

To make some comparisons, see Table 1 which contains storage requirements, operation counts and asymptotic error form for the special case of p mth order systems when $k=4$. For large p there is definite savings in storage with the alternative algorithms; however, Algorithm 2 requires a large amount of overhead, and Algorithms 3 and 4 may have trouble for nonuniform meshes. Actual accuracies achieved for typical examples can be seen in the next section.

TABLE 1
Comparison of algorithms for mth order systems ($m \leqq k = 4$).

Algorithm	1	2	3	4
storage ($* N$)	$4mp^2+4p$	$8p$	$8p$	$8mp$
operation counts:				
overhead ($* Np$)	$m(4-m)p+\frac{1}{2}(m^2+5m+20)$	$m(4-m)(5-m)p+28$	28	$28m$
per evaluation of $\mathbf{v}$	$(m^2+3m+1)p$	$(7m+1)p$	$(7m+1)p$	$8mp$
error bound for $\mathbf{v}$	h^{m+4}	h^8	$R^{8[3/m]}h^8$	$R^{24}h^8$

5. Numerical examples and conclusions. Many varied examples of systems have been tried using the four algorithms from the preceding section. These tended to be either easy problems where small N and uniform meshes were sufficient, or problems with boundary layers or some other kind of rapid change where nonuniform meshes are crucial for accurate results. Three typical examples are presented here. All calculations were done in REAL $*$ 8 arithmetic (14 hexadecimal digits) on an IBM 4341. The notation .5-4 for $.5\times10^{-4}$ is used in the tables. Gauss elimination with partial pivoting, but without row scaling, was used to solve linear systems. All displayed results used $k=4$ though other values have been tried with similar outcomes. The column heading e_l refers to $u_l - v_l$.

The first example has very smooth solutions and is presented to illustrate the rates of convergence predicted by the theory.

$$u_1''' = u_1' - 4\exp(2x)u_2 + 4(x^2+1)\exp(x),$$

$$u_2' = -xu_1 + xu_1'' - u_2 - 4x^2\exp(x) + (2x-1)\exp(-x),$$

$$u_1(0) = u_2(0) = u_1(1) = u_2(1) = 0,$$

so that $u_1(x) = x(x-1)\exp(x)$, $u_2(x) = x(x-1)\exp(-x)$. Here $m_1 = 3$, $m_2 = 1$ so $O(h^8)$ accuracy is expected at mesh points, while the order for max-norm errors depends on the algorithm being used. Table 2 displays the numerical errors for uniform meshes

TABLE 2
Numerical results for Example 1.

		Maximum error at knots							
	N	e_1	rate	e_1'	rate	e_1''	rate	e_2	rate
	8	.15-10		.76-10		.19-9		.24-10	
	12	.61-12	7.9	.30-11	8.0	.75-11	8.0	.94-12	8.0
	16	.62-13	8.0	.31-12	7.9	.76-12	8.0	.96-13	7.9
Algorithm		Maximum error over [0, 1]							
	8	.28-10		.15-8		.92-7		.17-7	
1	12	.14-11	7.4	.13-9	6.0	.12-7	5.0	.23-8	4.9
	16	.18-12	7.1	.24-10	5.9	.30-8	4.8	.56-9	4.9
	8	.16-10		.76-10		.20-8		.24-10	
2	12	.63-12	8.0	.33-11	7.7	.16-9	6.2	.96-12	7.9
	16	.64-13	8.0	.39-12	7.4	.30-10	5.8	.98-13	7.9
	8	.20-10		.25-9		.12-7		.37-7	
3	12	.79-12	8.0	.14-10	7.1	.11-8	5.9	.17-8	7.6
	16	.80-13	8.0	.19-11	6.9	.20-9	5.9	.18-9	7.8
	8	.10-6		.13-6		.16-6		.37-7	
4	12	.46-8	7.6	.59-8	7.6	.73-8	7.6	.17-8	7.6
	16	.50-9	7.7	.64-9	7.7	.79-9	7.7	.18-9	7.8

when $N=8$, 12, 16, as well as estimated rates. The latter are calculated by $\log e(N_1)/e(N_2) \div \log N_2/N_1$. The rates are in agreement with the theory; however, it is interesting to study the actual magnitude of the errors. The collocation solution (Algorithm 1) for u_1 using these meshes is actually better than for the asymptotically more accurate Algorithm 4, while for u_2 the results are comparable. Of course, eventually the results for Algorithm 4 would be superior, but this will not occur until extremely high accuracy is attained. Such behavior seems typical for those problems with very smooth solutions. Algorithm 2 yields the most accurate solutions for these N, with little accuracy being lost (except in e_1'') in the interpolation process. Algorithm 3 is only slightly better than Algorithm 1 for these accuracies, even though it has faster asymptotic convergence.

The second example (from [1]) has a spike at $x=0$ so variable meshes must be used.

$$u'' = -(\pi x \sin \pi x)u/\varepsilon - xu'/\varepsilon - \pi^2 \cos \pi x,$$

$$u(-1) = -2, \qquad u(1) = 0,$$

which has the solution $u(x) = \cos \pi x + \operatorname{erf}(x/\sqrt{2\varepsilon})/\operatorname{erf}(1/\sqrt{2\varepsilon})$. For $\varepsilon = 10^{-4}$, $N=32$ the points of Δ chosen symmetrically about the origin, $\Delta = \{\cdots, 0, .0045, .0092, .014, .019, .024, .03, .038, .045, .054, .067, .11, .25, .44, .65, .83, 1\}$, the results are summarized in Table 3. Note that this mesh was a consequence of attempting to equi-distribute an approximation to $\|u^{(6)}\|^{1/6}$ (similar behavior occurred when various other quantities were equi-distributed).

Here, Algorithm 4 really suffers because it interpolates data over seven subintervals to calculate v on one subinterval: the factors from local mesh ratios play an important role in the error. Algorithm 3 is also impacted by this, but to a much lesser extent as it only reads data from three adjacent intervals to calculate v on one. The surprise for

TABLE 3
Numerical results for Example 2.

	Maximum error at knots	
	e .35-7	e' .20-4
Algorithm	Maximum error over $[-1, 1]$	
1	.19-6	.25-3
2	.11-1	.27
3	.11-5	.38-3
4	.26-3	.10

this example is how poorly Algorithm 2 performed. The higher derivatives estimated from the differential equation using the approximations $\hat{u}$ and $\hat{u}'$ are quite large near ± 1, even though u is smooth there. This results from the $1/\varepsilon$ factor in the differential equation magnifying the errors in $\hat{u}$ and $\hat{u}'$; in fact, if $u'' = \pi^2 \cos \pi x$, $u''' = \pi^3 \sin \pi x$ are used to generate values for Algorithm 2 away from the spike at $x = 0$, then $e = .72\text{-}6$, $e' = 28\text{-}4$ which compares favorably with the collocation solution. Of course, in practice, one does not have the exact answer at hand!

The final example is chosen to mimic the behavior of another example from [1] without requiring such complicated constants.

$$u_1'' = 100u_1 - u_1' - 100u_2,$$

$$u_2'' = -10^4 u_1 + 10^4 u_2 - u_2',$$

$$u_1(0) + u_1'(0) = 3 + 101 \exp(-500)/50,$$

$$u_2(0) = 1 - 2\exp(-505),$$

$$u_1(5) - u_2(5) = 101(1 - \exp(-505))/50,$$

$$u_1'(5) + u_2(5) = 1 + 201 \exp(-505)/50.$$

Here, the exact answers are

$$u_1(x) = 1 - 2\exp(-x) + \{\exp[100(x-5)) - \exp(101x)\}/50,$$

$$u_2(x) = 1 - 2\exp(-x) - 2\exp[100(x-5)] + 2\exp(-101x).$$

This problem has mild boundary layers along with some oscillation; as in the previous example, nonuniform meshes are called for. Table 4 displays the results for $N = 20$ with $\Delta = \{0, .0095, .02, .032, .048, .068, .097, .15, .31, 1.25, 2.35, 3.67, 4.7, 4.85, 4.90, 4.93, 4.95, 4.97, 4.98, 4.99, 5\}$.

As earlier, Algorithm 4 is inferior due to the nonuniform mesh, Algorithm 3 is much less affected by this. Algorithm 2 is more accurate than Algorithm 1, though some accuracy is lost from that of the superconvergent values.

From examining these examples (and others not included), it is clear that no matter which alternative is used, at least some of the accuracy of the superconvergent quantities is often lost through the interpolation process. At times the results are inferior even to the collocation solution. Algorithm 4 has little to recommend it; it is no more efficient than Algorithm 3 and its superior asymptotic accuracy is rarely evident. More importantly, it suffers severely on problems with highly nonuniform meshes. Algorithm

TABLE 4
Numerical results for Example 3.

	Maximum errors at knots			
	e_1 .85-7	e_1' .28-6	e_2 .20-6	e_2' .29-4
Algorithm	Maximum error over [0, 5]			
1	.19-5	.31-4	.71-5	.31-2
2	.15-6	.19-5	.72-6	.19-3
3	.58-5	.58-4	.17-3	.58-2
4	.16-2	.16	.16	.16+2

2 can be expensive because of the extra function evaluations required, but it fairly consistently produced the most accurate results. Unfortunately, as Example 2 shows, this algorithm can produce quite poor results in isolated cases. In contrast, Algorithm 3 seems fairly attractive: it produces answers whose accuracy is comparable to that of the collocation solution, and it requires less storage and computation. The savings are considerable in the case of large systems (large p).

Acknowledgment. Portions of this paper were written during a "summer" visit to the Australian National University. The author is grateful to Mike Osborne and other members of the Department of Statistics for their hospitality.

REFERENCES

[1] U. ASCHER, J. CHRISTIANSEN AND R. RUSSELL, *A collocation solver for mixed order systems of boundary value problems*, Math. Comp., 33 (1979), pp. 659-679.
[2] ———, *Collocation software for boundary value* ODE's, ACM Trans. Math. Software, 7 (1981), pp. 209-222.
[3] U. ASCHER, S. PRUESS AND R. RUSSELL, *On spline basis selection for solving differential equations*, SIAM J. Numer. Anal., 20 (1983), pp. 121-142.
[4] G. BIRKHOFF AND C. DE BOOR, *Error bounds for spline interpolation*, J. Math. Mech., 13 (1964), pp. 827-836.
[5] J. CERUTTI, *Collocation for systems of ordinary differential equations*, Computer Sci. Tch. Report 230, Univ. Wisconsin, Madison, 1974.
[6] C. DE BOOR, *A Practicafi Guide to Splines*, Springer-Verlag, Madison, New York, 1978.
[7] C. DE BOOR AND B. SWARTZ, *Collocation at Gaussian points*, SIAM J. Numer. Anal., 10 (1973), pp. 582-606.
[8] E. HOUSTIS, *A collocation method for systems of nonlinear ordinary differential equations*, J. Math. Anal. Appl., 62 (1978), pp. 24-37.
[9] H. KELLER, *Numerical Solution of Two Point Boundary Value Problems*, CBMS Regional Conference Series in Applied Mathematics 24, Society for Industrial and Applied Mathematics, Philadelphia, 1976.
[10] S. PRUESS, *Estimating the eigenvalues of Sturm-Liouville problems by approximating the differential equation*, SIAM J. Numer. Anal., 10 (1973), pp. 55-68.
[11] M. SCHULTZ, *Spline Analysis*, Prentice Hall, Englewood Cliffs, 1973.
[12] B. SWARTZ AND R. VARGA, *Error bounds for spline and L-spline interpolation*, J. Approx. Theory, 6 (1972), pp. 1-49.

SIAM J. SCI. STAT. COMPUT.
Vol. 7, No. 1, January 1986

022

SMOOTHER INTERPOLANTS FOR ADAMS CODES*

H. A. WATTS† AND L. F. SHAMPINE‡

Abstract. The Adams family of ODE methods is based on polynomial interpolants to past values obtained on a discrete mesh. It is desirable, and in some circumstances essential, to have an interpolation scheme which produces globally continuous approximations to both the solution and its derivative. In this paper we describe several ways of achieving the desired result in an efficient manner, and we give particular emphasis to the task of achieving mathematical continuity by the computational algorithm.

Key words. ordinary differential equations, Adams methods, smoother interpolants

1. Introduction. Discrete valued methods for the solution of the initial value problem

$$y'(x)=f(x,y), \qquad a\leqq x\leqq b,$$
$$y(a) \quad \text{given}$$

produce approximate solution vectors $y_n \doteq y(x_n)$ on a mesh $a=x_0<x_1<\cdots$. They also produce an approximate derivative at mesh points by

$$y'_n=f_n=f(x_n,y_n).$$

The Adams methods are based on polynomial interpolants to the y_n and f_n values. As a consequence there are several natural polynomial approximations to $y(x)$ and $y'(x)$ which are as accurate for x between mesh points as the approximations at the mesh points themselves. Just which interpolant seems most natural depends on the point of view. In the next section the two most popular interpolants are described. In one respect, neither is natural; either the approximation to $y(x)$ or the approximation to $y'(x)$ has jumps at all the mesh points. For example, the interpolant of SODE [3] and of DE/STEP, INTRP [6] and its descendant DEABM [7] defines a globally continuous approximation to the derivative, but the interpolant itself has a jump at each mesh point of a size which is comparable to the local error tolerance.

In general, it is desirable, and in some circumstances essential, to work with an interpolant which produces globally continuous approximations to both the solution and its derivative. This paper describes several ways of achieving the desired result in an efficient manner. One of our goals is to implement the scheme so that roundoff effects are minimized when evaluating the interpolant at the mesh points; i.e., we would like for the mathematical continuity to be very nearly achieved by the computational algorithm.

2. Interpolation formulas. Suppose that the numerical solution has been advanced to x_n and that the data y_{n+1-j} and

$$f_{n+1-j}=f(x_{n+1-j},y_{n+1-j})$$

are available for $j=1,\cdots,k$. The explicit Adams–Bashforth formula of order k is defined in terms of the polynomial $P_{k,n}(x)$ of degree (at most) $k-1$ which is uniquely

* Received by the editors May 22, 1984, and in final form November 5, 1984. This work was performed at Sandia National Laboratories and was supported by the U.S. Department of Energy under contract DE-ACO4-76DP00789.

† Applied Mathematics Division, 2646, Sandia National Laboratories, Albuquerque, New Mexico 87185.

‡ Numerical Mathematics Division, 1642, Sandia National Laboratories, Albuquerque, New Mexico 87185.

determined by the interpolation conditions

$$P_{k,n}(x_{n+1-j}) = f_{n+1-j}, \qquad j = 1, \cdots, k.$$

The formula then defines a "predicted" solution p_{n+1} at $x_{n+1} = x_n + h_{n+1}$ by

$$p_{n+1} = y_n + \int_{x_n}^{x_{n+1}} P_{k,n}(t)\, dt.$$

A function evaluation is now made,

$$f^p_{n+1} = f(x_{n+1}, p_{n+1}),$$

and the new information is used to define a "corrected" approximation to $y(x_{n+1})$. There are two natural ways to proceed. One is to drop the oldest value f_{n+1-k}, and the other is to retain it and raise the degree of the interpolating polynomial. We shall suppose that the latter approach is used so that the integration is advanced with the result of an Adams–Moulton formula of order $k+1$ (local extrapolation). This is done by first defining the polynomial $P^*_{k+1,n}(x)$ of degree k which satisfies

$$P^*_{k+1,n}(x_{n+1-j}) = f_{n+1-j}, \qquad j = 1, \cdots, k,$$
$$P^*_{k+1,n}(x_{n+1}) = f^p_{n+1},$$

and then defining the corrected approximation y_{n+1} by

$$y_{n+1} = y_n + \int_{x_n}^{x_{n+1}} P^*_{k+1,n}(t)\, dt.$$

This process defines a numerical solution only at the end of the step. A very natural interpolant which defines an approximation for all $x_n < x \leqq x_{n+1}$ is

$$\eta(x) = y_n + \int_{x_n}^{x} P^*_{k+1,n}(t)\, dt = y_{n+1} + \int_{x_{n+1}}^{x} P^*_{k+1,n}(t)\, dt. \tag{1}$$

Notice that because of the way y_{n+1} is defined, $\eta(x)$ is a polynomial of degree $k+1$ which satisfies $\eta(x_n) = y_n$, $\eta(x_{n+1}) = y_{n+1}$, and

$$\eta'(x_{n+1-j}) = f_{n+1-j}, \qquad j = 1, \cdots, k,$$
$$\eta'(x_{n+1}) = f^p_{n+1}.$$

If this interpolant is used for each interval (x_n, x_{n+1}), it is clear that the resulting piecewise polynomial function is globally continuous. Unfortunately, its derivative is not. This is seen from the fact that the interpolant on (x_n, x_{n+1}) has

$$\eta'(x_{n+1}) = f^p_{n+1} = f(x_{n+1}, p_{n+1})$$

and the interpolant on (x_{n+1}, x_{n+2}) has

$$\eta'(x_{n+1}) = f_{n+1} = f(x_{n+1}, y_{n+1}).$$

In general $p_{n+1} \neq y_{n+1}$; indeed, a fraction of the difference $y_{n+1} - p_{n+1}$ is an estimate of the local error of the step. The jump in the derivative can be bounded by

$$\| f_{n+1} - f^p_{n+1} \| \leqq L \| y_{n+1} - p_{n+1} \|$$

where L is a local Lipschitz constant for f.

There is another natural way to interpolate (which is implemented in the codes of [3], [6], [7]). Using the polynomial $P_{k+1,n+1}(x)$, an approximation is defined for all

$x_n < x \leqq x_{n+1}$ by

$$y_I(x) = y_{n+1} + \int_{x_{n+1}}^{x} P_{k+1,n+1}(t)\, dt. \tag{2}$$

This is a polynomial of degree $k+1$ with $y_I(x_{n+1}) = y_{n+1}$, and the defining conditions for $P_{k+1,n+1}$ are

$$y_I'(x_{n+1-j}) = P_{k+1,n+1}(x_{n+1-j}) = f_{n+1-j}, \qquad j = 0, 1, \cdots, k.$$

This piecewise polynomial interpolant leads to a globally continuous approximation of $y'(x)$, but because normally $y_I(x_n) \neq y_n$, the interpolant itself has jumps at mesh points. The report [5] identifies the jump at x_n as

$$y_n - y_I(x_n) = h_{n+1} g_{k+1,1}[f_{n+1} - f^p_{n+1}]$$

where

$$h_{n+1} g_{k+1,1} = \int_{x_n}^{x_{n+1}} \omega_k(t)\, dt$$

and

$$\omega_k(t) = \prod_{j=1}^{k} \left(\frac{t - x_{n+1-j}}{x_{n+1} - x_{n+1-j}} \right). \tag{3}$$

The jump can be bounded by

$$\| y_n - y_I(x_n) \| \leqq h_{n+1} L g_{k+1,1} \| y_{n+1} - p_{n+1} \|.$$

It is argued in [5] that the right-hand side here is normally comparable in size to the local error tolerance which the user specifies.

3. Polynomial representations. Very efficient schemes have been based on backward divided difference representations for the underlying polynomial interpolants. In particular,

$$\begin{aligned} P_{k,n}(t) = {} & f[x_n] + (t - x_n) f[x_n, x_{n-1}] + (t - x_n)(t - x_{n-1}) f[x_n, x_{n-1}, x_{n-2}] \\ & + \cdots + (t - x_n)(t - x_{n-1}) \cdots (t - x_{n-k+2}) f[x_n, x_{n-1}, \cdots, x_{n-k+1}] \end{aligned} \tag{4}$$

with $f[x_n, \cdots, x_{n-i+1}]$ being the usual notation for the divided differences of the f_{n-j+1} values. One of the advantages of using the divided difference form is that it is easy to obtain the polynomial which interpolates to one additional data point. Thus

$$P^*_{k+1,n}(t) = P_{k,n}(t) + (t - x_n)(t - x_{n-1}) \cdots (t - x_{n-k+1}) f^p[x_{n+1}, x_n, \cdots, x_{n-k+1}] \tag{5}$$

where the superscript p on the divided difference is used to remind us that $P^*_{k+1,n}(t)$ interpolates to f^p_{n+1} at x_{n+1}. However, we could also express this polynomial by the mathematically equivalent form

$$\begin{aligned} P^*_{k+1,n}(t) = {} & f^p[x_{n+1}] + (t - x_{n+1}) f^p[x_{n+1}, x_n] + (t - x_{n+1})(t - x_n) f^p[x_{n+1}, x_n, x_{n-1}] \\ & + \cdots + (t - x_{n+1})(t - x_n) \cdots (t - x_{n-k+2}) f^p[x_{n+1}, x_n, \cdots, x_{n-k+1}], \end{aligned} \tag{6}$$

a fact we shall use later on. Similarly, we can write $P_{k+1,n+1}(t)$ either as

$$\begin{aligned} P_{k+1,n+1}(t) = {} & f[x_{n+1}] + (t - x_{n+1}) f[x_{n+1}, x_n] + (t - x_{n+1})(t - x_n) f[x_{n+1}, x_n, x_{n-1}] \\ & + \cdots + (t - x_{n+1})(t - x_n) \cdots (t - x_{n-k+2}) f[x_{n+1}, x_n, \cdots, x_{n-k+1}], \end{aligned} \tag{7}$$

the form used in the interpolation routine in [3], [6], [7], or else as

$$P_{k+1,n+1}(t) = P_{k,n}(t) + (t-x_n)(t-x_{n-1})\cdots(t-x_{n-k+1})f[x_{n+1}, x_n, \cdots, x_{n-k+1}]. \tag{8}$$

If we define

$$\psi_i(n+1) = x_{n+1} - x_{n-i+1} = \sum_{j=0}^{i-1} h_{n-j+1},$$

it is easy to see that for $i \geqq 1$,

$$f^p[x_{n+1}, x_n, \cdots, x_{n-i+1}] = f[x_{n+1}, x_n, \cdots, x_{n-i+1}] + \frac{f^p_{n+1} - f_{n+1}}{\psi_1(n+1)\psi_2(n+1)\cdots\psi_i(n+1)}. \tag{9}$$

It is convenient and economical to modify the divided differences so that they reduce to backward differences when the step sizes are constant. This is achieved by defining

$$\phi_1(n+1) = f[x_{n+1}] = f_{n+1},$$

$$\phi_{i+1}(n+1) = \psi_1(n+1)\psi_2(n+1)\cdots\psi_i(n+1)f[x_{n+1}, x_n, \cdots, x_{n-i+1}], \qquad i \geqq 1.$$

Using these modified divided differences, we can now express (9) more simply as

$$\phi^p_i(n+1) = \phi_i(n+1) + (f^p_{n+1} - f_{n+1}), \tag{10}$$

which is valid for each $i \geqq 1$. Note also that we can write

$$\phi_i(n+1) - \phi_{i+1}(n+1) = \frac{\psi_1(n+1)\cdots\psi_{i-1}(n+1)}{\psi_1(n)\cdots\psi_{i-1}(n)}\phi_i(n) \equiv \phi^*_i(n), \tag{11}$$

which relates the differences at steps n and $n+1$ (using the same notation as in [2], [6]).

When integrating the underlying polynomial interpolants, we also find that it is convenient to introduce a normalized variable s such that $t = x_n + sh_{n+1}$ for integrations of (4), (5) or (8) and such that $t = x_{n+1} - sh_{n+1}$ for integrations of (6) or (7). (The implementation of the $y_I(x)$ interpolant in [3], [6], [7] uses $t = x_{n+1} + s(x - x_{n+1})$.) Furthermore, effective implementation of predictor-corrector schemes, derived from integrating (4) and (5) to produce p_{n+1} and y_{n+1}, respectively, require efficient use of storage and this dictates that some computations be performed in a certain order and in a certain way. Without going into the details, which can be found in [2] or [6], the polynomial representations for (4) and (5), which are used in [3], [6], [7], are

$$P_{k,n}(t) = \sum_{i=1}^{k} c_{i,n}(s)\phi^*_i(n), \tag{12}$$

and

$$P^*_{k+1,n}(t) = P_{k,n}(t) + c_{k+1,n}(s)\phi^p_{k+1}(n+1), \tag{13}$$

where the $c_{i,n}(s)$ coefficients are just the $\omega_{i-1}(t)$ of (3) in the new variable, and which can be written as

$$c_{i,n}(s) = \begin{cases} 1, & i = 1, \\ s, & i = 2, \\ \gamma_{i,n}(s)c_{i-1,n}(s), & i \geqq 3, \end{cases} \tag{14}$$

with

$$\gamma_{i,n}(s) = 1 + \frac{(s-1)h_{n+1}}{\psi_{i-1}(n+1)} \equiv 1 + (s-1)\alpha_{i-1}(n+1). \tag{15}$$

Using (11), (12), (14), (15) with the representation of $P_{k+1,n+1}(t)$ given by (8), we obtain

$$
\begin{aligned}
P_{k+1,n+1}(t) &= P_{k,n}(t) + c_{k+1,n}(s)\phi_{k+1}(n+1) \\
&= \sum_{i=1}^{k} c_{i,n}(s)\{\phi_i(n+1) - \phi_{i+1}(n+1)\} + c_{k+1,n}(s)\phi_{k+1}(n+1) \qquad (16) \\
&= \phi_1(n+1) + \sum_{i=1}^{k} \{c_{i+1,n}(s) - c_{i,n}(s)\}\phi_{i+1}(n+1).
\end{aligned}
$$

This form will be useful in the subsequent development. We shall also need the following mathematically equivalent formulations: Setting $h = x - x_{n+1}$ and letting $c^I_{i,n+1}(s, h)$ denote the coefficients defined by the interpolant $y_I(x)$, corresponding to the use of form (7), we have (upon employing (10) and noting the similarity of (6) to (7))

$$
\begin{aligned}
P_{k+1,n+1}(t) &= \sum_{i=1}^{k+1} c^I_{i,n+1}(s, h)\phi_i(n+1) \\
&= \sum_{i=1}^{k+1} c^I_{i,n+1}(s, h)\phi^p_i(n+1) + (f_{n+1} - f^p_{n+1}) \sum_{i=1}^{k+1} c^I_{i,n+1}(s, h) \qquad (17) \\
&= P^*_{k+1,n}(t) + (f_{n+1} - f^p_{n+1}) \sum_{i=1}^{k+1} c^I_{i,n+1}(s, h).
\end{aligned}
$$

4. Smoother interpolants. The two interpolants $\eta(x)$ and $y_I(x)$ defined in § 2 have the same order of accuracy; it is their global smoothness that distinguishes them in practice. Stetter [9] has suggested that $\eta(x)$ should be preferred for approximating $y(x)$, and $y'_I(x)$ for approximating $y'(x)$. This is intuitively clear based on our earlier observations about the behavior of the interpolants at the mesh points. He also points out that if the routine INTRP in [6] were supplied with the $\phi^p_i(n+1)$ differences instead of the $\phi_i(n+1)$ values, the polynomial $P^*_{k+1,n}(t)$ would be obtained instead of $P_{k+1,n+1}(t)$ (compare (6) and (7)), yielding, therefore, $\eta(x)$ and $\eta'(x)$ instead of $y_I(x)$ and $y'_I(x)$. Of course it is too expensive to retain both sets of differences, and it is the most recent $\phi_i(n+1)$ values which must be kept. A simple way of modifying an algorithm like INTRP to compute $\eta(x)$ and $y'_I(x)$ simultaneously is the following (suggested by Krogh [4]): Integrating (17) and using (1) and (2), we find

$$
\begin{aligned}
y_I(x) &= y_{n+1} + h \sum_{i=1}^{k+1} g^I_{i,1}(h)\phi_i(n+1) \\
&= \eta(x) + (f_{n+1} - f^p_{n+1})h \sum_{i=1}^{k+1} g^I_{i,1}(h) \qquad (18)
\end{aligned}
$$

where $h = x - x_{n+1}$ and the $g^I_{i,1}(h)$ are precisely those coefficients already computed in INTRP,

$$
g^I_{i,1}(h) = \int_0^1 c^I_{i,n+1}(s, h)\, ds.
$$

Thus, one can form $\eta(x)$ by "correcting" $y_I(x)$ provided that $f_{n+1} - f^p_{n+1}$ is made available. This is certainly a simple and inexpensive way to define approximations to the solution and its derivative which are globally continuous, at least mathematically.

Numerical continuity is apparently not achievable with this form (see the results in § 6) because roundoff cannot be eliminated at the mesh points. Furthermore, this approach is conceptually a bit unsatisfactory because the derivative approximation is not the derivative of the solution approximation.

Stetter also discusses another interpolant, requiring an additional function evaluation per step, which is defined as a correction to $\eta(x)$. Based on his analysis, he concludes that this results in an improved interpolation procedure. We have not implemented and tested this interpolant because it is too costly except in the context of using the asymptotically correct local error estimate which he proposes. Moreover, the derivative of the new interpolant has a jump at each mesh point, just as $\eta(x)$ does.

Stetter's conclusions are plausible enough, but just how smooth should the interpolant be globally? The normal assumption is that $f(x, y)$ is continuous, so that $y(x)$ itself is at least globally C^1. This suggests increasing the order of interpolation to provide a globally C^1 interpolant. This we describe how to do in the remainder of the paper. The worth of a smoother interpolant is easily seen in the context of the common task of finding where a given function $g(x, y(x), y'(x))$ has a root. The solution $y(x)$ and its derivative $y'(x)$ are replaced by interpolants so it is obviously valuable to avoid jumps in the approximating functions. One of us (HAW) was recently extending the DEABM code to provide this capability and recognized that the present study was needed.

To approximate $y(x)$ for $x_n < x \leqq x_{n+1}$ we consider a polynomial $T(x)$ of degree $k+2$ which satisfies

$$T(x_n) = y_n, \qquad T(x_{n+1}) = y_{n+1},$$
$$T'(x_{n+1-j}) = f_{n+1-j}, \qquad j = 0, 1, \cdots, k.$$

This polynomial interpolates all the data on hand describing the solution of the problem and obviously furnishes a globally C^1 approximation. The trick is to evaluate $T(x)$ in an efficient manner so that the continuity conditions are well satisfied when the finite precision arithmetic of the computer is used. It is natural to proceed from either $\eta(x)$ or $y_I(x)$ because efficient, numerically effective schemes for their evaluation are known, and they need only small "corrections" to get rid of their jumps and so result in $T(x)$. It is immediately verified that $T(x)$ can be written in the form

$$T(x) = y_I(x) + [y_n - y_I(x_n)] \frac{Q(x)}{Q(x_n)}$$

where

$$Q(x) = \int_{x_{n+1}}^{x} (t - x_{n+1}) \omega_k(t)\, dt$$

and $\omega_k(t)$ is given by (3). Integration by parts leads to

$$Q(x) = h(h + \psi_k(n+1)]\, g^I_{k+1,1}(h) - h^2 g^I_{k+1,2}(h),$$

upon using the notation explained earlier along with $h = x - x_{n+1}$. This formulation requires the evaluation of

$$g^I_{k+1,2}(h) = \int_0^1 \int_0^{s_1} c^I_{k+1,n+1}(s, h)\, ds\, ds_1,$$

but, as noted in [8], a simple extension of the scheme already being used allows us to evaluate $Q(x)$ and, hence, $T(x)$ with reasonable efficiency.

Notice that we need to evaluate $y_I(x_n)$, $Q(x_n)$ as well as $y_I(x)$, $Q(x)$, meaning that the interpolation coefficients $g^I_{i,1}$ and $g^I_{k+1,2}$ must be computed for two different values of h. We shall (loosely) describe this situation as "having to perform two interpolations". Obviously it would be better to have a scheme which computes only one set of interpolation coefficients, especially when dealing with problems requiring answers at a great many specified output points. Although it is easy enough to organize matters so as to form the coefficients $g^I_{i,1}(-h_{n+1})$ and $g^I_{k+1,2}(-h_{n+1})$ in an initialization phase and then reuse them for each $x_n < x < x_{n+1}$, we do not consider it to be worth a design inconvenient to users. Instead, we describe an alternative approach below for reducing the work.

A seemingly different formulation is provided by the following approach [4]: Consider adding a higher order difference term to (7) and integrating it in the same manner as described previously. Making the natural extension to $c^I_{k+2,n+1}(s, h)$ and looking at the first part of (18), this suggests writing

$$T(x) = y_I(x) + hg^I_{k+2,1}(h)e,$$

where e represents some higher order difference expression and, as before, h is the quantity $x - x_{n+1}$. All interpolation conditions will be satisfied provided e is chosen by the requirement $T(x_n) = y_n$. Thus, we obtain

$$T(x) = y_I(x) - \left[\frac{hg^I_{k+2,1}(h)}{h_{n+1}g^I_{k+2,1}(-h_{n+1})}\right][y_n - y_I(x_n)].$$

However, the recursive scheme used to define $y_I(x)$ shows that (see, e.g., [2], [6] or [8])

$$g^I_{k+2,1}(h) = \left(\frac{h + \psi_k(n+1)}{\psi_{k+1}(n+1)}\right) g^I_{k+1,1}(h) - \frac{h}{\psi_{k+1}(n+1)}\, g^I_{k+1,2}(h).$$

Comparing this definition to the form of $Q(x)$, we see that this representation for $T(x)$ leads to an algorithm which is virtually identical to the previous one. So nothing is to be gained with this formulation whenever the recursion is used to evaluate the integral coefficients.

In order to eliminate most of the work arising from the extra set of interpolation coefficients, it is necessary to use a different approach for the integration of $P_{k+1,n+1}(t)$. Specifically, using the form (16), we define

$$\begin{aligned} S(x) &= y_n + \int_{x_n}^{x} P_{k+1,n+1}(t)\, dt \\ &= y_n + h_{n+1}\left\{ g_{1,1}(\xi)\phi_1(n+1) + \sum_{i=1}^{k} [g_{i+1,1}(\xi) - g_{i,1}(\xi)]\phi_{i+1}(n+1) \right\} \end{aligned} \tag{19}$$

where $\xi = (x - x_n)/h_{n+1}$ and

$$g_{i,1}(\xi) = \int_0^{\xi} c_{i,n}(s)\, ds. \tag{20}$$

We remark that the $g_{i,1}(1)$ values for $i = 1, \cdots, k+1$ are precisely the coefficients generated in defining p_{n+1} and y_{n+1}. Now, following the earlier construction, we set

$$T(x) = S(x) + [y_{n+1} - S(x_{n+1})]\frac{R(x)}{R(x_{n+1})} \tag{21}$$

with

$$R(x)=\int_{x_n}^{x}(t-x_{n+1})\omega_k(t)\,dt=h_{n+1}^2\{(\xi-1)g_{k+1,1}(\xi)-g_{k+1,2}(\xi)\}.$$

This arises by integration by parts and defining

$$g_{k+1,2}(\xi)=\int_0^{\xi}\int_0^{s_1}c_{k+1,n}(s)\,ds\,ds_1. \tag{22}$$

Then

$$\sigma\equiv\frac{R(x)}{R(x_{n+1})}=\frac{(\xi-1)g_{k+1,1}(\xi)-g_{k+1,2}(\xi)}{-g_{k+1,2}(1)}. \tag{23}$$

Unfortunately, the quantity $g_{k+1,2}(1)$ is not generally available. It can, however, be obtained readily as we shall see in the next section. Although we have not succeeded in eliminating completely the cost of "an extra interpolation," this approach represents a substantial improvement over the procedure described in [8] and discussed earlier.

5. Implementation matters. In order to obtain good numerical continuity with the interpolant $T(x)$ defined by (19), (21), (23), we rearrange the computations as follows: For $x_n<x\leqq x_{n+1}$ and $\xi=(x-x_n)/h_{n+1}$

$$\begin{aligned}T(x)&=(1-\sigma)y_n+\sigma y_{n+1}\\&\quad+h_{n+1}\Big\{[g_{1,1}(\xi)-\sigma g_{1,1}(1)]\phi_1(n+1)\\&\qquad+\sum_{i=1}^{k}([g_{i+1,1}(\xi)-g_{i,1}(\xi)]-\sigma[g_{i+1,1}(1)-g_{i,1}(1)])\phi_{i+1}(n+1)\Big\}\end{aligned} \tag{24}$$

and

$$\begin{aligned}T'(x)&=(y_n-y_{n+1})\mu/h_{n+1}+[c_{1,n}(\xi)+\mu g_{1,1}(1)]\phi_1(n+1)\\&\quad+\sum_{i=1}^{k}([c_{i+1,n}(\xi)-c_{i,n}(\xi)]+\mu[g_{i+1,1}(1)-g_{i,1}(1)])\phi_{i+1}(n+1)\end{aligned} \tag{25}$$

where $\mu=(\xi-1)c_{k+1,n}(\xi)/g_{k+1,2}(1)$. The advantage of the form (24) for obtaining y_n and y_{n+1} numerically as σ tends to 0 or 1 is obvious. When $x=x_n$, we have $\xi=0$ and all $g_{i,j}(0)=0$, hence $\sigma=0$ and $T(x_n)=y_n$. To obtain $T(x_{n+1})=y_{n+1}$, we must have ξ exactly equal to 1, which leads to σ being exactly 1. While $\xi=(x_{n+1}-x_n)/h_{n+1}=1$ in principle, care must be taken to ensure this with computer arithmetic. This is because the step size h_{n+1} is formed first, and then $x_{n+1}=x_n+h_{n+1}$ is defined. Thus the floating point result of $x_{n+1}-x_n$ can differ from the stored $h_{n+1}=\psi_1(n+1)$ by as much as a unit roundoff in the larger of x_n, x_{n+1}, and h_{n+1}. This is easily remedied, however, by redefining $h_{n+1}=x_{n+1}-x_n$ as a variable local to the interpolation routine.

For $T'(x)$ we note that when $\xi=1$, we have all $c_{i,n}(1)=1$ and $\mu=0$, hence $T'(x_{n+1})=\phi_1(n+1)\doteq f_{n+1}$. Although $\phi_1(n+1)$ equals f_{n+1} in principle, this is not usually the case in practice. With Adams codes, there are many differences computed (and retained) in the course of advancing a single step. In order to conserve storage, a compact (and efficient) scheme is used, whereby the various differences created are written on top of the previously used quantities. For example, in the codes we have

referenced, the values defined for $\phi_i(n+1)$ are obtained by the following steps:

$$\phi^e_{k+1}(n+1) = 0,$$
$$\phi^e_i(n+1) = \phi^e_{i+1}(n+1) + \phi^*_i(n), \qquad i = k, k-1, \cdots, 1,$$
$$\phi_{k+1}(n+1) = f_{n+1} - \phi^e_1(n+1),$$
$$\phi_i(n+1) = \phi^e_i(n+1) + \phi_{k+1}(n+1), \qquad i = k, k-1, \cdots 1.$$

Thus it is clear that $\phi_1(n+1)$ will be only approximately equal to f_{n+1} in computer arithmetic. Similarly, we find that for $\xi = 0$, all $c_{i,n}(0) = 0$ except $c_{1,n}(0) = 1$, so that $\mu = 0$ and $T'(x_n) = \phi_1(n+1) - \phi_2(n+1) \doteq f_{n+1} - (f_{n+1} - f_n) \doteq f_n$. Although we would prefer that the interpolant derivatives match the computed function values better, a discrepancy is not as important as when occurring in the solution approximation. Since we do not have a simple way to eliminate this drawback, we shall content ourselves with this interpolant.

The new interpolation algorithm requires the evaluation of $c_{1,n}(\xi), \cdots, c_{k+1,n}(\xi)$, $g_{1,1}(\xi), \cdots, g_{k+1,1}(\xi)$, $g_{k+1,2}(\xi)$, and $g_{k+1,2}(1)$. (Recall the comment about $g_{1,1}(1), \cdots, g_{k+1,1}(1)$ already being available.) The $c_{i,n}(\xi)$ values are simple and inexpensive to obtain; it is the integral quantities $g_{i,j}$ that require some work. In order to achieve numerical continuity, the algorithm for evaluating $g_{i,j}(\xi)$ must be compatible with that used for evaluating $g_{i,j}(1)$ and, in fact, be identical when ξ is taken to be 1. As already mentioned, Krogh [2] devised a scheme for evaluating such coefficients based on recursive evaluation of repeated integrals of the $c_{i,n}(s)$, which was used by Shampine and Gordon [6]. Referring to (14), (15), (20), (22), and defining $g_{i,q}(\xi)$ to be $(q-1)!$ times the q-fold integral of $c_{i,n}(s)$, repeated integration by parts leads to the recursion

$$q_{i,q}(\xi) = [1 + (\xi - 1)\alpha_{i-1}(n+1)]\, g_{i-1,q}(\xi) - \alpha_{i-1}(n+1)\, g_{i-1,q+1}(\xi) \tag{26}$$

for $q = 1, \cdots, k-i+3$ and $i = 2, \cdots, k+1$. From the definition we find that

$$g_{1,q}(\xi) = \frac{\xi^q}{q}, \qquad g_{2,q}(\xi) = \frac{\xi^{q+1}}{q(q+1)}.$$

Evaluating (26) can be visualized as forming a triangular tableau where the column of values $g_{2,q}(\xi)$ for $q = 1, \cdots, k+1$ can be used to initialize the process. By sweeping from the top downward and overwriting previous values (i.e., the jth pass determines $g_{j+1,q}(\xi)$ for q beginning with 1 and proceeding to $k-j+2$), these computations are performed within a single vector of storage. However, an additional working vector is also needed to save the $g_{i,1}(\xi)$ values of interest before they are lost by overwriting in the next pass. Note that $g_{k+1,2}(\xi)$ is the last value computed by the procedure.

When $\xi = 1$, (26) reduces to

$$g_{i,q}(1) = g_{i-1,q}(1) - \alpha_{i-1}(n+1) g_{i-1,q+1}(1)$$

which is exactly the recursion used in [3], [6], [7] to generate the coefficients needed for p_{n+1} and y_{n+1}. These values can now be used in the interpolation algorithm. Unfortunately, we also need $g_{k+1,2}(1)$, and this quantity is not generally available. It can, however, be evaluated readily in the interpolation routine once the proper information has been stored and made available. Basically, the idea is to compute a "diagonal extension to the triangular array" by a simpler recursive operation of the form

$$\delta_{j+1} = g_{j,k-j+2}(1) - \alpha_j(n+1)\delta_j \tag{27}$$

for $j = M, \cdots, k$. Here $M \geqq 2$ is an appropriate index determined by the sequence of lowering and raising of the order of the integration method and changing of the step size. The δ_j can be likened to entries $g_{j,k-j+3}(1)$ in the triangular tableau. The $g_{j,k-j+2}(1)$ are previously computed values which have been stored during the computation of y_{n+1}. Actually, they represent a portion of the diagonal entries of the triangular array diagram corresponding to order k. When $M = 2$, (27) is started off with $\delta_2 = 1/(k+1)(k+2)$; otherwise, $\delta_M = g_{M,k-M+3}(1)$, a value presumed to have been stored. Finally, the item of interest $g_{k+1,2}(1) = \delta_{k+1}$. Notice that the values $g_{1,1}(1), \cdots, g_{k+1,1}(1)$ are not regenerated, and so a computational savings is achieved.

We remark that it will not always be necessary to compute $g_{k+1,2}(1)$ in the interpolation routine. Rather, advantage can be taken of the fact that in the process of computing some $g_{i,1}(1)$ in the main recursion scheme, the quantity $g_{i-1,2}(1)$ has also been computed and can be stored. Thus, the needed $g_{k+1,2}(1)$ value will already be available in those circumstances when the current order k is smaller than on an earlier step taken, while the step size has remained unchanged throughout this sequence of events.

For further details regarding the implementation see [10]. The algorithmic structure is similar to that which is described in [6]. Reference [10] also contains a listing of the interpolation routine along with a correction to the STEP code given in [6] and being used in DEPAC [7]. (In certain circumstances, higher order integration coefficients $g_{i,1}$ are not formed correctly.)

6. Numerical results. In this section we provide some computational statistics regarding the various interpolants discussed earlier. Although our primary emphasis has been on obtaining a smoother interpolant, we also present some results reflecting on the accuracy of the interpolated values. We display statistics gathered from three problems taken out of a larger test collection (11 problems in all) which we examined. These are problems A1, B4, and D1 in [1], except for a longer integration interval for A1.

Problem 1. $y' = -y$, $0 \leqq x \leqq 50$, $y(0) = 1$.

Problem 2. Integral surface of a torus, $0 \leqq x \leqq 20$,

$$\begin{aligned} y_1' &= -y_2 - y_1 y_3 / r, & y_1(0) &= 3, \\ y_2' &= y_1 - y_2 y_3 / r, & y_2(0) &= 0, \\ y_3' &= y_1 / r, & y_3(0) &= 0. \end{aligned}$$

Problem 3. Newton's equations of motion, $0 \leqq x \leqq 20$,

$$\begin{aligned} y_1'' &= -y_1 / r^3, & y_1(0) &= 0.9, & y_1'(0) &= 0, \\ y_2'' &= -y_2 / r^3, & y_2(0) &= 0, & y_2'(0) &= \sqrt{11/9}. \end{aligned}$$

In the last two problems r is defined by $r^2 = y_1^2 + y_2^2$.

We solved these problems for several tolerances using a mixed error test in which both the relative and absolute error parameters are taken to be the same—10^{-2}, 10^{-4}, 10^{-6}, or 10^{-8}. The results were obtained on a CYBER 855 for which the unit roundoff value (relative machine precision) is approximately $7.1 * 10^{-15}$.

Let $I_n(x)$ denote one of the interpolants, $y_I(x)$, $\eta(x)$, or $T(x)$, to be evaluated for $x_n \leqq x \leqq x_{n+1}$. Table 1 shows the relative discrepancies $\delta = \max_n \delta_n$, $\delta' = \max_n \delta_n'$ in achieving the continuity conditions at the ends of the interpolation intervals, where

$$\delta_n = \max \left(|I_n(x_n) - y_n| / |y_n|, |I_n(x_{n+1}) - y_{n+1}| / |y_{n+1}| \right),$$

and similarly for δ'_n. In the case of systems we display the values from the worst behaving components. The computational results were produced by using the INTRP routine for $y_I(x)$ and $y'_I(x)$, (18) and its corresponding derivative form to define $\eta(x)$ and $\eta'(x)$, and (24) and (25) for $T(x)$ and $T'(x)$. Table 1 shows that $T(x)$ is globally continuous and that the (computational) jumps in $T'(x)$ at the mesh points are small indeed. The jumps occurring in y_I and η' are quite visible, as expected. In the tables, the notation 1.2(−3) means $1.2*10^{-3}$.

TABLE 1
Relative discrepancies δ in satisfying continuity conditions.

Tol.	y_I	y'_I	η	η'	T	T'
			Problem 1			
10^{-2}	2.9 (+3)	1.7 (−11)	3.6 (−11)	57.	0.	2.4 (−12)
10^{-4}	26.	2.3 (−13)	5.3 (−13)	15.	0.	7.1 (−14)
10^{-6}	24.	5.6 (−13)	6.2 (−13)	17.	0.	1.4 (−13)
10^{-8}	22.	2.2 (−13)	3.9 (−11)	11.	0.	8.4 (−14)
			Problem 2			
10^{-4}	2.2 (−4)	2.7 (−12)	2.2 (−12)	.27	0.	2.9 (−14)
10^{-6}	5.2 (−5)	7.2 (−12)	7.1 (−12)	3.7 (−3)	0.	1.8 (−14)
10^{-8}	1.3 (−7)	2.5 (−11)	9.0 (−12)	6.0 (−2)	0.	2.1 (−14)
			Problem 3			
10^{-4}	4.1 (−3)	5.6 (−12)	4.9 (−5)	2.9 (−2)	0.	2.2 (−14)
10^{-6}	4.4 (−5)	1.2 (−11)	2.0 (−8)	5.0 (−4)	0.	8.8 (−14)
10^{-8}	2.7 (−5)	4.2 (−11)	4.2 (−11)	3.6 (−5)	0.	3.0 (−13)

Next, we display some statistics aimed at comparing the accuracies of the various interpolants. Again, let $I_n(x)$ denote one of the interpolants for $x_n \leqq x \leqq x_{n+1}$ and define

$$t_{i,n} = x_n + \frac{i}{10} h_{n+1} \quad \text{for } i = 1, \cdots, 9,$$

$$\varepsilon_n = \max\left(|y(x_n) - y_n|, |y(x_{n+1}) - y_{n+1}|\right),$$

$$\xi_{i,n} = |y(t_{i,n}) - I_n(t_{i,n})|,$$

$$\beta_n = \sqrt{\frac{1}{9}\sum_{i=1}^{9} \xi_{i,n}^2},$$

N = total number of steps taken.

The interpolation error measures presented in Table 2 are the averages of the (β_n/ε_n) and $(\beta'_n/\varepsilon'_n)$ quantities based on the number of steps taken.

These computations confirmed the expected behavior of the various interpolants. Values of $T(x)$ offer a distinct improvement over those of $y_I(x)$, certainly in the smoothness and possibly in a slight reduction in the error of the interpolated values. There appears to be a slight degradation in the accuracy of interpolated derivative values $T'(x)$. Because we have examined other implementations for $T'(x)$ and found the same behavior, we are inclined to think this is due to the fact that the degree of

the $T(x)$ polynomial is one higher than that of $y_I(x)$. In general, errors in $T(x)$ tend to be rather uniform throughout each step interval, but the errors in $T'(x)$ on the interior of each step interval can be an order of magnitude worse than at the neighboring mesh points.

TABLE 2
Error measures.

Tol.	N	y_I	y_I'	η	η'	T	T'
			Problem 1				
10^{-2}	31	1.3	0.85	0.45	1.1	0.65	1.1
10^{-4}	46	1.2	1.9	0.55	1.5	0.69	2.5
10^{-6}	74	1.1	1.9	0.68	2.1	0.76	1.8
10^{-8}	91	1.1	3.1	0.69	4.1	0.76	3.7
			Problem 2				
10^{-4}	101	0.97	1.6	0.91	2.2	0.89	1.9
10^{-6}	144	0.99	1.6	0.97	2.5	0.96	2.4
10^{-8}	182	0.97	2.5	0.94	3.9	0.95	4.9
			Problem 3				
10^{-4}	86	0.93	1.2	0.86	1.6	0.86	1.8
10^{-6}	135	0.93	1.3	0.87	2.1	0.87	2.0
10^{-8}	208	0.93	1.7	0.93	1.5	0.93	1.9

REFERENCES

[1] T. E. HULL, W. H. ENRIGHT, B. M. FELLEN AND A. E. SEDGWICK, *Comparing numerical methods for ordinary differential equations*, SIAM J. Numer. Anal., 9 (1972), pp. 603–637.

[2] F. T. KROGH, *Changing step size in the integration of differential equations using modified divided differences*, Proc. Conference on the Numerical Solution of Ordinary Differential Equations, Lecture Notes in Mathematics 362, Springer-Verlag, New York, 1974.

[3] ———, *Preliminary usage documentation for the variable order integrators* SODE *and* DODE, Computing Memorandum 399, Jet Propulsion Laboratory, Pasadena, CA, 1975.

[4] ———, *Private communication.*

[5] L. F. SHAMPINE, *Local error estimation in Adams codes*, Rept. SAND82-2998, Sandia National Laboratories, Albuquerque, NM, 1982.

[6] L. F. SHAMPINE AND M. K. GORDON, *Computer Solution of Ordinary Differential Equations: The Initial Value Problem*, W. H. Freeman, San Francisco, 1975.

[7] L. F. SHAMPINE AND H. A. WATTS, DEPAC—*Design of a user oriented package of ODE solvers*, Rept. SAND79-2374, Sandia National Laboratories, Albuquerque, NM, 1979.

[8] ———, *A smoother interpolant for* DE/STEP, INTRP *and* DEABM, Rept. SAND83-1226, Sandia National Laboratories, Albuquerque, NM, 1983.

[9] H. J. STETTER, *Interpolation and error estimation in Adams* PC-*codes*, SIAM J. Numer. Anal., 16 (1979), pp. 311–323.

[10] H. A. WATTS, *A smoother interpolant for* DE/STEP, INTRP and DEABM: II, Rept. SAND84-0293, Sandia National Laboratories, Albuquerque, NM, 1984.

SIAM J. SCI. STAT. COMPUT.
Vol. 7, No. 1, January 1986

023

TIMELY COMMUNICATIONS

This is the first paper to be published under the new "timely communications" policy for the SIAM Journal on Scientific and Statistical Computing. Papers that have significant timely content and do not exceed five pages automatically will be considered for a separate section of this journal with an accelerated reviewing process. It will be possible for the note to appear approximately six months after the date of acceptance. The editors are pleased to launch this section with a note on the use of a new computer organization that is likely to be the start of a revolution in scientific and statistical computation.

IMPLEMENTING DENSE LINEAR ALGEBRA ALGORITHMS USING MULTITASKING ON THE CRAY X-MP-4 (OR APPROACHING THE GIGAFLOP)*

JACK J. DONGARRA† AND TOM HEWITT‡

Abstract. This note describes some experiments on simple, dense linear algebra algorithms. These experiments show that the CRAY X-MP is capable of small-grain multitasking arising from standard implementations of *LU* and Cholesky decomposition. The implementation described here provides the "fastest" execution rate for *LU* decomposition, 718 MFLOPS for a matrix of order 1000.

Introduction. Over the past few months we have been experimenting with some simple linear algebra algorithms on the CRAY X-MP-4 multiprocessor. The CRAY X-MP family is a general-purpose multiprocessor system. It inherits the basic vector functions of CRAY-1S, with major architectural improvements for each individual processor. The CRAY X-MP-4 system is a four-processor model housed in a physical chassis identical to the CRAY-1S. The system can be used to perform simultaneous scalar and vector processing of either independent job streams or independent tasks within one job. Hardware in the X-MP enables multiple processors to be applied to a single Fortran program in a timely and coordinated manner.

All processors share a central bipolar memory (of up to 16 million words), organized in 64 interleaved memory banks. Each processor has four memory ports: two for vector fetches, one for vector stores, and one for independent I/O operations. In other words, the total memory bandwidth of the four processors is up to sixteen times that of the CRAY-1S system.

This note describes results obtained from three experiments: *LU* decomposition based on matrix-vector operations, *LU* based on a "best" implementation for the architecture, and an implementation of Cholesky decomposition based on matrix-vector operations.

***LU* decomposition.** The versions of *LU* and Cholesky factorization, used here, are based on matrix-vector modules that allow for a high level of granularity, permitting high performance in a number of different environments, see [2].

The algorithm designed to give the "best" performance on the X-MP architecture is worth noting. It is based on standard Gaussian elimination with partial pivoting. The algorithm is organized such that it zeros out three columns (below the diagonal)

* Received by the editors August 17, 1985.

† Mathematics and Computer Science Division, Argonne National Laboratory, Argonne, Illinois 60439. The work of this author was supported in part by the Applied Mathematical Sciences subprogram of the Office of Energy Research, U.S. Department of Energy, under contract W-31-109-Eng-38.

‡ CRAY Research Inc., Chippewa Falls, Wisconsin 54701.

of the matrix, then applies these transformations to the rest of the matrix. The application of the transformations to the remainder of the matrix is split up among the processors. An assembly language kernel is used to apply the three pivot rows for simultaneous row operations. This reduces memory traffic and allows a single processor of an X-MP to obtain its theoretical maximum sustainable computation rate of 198 MFLOPS. The assembly language kernel was multitasked in the experiment among two, three, and four CPUs. Fortran-callable assembly language synchronization subroutines were used. These require less than half a microsecond for synchronization.

The process of finding the three pivot rows was not multitasked. However, it was written as an assembly language kernel to reduce overhead for small problems. As the factorization proceeds, the size of the relevant vector and submatrix decreases. The final 15×15 block of the reduction was performed by an unrolled [2] version of standard Gaussian elimination. This portion is entirely single-threaded Fortran and runs at two to three times the speed of the Fortran which has not been unrolled. Synchronization was accomplished through a fork-and-join mechanism using the cluster (shared) registers of the X-MP.

Listed in Table 1 are the results of the experiments; speedups range from 1.3 on a problem of size 50×50 to 3.8 for a matrix of order 1000×1000. The performance ranges from 97 MFLOPS to an impressive 718 MFLOPS for four processors on a 1000×1000 matrix. For small problems, startup times dominate the overall performance. It is interesting to note that a system of equations of order 1000 can now be factored and solved in under a second! (The same problem on a VAX 11/780 would take roughly two hours to complete.)

TABLE 1
High-performance LU decomposition.

	MFLOPS #processors				Speedup over 1 processor #processors		
Order	1	2	3	4	2	3	4
50	97	124	135	145	1.28	1.39	1.49
100	145	230	281	325	1.64	1.94	2.24
200	172	330	426	526	1.81	2.47	3.05
400	183	353	507	652	1.93	2.77	3.56
600	186	364	535	689	1.96	2.87	3.70
1000	188	372	550	718	1.98	2.92	3.81

For comparison we give in Table 2 the results for *LU* decomposition based on matrix-vector operations. The parallelism here is gained by simply splitting the matrix-vector operation across the processors (see [1] for details on the algorithm). The

TABLE 2
LU based on matrix-vector operations.

Order	MFLOPS (4 processors)	Ratio of algorithms from Table 1/Table 2
50	57	2.54
100	167	1.95
200	343	1.53
400	537	1.21
600	608	1.13
1000	675	1.06

matrix-vector modules have been coded in assembly language. The same number of operations is performed here as in the version that gives "best" performance.

As we can see, for lower order matrix problems, the high-performance algorithm is far more efficient. For large problems, however, there is not too much difference; for the matrix of order 1000 there is only a difference of 6% in the running time. This fact shows one of the strengths in using the matrix-vector design for algorithms of this nature. The matrix-vector modules can be easily changed as we go to a different architecture, but the basic algorithm is unaltered.

Cholesky decomposition. A version of Cholesky decomposition for a symmetric positive definite matrix was implemented on the CRAY X-MP-4 based on matrix-vector routines (see [2] for algorithm details). Table 3 gives the performance of that routine when run on 1 and 4 processors. In addition, the algorithm was reorganized to compute information necessary to perform four steps of the decomposition during the same step. This results in four independent "full size" matrix-vector multiplications. This information is listed in the last column of Table 3.

TABLE 3
Cholesky decomposition based on matrix-vector operations.

Order	MFLOPS 1 processor	MFLOPS 4 processors (MV split)	Speedup	MFLOPS 4 processors (4 MV ops)
50	59	52	.88	97
100	117	154	1.32	264
200	163	345	2.12	460
400	184	544	2.96	631
600	189	623	3.30	683
1000	193	689	3.58	733

Synchronization was accomplished via the X-MP shared registers and semaphores. Multiprocessing overhead in this case is largely the result of code changes and of the breaking of a large piece of work into smaller pieces—each of which has essentially the same startup time as the original large piece.

The apparent overhead for multitasking small problems is largely caused by the following items:

1. Single-threaded portions of work. Finding pivot rows, scaling rows of the matrix, the scalar square root in Cholesky decomposition—these are all of great importance in the current study.
2. Less parallel work per processor in the computational kernel. Even in scalar operations the X-MP uses a considerable amount of parallelism. In vector mode, 100 floating-point operations can be performed in the time it takes to call a subroutine and scores of flops in the time required to initialize a DO loop.
3. Additional memory-bank conflicts. The single CPU times were run with no activity in the other three CPUs. When four processors are active, additional conflicts will occur, though in this case the effect is small as the matrix-vector operations are conflict-insensitive on the X-MP.
4. Synchronization overhead. This is dominated by the time a CPU is waiting for all of its vector memory references to be completed. Thus, it is generally wise to complete all vector memory references before synchronization, as it eliminates the possibility of an inter-CPU memory race condition.

Conclusions. The CRAY X-MP is capable of small granularity multitasking. For large problems that are both vectorized and parallelized, performance of the X-MP-4 should be in the range of 400 to 700 MFLOPS. For smaller problems the startup time of the parallel processes can dominate the execution time. This feature becomes more important as more processors are applied to the problem.

Note. Since the time this work was conducted, CRAY Inc. Research has introduced micro-tasking, which provides a mechanism for the user to conveniently, and with low overhead, exploit small-granularity parallelism from Fortran programs with compiler directives.

REFERENCES

[1] STEVE S. CHEN, JACK J. DONGARRA AND CHRISTOPHER C. HSIUNG, *Multiprocessing for linear algebra algorithms on the* CRAY X-MP-2: *Experiences with small granularity*, J. Parallel and Distributed Computing, 1 (1984), pp. 22–31.

[2] JACK J. DONGARRA AND STANLEY C. EISENSTAT, *Squeezing the most out of an algorithm in* CRAY *Fortran*, ACM Trans. Math. Software, 10 (1984), pp. 221–230.

SIAM J. SCI. STAT. COMPUT.
Vol. 7, No. 2, April 1986

A DIRECT METHOD FOR COMPUTING HIGHER ORDER FOLDS*

ZHONG-HUA YANG† AND H. B. KELLER‡

Abstract. We consider the computation of higher order fold or limit points of two parameter-dependent nonlinear problems. A direct method is proposed and an efficient implementation of the direct method is presented. Numerical results for the thermal ignition problem are given.

Key words. two-parameter nonlinear problems, simple limit points, higher order fold points, double extended systems, Newton's methods

1. Introduction. This paper is concerned with the computation of special kinds of singular points, which are called (simple) higher order fold or limit points. They may arise in two parameter nonlinear problems of the form

$$f(\lambda, \mu, x) = 0 \tag{1.1}$$

where $\lambda, \mu \in \mathbb{R}$, $x \in X$, a Banach space, and f is a C^3 mapping from $\mathbb{R} \times \mathbb{R} \times X \to X$. A problem in the theory of thermal ignition is one such problem [1], [2], [3] which we treat. Two parameter nonlinear problems arise in many other physical applications [6], [8]. The problem in thermal ignition has the form

$$\begin{aligned} Lx &= h(\lambda, \mu, x), \\ Bx &= 0 \end{aligned} \tag{1.2}$$

where L is a uniformly elliptic differential operator, B is a boundary operator, λ is a rate parameter, μ is related to the activation energy, and h has the form

$$h(\lambda, \mu, x) = \lambda \exp\left(\frac{x}{1+\mu x}\right). \tag{1.3}$$

The solution x is the dimensionless temperature. Of particular interest are the values λ_0 and μ_0 which correspond to the loss of criticality in the exothermic reaction described by (1.2). These values correspond to "folds" or "limit" points.

Spence and Werner [10] proved that a cubic fold point (λ_0, μ_0, x_0) of f with regard to λ corresponds to a quadratic fold point $(\lambda_0, \mu_0, x_0, \phi_0)$ of an extended system, F, of f, provided certain conditions are satisfied. They located the cubic fold by using a continuation method [5] to compute the quadratic fold point of an "extended system". The main idea in this paper is to reduce a problem with cubic folds to a *regular* problem by using a larger "double extended system". We also present an efficient implementation for solving the larger "double extended system". Thus we locate a cubic fold directly, without any continuation. Related techniques in [11] show how to find isolas and cusps using extended systems.

In § 2 we give a brief review of simple fold points, the degree of a fold, and extended systems. The main idea of our treatment of higher degree fold points is contained in Theorem 2.1. The efficient implementation of Newton's method is given in § 3. In § 4 we give numerical results.

* Received by the editors May 15, 1984, and in revised form February 25, 1985. This work was supported by the U.S. Department of Energy under contract EY-76-S-03-0767, project agreement no. 12, and the U.S. Army Research Office under contract DAAG 29-78-C-0011.

† Department of Mathematics, Shanghai University of Science and Technology, The People's Republic of China.

‡ Applied Mathematics, California Institute of Technology, Pasadena, California 91125.

2. Folds, degree of a fold, and extended systems. First we review some of the definitions and the main results about folds. Let Y be a Banach space and consider the C^3 mapping

$$F: \begin{cases} \mathbb{R} \times Y \to Y, \\ (\mu, y) \to F(\mu, y). \end{cases}$$

We use the notation $F_\mu(a)$, $F_{\mu\mu}(a)$, $F_y(a)$, $F_{\mu y}(a)$, $F_{yy}(a)$, $F_{yyy}(a)$, $\cdots$, to denote the partial Fréchet-derivatives of F at $a = (\mu, y) \in \mathbb{R} \times Y$. We denote the dual pairing of $y \in Y$ and $\psi \in Y^*$ by ψy.

DEFINITION 2.1. A point $a_0 = (\mu_0, y_0) \in \mathbb{R} \times Y$ is a *fold point* of F (with respect to μ) if

$$F(a_0) = 0, \tag{2.1}$$

$$\text{Ker } F_y(a_0) \neq 0, \tag{2.2}$$

$$F_\mu(a_0) \notin \text{Range } F_y(a_0). \tag{2.3}$$

DEFINITION 2.4. A fold point a_0 is a *simple fold* of F if in addition to (2.1)-(2.3)

$$\dim \text{Ker } F_y(a_0) = \text{codim Range } F_y(a_0) = 1. \tag{2.4a}$$

In this case there exist nontrivial $\phi_0 \in Y$ and $\psi_0 \in Y^*$ such that

$$\text{Ker } F_y(a_0) = \{\alpha\phi_0 | \alpha \in \mathbb{R}\}, \tag{2.4b}$$

$$\text{Range } F_y(a_0) = \{y \in Y | \psi_0 y = 0\}. \tag{2.4c}$$

As is well known, near a simple fold point a_0, the zero set of F, denoted by $F^{-1}(0)$, is a smooth curve

$$\Gamma: F^{-1}(0) \cap U = \{[\mu(s), y(s)] \| s - s_0| \leqq \delta\}.$$

Here U is a neighborhood of the fold point a_0, δ is positive and $\mu(\cdot)$, $y(\cdot)$ are smooth mappings satisfying

$$\mu(s_0) = \mu_0, \quad y(s_0) = y_0, \quad |\mu'(s)| + \|y'(s)\| > 0.$$

Along Γ we have the identity

$$F(\mu(s), y(s)) = 0 \tag{2.5a}$$

and we can differentiate it with respect to s as many times as the smoothness of F allows. In place of $F_\mu(\mu(s), y(s)), \cdots, F_{yyy}(\mu(s), y(s))$, we shall write $F_\mu(s), \cdots, F_{yyy}(s)$. Then we get by differentiating in (2.5a)

$$F_\mu(s)\mu'(s) + F_y(s)y'(s) \equiv 0, \qquad |s - s_0| < \delta. \tag{2.5b}$$

Obviously (2.3) and (2.4) imply from (2.5) evaluated at s_0, that

$$\mu'(s_0) = 0, \tag{2.6a}$$

$$y'(s_0) = \alpha\phi_0 \quad \text{for some } \alpha \in \mathbb{R},\ \alpha \neq 0 \text{ (say } \alpha = 1). \tag{2.6b}$$

The first nonvanishing derivative of $\mu(s)$ at s_0 determines the "degree" of the fold. We formalize this in

DEFINITION 2.7. A simple fold point $a_0 \in \mathbb{R} \times Y$ is said to have degree m if $d^p\mu(s_0)/ds^p = 0$ for all $p < m$ and $d^m\mu(s_0)/ds^m \neq 0$.

The result in (2.6) implies that *all simple folds have degree two or greater.* To actually find the degree of a simple fold we need only differentiate further in (2.5a)

or (2.5b) and find the first nonvanishing derivative of $\mu(s)$ at $s_0 = s$. Thus from (2.5b) we obtain

$$F_\mu(s)\mu''(s) + F_y(s)y''(s) + F_{\mu\mu}(s)\mu'(s)\mu'(s) + 2F_{\mu y}(s)\mu'(s)y'(s) + F_{yy}(s)y'(s)y'(s) = 0, \qquad |s - s_0| < \delta. \tag{2.5c}$$

With a simple fold at $s = s_0$ we use (2.6) and (2.4) in the above, apply ψ_0 and note that (2.3) and (2.4c) imply $\psi_0 F_\mu(s_0) \neq 0$ to get

$$\mu''(s_0) = \frac{\psi_0 F_{yy}(s_0)\phi_0\phi_0}{\psi_0 F_\mu(s_0)}. \tag{2.7a}$$

So *a simple fold is of degree two if and only if*

$$\psi_0 F_{yy}(s_0)\phi_0\phi_0 \neq 0. \tag{2.7b}$$

We introduce the *extended* or *inflated mapping*

$$G: \begin{cases} \mathbb{R} \times Y \times Y \to \mathbb{R} + Y \times Y, \\ (\mu, y, \phi) \to \begin{pmatrix} l\phi - 1 \\ F(\mu, y) \\ F_y(\mu, y)\phi \end{pmatrix}, \end{cases} \tag{2.8}$$

where $l \in Y^*$ is chosen later on in § 3. It is not difficult to show (see [10, Thm. 2.1]) that if

$$G(\mu_0, y_0, \phi_0) = 0 \tag{2.9a}$$

and (μ_0, y_0) is a simple fold of F of degree two, then

$$DG^0 = \begin{bmatrix} 0 & 0 & l \\ F^0_\mu & F^0_y & 0 \\ F^0_{\mu y}\phi_0 & F^0_{yy}\phi_0 & F^0_y \end{bmatrix} \tag{2.9b}$$

is nonsingular. As a consequence, the system $G(\mu, y, \phi) = 0$ can be solved by Newton's method in some neighborhood of (μ_0, y_0, ϕ_0). $G(\mu, y, \phi) = 0$ is called an *extended system* for $F(\mu, y) = 0$. Various kinds of extended systems have been used by different authors [6], [9], [10] following their introduction by Keener and Keller in [4].

We next consider two parameter nonlinear problems involving the smooth mapping

$$f: \begin{cases} \mathbb{R} \times \mathbb{R} \times X \to X, \\ (\lambda, \mu, x) \to f(\lambda, \mu, x). \end{cases} \tag{2.10}$$

For some fixed value of $\mu = \mu_0$ we assume that

$$g(\lambda, x) \equiv f(\lambda, \mu_0, x) = 0$$

has a simple fold point (λ_0, x_0) with respect to λ, according to Definitions 2.1 and 2.4.

We introduce, in exact analogy with (2.8), an extended system for $f(\lambda, \mu, x) = 0$

$$F(\lambda, \mu, x, \phi) \equiv \begin{pmatrix} l\phi - 1 \\ f(\lambda, \mu, x) \\ f_x(\lambda, \mu, x)\phi \end{pmatrix} = 0. \tag{2.11}$$

Here F is a mapping from $\mathbb{R} \times \mathbb{R} \times X \times X$ to $\mathbb{R} \times X \times X$. If we denote $Y \equiv \mathbb{R} \times X \times X$ and $y \equiv (\lambda, x, \phi)^T \in Y$, the extended system $F(\lambda, \mu, x, \phi) = 0$ can be written as $F(\mu, y) = 0$.

Our main idea is to extend this extended system, $F(\mu, \lambda) = 0$, *again* and to get the *doubly extended system*

$$H(\mu, y, \Phi) \equiv \begin{pmatrix} L\Phi - 1 \\ F(\mu, y) \\ F_y(\mu, y)\Phi \end{pmatrix} = 0 \tag{2.12}$$

where $\Phi \in Y$, $L \in Y^*$. A specific $L \equiv (0, l, 0)$ will be chosen later on in order to simply (2.12). Using this system we obtain

THEOREM 2.13. *Assume* $F_\mu^0 \notin \text{Range } F_y^0$. *Then a third degree simple fold point* (λ_0, μ_0, x_0) *of* $f(\lambda, \mu, x)$ *with respect to* λ *corresponds to a regular solution* $(\lambda_0, \mu_0, x_0, \phi_0, v_0)$ *of the inflated system*

$$\begin{pmatrix} l\phi - 1 \\ lv \\ f(\lambda, \mu, x) \\ f_x(\lambda, \mu, x)\phi \\ f_{xx}\phi\phi + f_x v \end{pmatrix} = 0. \tag{2.14}$$

Proof. According to Spence and Werner [10, Thm. 3.1] a third degree fold point (λ_0, μ_0, x_0) of $f(\lambda, \mu, x)$ with respect to λ corresponds to a second degree fold point $(\mu_0, y_0) = (\lambda_0, \mu_0, x_0, \phi_0)$ of $F(\mu, y)$ in (2.11) with respect to μ provided $F_\mu^0 \notin \text{Range } F_y^0$. Further applying [10, Thm. 2.1] to $F(\mu, y)$ we get that a second degree fold point (μ_0, y_0) of $F(\mu, y)$ with respect to μ corresponds to a regular solution (μ_0, y_0, Φ_0) of the doubly extended system $H(\mu, y, \Phi) = 0$ in (2.12) i.e. $H(\mu, y, \Phi) = 0$ is a regular system at (μ_0, y_0, Φ_0), provided $L\Phi = 1$.

Next we show that the double extended system $H(\mu, y, \Phi) = 0$ is equivalent to (2.14) for a particular L. Let

$$\Phi = \begin{pmatrix} \sigma \\ u \\ v \end{pmatrix}$$

and choose $L = (0, l, 0)$. Then (2.12) becomes

$$L\Phi - 1 = lu - 1 = 0, \tag{2.15a}$$

$$F(\mu, y) = \begin{pmatrix} l\phi - 1 \\ f(\lambda, \mu, x) \\ f_x(\lambda, \mu, x)\phi \end{pmatrix} = 0, \tag{2.15b, c, d}$$

$$F_y(\mu, y)\Phi = \begin{pmatrix} 0 & 0 & l \\ f_\lambda & f_x & 0 \\ f_{\lambda x}\phi & f_{xx}\phi & f_x \end{pmatrix} \begin{pmatrix} \sigma \\ u \\ v \end{pmatrix} = \begin{pmatrix} lv \\ \sigma f_\lambda + f_x u \\ \sigma f_{\lambda x}\phi + f_{xx}\phi u + f_x v \end{pmatrix} = 0. \tag{2.15e, f, g}$$

By Definition 2.1, we know that $f_\lambda \notin \text{Range } f_x$ at a fold point. From (2.15f): $\sigma f_\lambda + f_x u = 0$. We thus get $\sigma = 0$ and then $u \in N(f_x)$. From (2.15d) and Definition 2.4 of a simple fold we have $u = \alpha\phi_0$. Using this u in (2.15a) we get $\alpha = 1$ in order to satisfy (2.15b). The solution of (2.15) is thus

$$\mu = \mu_0, \quad y = y_0 \equiv \begin{pmatrix} \lambda_0 \\ x_0 \\ \phi_0 \end{pmatrix}, \quad \Phi = \Phi_0 \equiv \begin{pmatrix} 0 \\ \phi_0 \\ v_0 \end{pmatrix}. \tag{2.16}$$

Here v_0 satisfies

$$lv_0 = 0, \qquad f^0_{xx}\phi_0\phi_0 + f^0_x v_0 = 0,$$

and μ_0 and y_0 satisfy $F(\mu_0, y_0) = 0$. This shows that $(\lambda_0, \mu_0, x_0, \phi_0, v_0)$ is also a solution of (2.14).

On the other hand, if we know the solution $(\lambda_0, \mu_0, x_0, \phi_0, v_0)$ of (2.14), we can easily construct a solution of (2.15) as in (2.16). Actually we have reduced (2.15) to (2.14), which is also a regular system, by choosing the particular $L \equiv (0, l, 0)$. □

Since the inflated system (2.14) is regular, we can solve it by using Newton's method. The solution of (2.14) is just the third degree fold point with respect to λ of the original two parameter nonlinear problem, $f(\lambda, \mu, x) = 0$.

We now turn to the efficient solutions of (2.14).

3. Efficient implementation of Newton's method. After discretization (2.14) becomes a finite-dimensional nonlinear system. Let $x, \phi, v \in E^n$, the dimension of (2.14) is actually $3n-2$ because we can choose $l\phi = \phi_r = 1$, $lv = v_r = 0$, where r is a positive integer in $1 \leqq r \leqq n$. For convenience we shall choose $r = 1$ and the discretized system of (2.14) is denoted by the same notation. Newton's method applied to (2.14) yields:

$$\begin{bmatrix} 0 & 0 & 0 & l & 0 \\ 0 & 0 & 0 & 0 & l \\ f_\mu & f_\lambda & f_x & 0 & 0 \\ f_{\mu x}\phi & f_{\lambda x}\phi & f_{xx}\phi & f_x & 0 \\ f_{\mu xx}\phi\phi + f_{\mu x}v & f_{\lambda xx}\phi\phi + f_{\lambda x}v & f_{xxx}\phi\phi + f_{xx}v & 2f_{xx}\phi & f_x \end{bmatrix}^{(\nu)} \begin{bmatrix} \mu^{\nu+1} - \mu^\nu \\ \lambda^{\nu+1} - \lambda^\nu \\ x^{\nu+1} - x^\nu \\ \phi^{\nu+1} - \phi^\nu \\ v^{\nu+1} - v^\nu \end{bmatrix}$$

(3.1)

$$= \begin{bmatrix} -l\phi + 1 \\ -lv \\ -f \\ -f_x\phi \\ -f_{xx}\phi\phi - f_x v \end{bmatrix}^{(\nu)}.$$

Here superscript (ν) denotes evaluation of the coefficient matrix and the right-hand side at $(\mu^\nu, \lambda^\nu, x^\nu, \phi^\nu, v^\nu)$. The starting value is $(\mu^0, \lambda^0, x^0, \phi^0, v^0)$. We write $\delta\mu^\nu = \mu^{\nu+1} - \mu^\nu$, $\delta\lambda^\nu = \lambda^{\nu+1} - \lambda^\nu$, $\delta x^\nu = x^{\nu+1} - x^\nu$, $\delta\phi^\nu = \phi^{\nu+1} - \phi^\nu$, $\delta v^\nu = v^{\nu+1} - v^\nu$.

In expanded form, and with the superscripts of $(\delta\mu, \delta\lambda, \delta x, \delta\phi, \delta v)$ suppressed, (3.1) can be written as

$$\delta\phi_1 = 0, \tag{3.2}$$

$$\delta\nu_1 = 0, \tag{3.3}$$

$$\mathbb{A}\delta x + \delta\lambda \cdot \mathbb{D}_1 + \delta\mu \cdot \mathbb{D}_2 = \mathbb{C}_1, \tag{3.4}$$

$$\mathbb{A}\delta\phi + \delta\lambda \cdot \mathbb{D}_3 + \delta\mu \cdot \mathbb{D}_4 + \mathbb{B}_1\delta x = \mathbb{C}_2, \tag{3.5}$$

$$\mathbb{A}\delta v + \delta\lambda \cdot \mathbb{D}_5 + \delta\mu \cdot \mathbb{D}_6 + 2\mathbb{B}_1\delta\phi + \mathbb{B}_2\delta x = \mathbb{C}_3. \tag{3.6}$$

Here we have introduced

$$\mathbb{A} = f_x(\lambda^\nu, \mu^\nu, x^\nu), \qquad \mathbb{B}_1 = f_{xx}(\lambda^\nu, \mu^\nu, x^\nu)\phi^\nu,$$

$$\mathbb{B}_2 = f_{xxx}(\lambda^\nu, \mu^\nu, x^\nu)\phi^\nu\phi^\nu + f_{xx}(\lambda^\nu, \mu^\nu, x^\nu)v^\nu,$$

$$
\begin{aligned}
&\mathbb{D}_1 = f_\lambda(\lambda^\nu, \mu^\nu, x^\nu), \qquad \mathbb{D}_2 = f_\mu(\lambda^\nu, \mu^\nu, x^\nu),\\
&\mathbb{D}_3 = f_{\lambda x}(\lambda^\nu, \mu^\nu, x^\nu)\phi^\nu, \qquad \mathbb{D}_4 = f_{\mu x}(\lambda^\nu, \mu^\nu, x^\nu)\phi^\nu,\\
&\mathbb{D}_5 = f_{\lambda xx}(\lambda^\nu, \mu^\nu, x^\nu)\phi^\nu\phi^\nu + f_{\lambda x}(\lambda^\nu, \mu^\nu, x^\nu)v^\nu,\\
&\mathbb{D}_6 = f_{\mu xx}(\lambda^\nu, \mu^\nu, x^\nu)\phi^\phi\gamma^\nu + f_{\mu x}(\lambda^\nu, \mu^\nu, x^\nu)v^\nu,\\
&\mathbb{C}_1 = -f(\lambda^\nu, \mu^\nu, x^\nu), \qquad \mathbb{C}_2 = -f_x(\lambda^\nu, \mu^\nu, x^\nu)\phi^\nu,\\
&\mathbb{C}_3 = -f_{xx}(\lambda^\nu, \mu^\nu, x^\nu)\phi^\nu\phi^\nu - f_x(\lambda^\nu, \mu^\nu, x^\nu)v^\nu.
\end{aligned} \tag{3.7}
$$

Now let

$$
\begin{aligned}
\delta s^T &= (\delta\lambda, \delta x_2 - \delta x_1\phi_2^\nu, \cdots, \delta x_n - \delta x_1\phi_n^\nu),\\
\delta t^T &= (\delta\lambda, \delta\phi_2, \cdots, \delta\phi_n),\\
\delta r^T &= (\delta\lambda, \delta v_2, \cdots, \delta v_n),
\end{aligned}
$$

and

$$
\mathscr{A} = (\mathbb{D}_1 \,\vdots\, \tilde{\mathbb{A}}),
$$

i.e., $\mathbb{A}$ with first column replaced by $\mathbb{D}_1$. We rewrite (3.4), (3.5), (3.6) as

$$
\mathscr{A}\delta s = \mathbb{C}_1 + \delta x_1 \cdot \mathbb{C}_2 - \delta\mu \cdot \mathbb{D}_2, \tag{3.8}
$$

$$
\mathscr{A}\delta t = \mathbb{C}_2 - \mathbb{B}_1\delta x + \delta\lambda \cdot (\mathbb{D}_1 - \mathbb{D}_3) - \delta\mu \cdot \mathbb{D}_4, \tag{3.9}
$$

$$
\mathscr{A}\delta r = \mathbb{C}_3 - 2\mathbb{B}_1\delta\phi - \mathbb{B}_2\delta x + \delta\lambda \cdot (\mathbb{D}_1 - \mathbb{D}_5) - \delta\mu \cdot \mathbb{D}_6. \tag{3.10}
$$

Close to the fold point, $\mathscr{A}$ will be nonsingular by [7, Thm. 1] with $Px = x - x_1\phi^\nu$ and the condition $(I-P)\phi_0 \neq 0$ is satisfied by the ϕ_0 given in (2.4b). Thus (3.8) can be solved for δs in terms of δx_1 and $\delta\mu$. By solving $\mathscr{A}\alpha = \mathbb{C}_1$, $\mathscr{A}\beta = \mathbb{C}_2$, $\mathscr{A}\xi = \mathbb{D}_2$ we obtain

$$
\delta s = \alpha + \delta x_1 \cdot \beta - \delta\mu \cdot \xi,
$$

i.e.,

$$
\delta\lambda = \alpha_1 + \delta x_1 \cdot \beta_1 - \delta\mu \cdot \xi_1, \tag{3.11}
$$

$$
\delta x^T = (\delta x_1, \alpha_2 + \delta x_1(\beta_2 + \phi_2^\nu) - \delta\mu \cdot \xi_2, \cdots, \alpha_n + \delta x_1(\beta_n + \phi_n^\nu) - \delta\mu \cdot \xi_n). \tag{3.12}
$$

Substituting (3.11), (3.12) into (3.9) gives

$$
\mathscr{A}\delta t = \mathbb{C}_4 + \delta x_1 \cdot \mathbb{C}_5 + \delta\mu \cdot \mathbb{C}_6,
$$

where

$$
\mathbb{C}_4 = \mathbb{C}_2 - \mathbb{B}_1\begin{pmatrix} 0 \\ \alpha_2 \\ \vdots \\ \alpha_n \end{pmatrix} + \alpha_1(\mathbb{D}_1 - \mathbb{D}_3),
$$

$$
\mathbb{C}_5 = -\mathbb{B}_1\begin{pmatrix} 1 \\ \beta_2 + \phi_2^\nu \\ \vdots \\ \beta_n + \phi_n^\nu \end{pmatrix} + \beta_1(\mathbb{D}_1 - \mathbb{D}_3),
$$

$$\mathbb{C}_6 = \mathbb{B}_1 \begin{pmatrix} 0 \\ \xi_2 \\ \vdots \\ \xi_n \end{pmatrix} - \xi_1(\mathbb{D}_1 - \mathbb{D}_3) - \mathbb{D}_4.$$

Then (3.9) can be solved for δt in terms of δx_1 and $\delta\mu$. By solving $\mathscr{A}\gamma = \mathbb{C}_4$, $\mathscr{A}\eta = \mathbb{C}_5$, $\mathscr{A}\zeta = \mathbb{C}_6$ we get

$$\delta t = \gamma + \delta x_1 \cdot \eta + \delta\mu \cdot \zeta,$$

i.e.,

$$\delta\lambda = \gamma_1 + \delta x_1 \cdot \eta_1 + \delta\mu \cdot \zeta_1, \tag{3.13}$$

$$\delta\phi^T = (0, \gamma_2 + \delta x_1 \cdot \eta_2 + \delta\mu \cdot \zeta_2, \cdots, \gamma_n + \delta x_1 \cdot \eta_n + \delta\mu \cdot \zeta_n). \tag{3.14}$$

Substituting (3.11), (3.12), (3.14) into (3.10) gives

$$\mathscr{A}\delta r = \mathbb{C}_7 + \delta x_1 \mathbb{C}_8 + \delta\mu\mathbb{C}_9,$$

where

$$\mathbb{C}_7 + \mathbb{C}_3 - 2\mathbb{B}_1 \begin{pmatrix} 0 \\ \gamma_2 \\ \vdots \\ \gamma_n \end{pmatrix} - \mathbb{B}_2 \begin{pmatrix} 0 \\ \alpha_2 \\ \vdots \\ \alpha_n \end{pmatrix} + \alpha_1(\mathbb{D}_1 - \mathbb{D}_7),$$

$$\mathbb{C}_8 = -2\mathbb{B}_1 \begin{pmatrix} 0 \\ \eta_2 \\ \vdots \\ \eta_n \end{pmatrix} - \mathbb{B}_2 \begin{pmatrix} 1 \\ \beta_2 + \phi_2^{\nu} \\ \vdots \\ \beta_n + \phi_n^{\nu} \end{pmatrix} + \beta_1(\mathbb{D}_1 - \mathbb{D}_5),$$

$$\mathbb{C}_9 = -2\mathbb{B}_1 \begin{pmatrix} 0 \\ \zeta_2 \\ \vdots \\ \zeta_n \end{pmatrix} + \mathbb{B}_2 \begin{pmatrix} 0 \\ \xi_2 \\ \vdots \\ \xi_n \end{pmatrix} - \mathbb{D}_6 - \xi_1(\mathbb{D}_1 - \mathbb{D}_5).$$

Now (3.10) can be solved for δr in terms of δx_1 and $\delta\mu$. By solving $\mathscr{A}\varepsilon = \mathbb{C}_7$, $\mathscr{A}\sigma = \mathbb{C}_8$, $\mathscr{A}\zeta = \mathbb{C}_9$ we get:

$$\delta r = \varepsilon + \delta x_1 \cdot \sigma + \delta\mu \cdot \tau.$$

Thus

$$\delta\lambda = \varepsilon_1 + \delta x_1 \cdot \sigma_1 + \delta\mu \cdot \tau_1, \tag{3.15}$$

$$\delta v^T = (0, \varepsilon_2 + \delta x_2 \cdot \sigma_2 + \delta\mu \cdot \tau_2, \cdots, \varepsilon_n + \delta x_1 \cdot \sigma_n + \delta\mu \cdot \tau_n). \tag{3.16}$$

Finally we solve for $\delta\lambda$, $\delta\mu$ and δx_1 from (3.11), (3.13), (3.15) and we get δx, $\delta\phi$, δv by substituting $\delta\lambda$, $\delta\mu$ and δx_1 into (3.12), (3.14), (3.16). This concludes one step of Newton's method (3.1) applied to (2.14). Our indicated algorithm for solving the linear system defining the Newton iterates is similar to one proposed in [7].

4. Numerical example. We consider the boundary value problem

$$f(\lambda, \mu, x) \equiv x'' + \lambda \exp\left(\frac{x}{1+\mu x}\right) = 0, \tag{4.1a}$$

$$x(0) = x(1) = 0 \tag{4.1b}$$

TABLE 1
$n = 9$.

Iteration #	λ	μ	$x(\frac{1}{2})$	$\delta\lambda$	$\delta\mu$	$\delta x(\frac{1}{2})$
0	0.7 E+01	0.2 E+00	0.0 E+00			
1	0.38127545670 E+01	0.38551147946 E+00	0.16750981363 E+01	−0.31872454330 E+01	0.18551147946 E+00	0.16750981363 E+01
2	0.78576773020 E+01	0.17740338333 E+00	0.32434872161 E+01	0.40449227350 E+01	−0.20810809613 E+00	0.15683890798 E+01
3	0.39294985451 E+01	0.19744746492 E+00	0.37910406669 E+01	−0.39281787570 E+01	0.20044081587 E−01	0.54755345082 E+00
4	0.51463711104 E+01	0.24714689785 E+00	0.40574821373 E+01	0.12168725653 E+01	0.49699432929 E−01	0.26644147031 E+00
5	0.52923685180 E+01	0.24796233631 E+00	0.46889326799 E+01	0.14599740760 E+00	0.81543845745 E−03	0.61345054267 E+00
6	0.52309912609 E+01	0.24582359236 E+00	0.48890910376 E+01	−0.61377257101 E−01	−0.21387439436 E−02	0.20015835770 E+00
7	0.52293709363 E+01	0.24579987270 E+00	0.48968723523 E+01	−0.16203245848 E−02	−0.23719661879 E−04	0.77813146765 E−02
8	0.52293707956 E+01	0.24579981852 E+00	0.48968119436 E+01	−0.14068732957 E−06	−0.54179117830 E−07	0.39591269964 E−04
9	0.52293707955 E+01	0.24579981852 E+00	0.48969119462 E+01	−0.10647327667 E−09	−0.40428618326 E−11	0.26586630529 E−08

TABLE 2
$n = 19$.

Iteration #	λ	μ	$x(\frac{1}{2})$	$\delta\lambda$	$\delta\mu$	$\delta x(\frac{1}{2})$
0	0.7 E+01	0.2 E+00	0.0 E+00			
1	0.38133643303 E+01	0.38694965387 E+00	0.16752546596 E+01	−0.31866356697 E+01	0.18694965387 E+00	0.16752546596 E+01
2	0.78681797824 E+01	0.17459998937 E+00	0.32456385433 E+01	0.40548154521 E+01	−0.21234966450 E+00	0.15703838837 E+01
3	0.39125396675 E+01	0.19553130192 E+00	0.37837344202 E+01	−0.39556401149 E+01	0.20931312549 E−01	0.53809587685 E+00
4	0.51300512125 E+01	0.24621517892 E+00	0.40284572662 E+01	0.12175115451 E+01	0.50683877001 E−01	0.24472284601 E+00
5	0.52925071538 E+01	0.24810248016 E+00	0.46670924778 E+01	0.16245594123 E+00	0.18873012309 E−02	0.63863521164 E+00
6	0.52319102611 E+01	0.24582167561 E+00	0.48872921080 E+01	−0.60596892665 E−01	−0.22808045459 E−02	0.22019963021 E+00
7	0.52294854529 E+01	0.24578163304 E+00	0.48965402817 E+01	−0.24248082009 E−02	−0.40042569896 E−04	0.92481736701 E−02
8	0.52204859595 E+01	0.24578158539 E+00	0.48965672079 E+01	0.50660058720 E−06	−0.47653539058 E−07	0.26926185652 E−04
9	0.52294859594 E+01	0.24578158538 E+00	0.48965672065 E+01	−0.59191774348 E−10	−0.20462101792 E−11	−0.14292576328 E−08

TABLE 3
$n = 39$.

Iteration #	λ	μ	$x(\frac{1}{2})$	$\delta\lambda$	$\delta\mu$	$\delta x(\frac{1}{2})$
0	0.7 E+01	0.2 E+00	0.0 E+00			
1	0.38134020933 E+01	0.38717356794 E+00	0.16752643044 E+01	−0.31865979067 E+01	0.18717356794 E+00	0.16752643044 E+01
2	0.78690715249 E+01	0.17410257528 E+00	0.32456744296 E+01	0.40556694315 E+01	−0.21307099265 E+00	0.15704101251 E+01
3	0.39108904581 E+01	0.19519834690 E+00	0.37815078546 E+01	−0.39581810667 E+01	0.21095771611 E−01	0.53583342504 E+00
4	0.51271775286 E+01	0.24604229399 E+00	0.40229730337 E+00	0.12162870705 E+01	0.50843947092 E−01	0.24146517909 E+00
5	0.52924447855 E+01	0.24812918764 E+00	0.46629114939 E+01	0.16526725690 E+00	0.20868936530 E−02	0.63993846022 E+00
6	0.52320878199 E+01	0.24582419296 E+00	0.48869773788 E+01	−0.60356965561 E−01	−0.23049946814 E−02	0.22406588489 E+00
7	0.52294930642 E+01	0.24578054417 E+00	0.48965280488 E+01	−0.25947557338 E−02	−0.43648786706 E−04	0.95506699510 E−02
8	0.52294937243 E+01	0.24578049862 E+00	0.48965490197 E+01	0.66014850839 E−06	−0.45555808624 E−07	0.20970992874 E−04
9	0.52294937243 E+01	0.24578049861 E+00	0.48965490196 E+01	−0.40171603924 E−10	−0.13065842095 E−11	−0.19147290868 E−09

which describes an exothermic chemical reaction in an infinite slab [3]. It is discretized on the mesh $t_j = jh$, $j = 0, 1, 2, \cdots, n+1$ using the Collatz Mehrstellenverfahren:

$$x(t_{j-1}) - 2x(t_j) + x(t_{j+1}) - \frac{h^2}{12}[x''(t_{j-1}) + 10x''(t_j) + x''(t_{j+1})] = h^2 x''(t_j) + O(h^6).$$

The discretized form of (4.1) is thus:

$$Ax + E(\lambda, \mu, x) = 0, \; x_0 = x_{n+1} = 0, \qquad x \in \mathbb{R}^n, \tag{4.2}$$

where

$$E \equiv (E_1, \cdots, E_n)^T, \tag{4.3a}$$

$$E_i \equiv \frac{h^2}{12} \cdot \lambda \left[\exp\left(\frac{x_{i-1}}{1+\mu x_{i-1}}\right) + 10 \exp\left(\frac{x_i}{1+\mu x_i}\right) + \exp\left(\frac{x_{i+1}}{1+\mu x_{i+1}}\right) \right] \tag{4.3b}$$

$$A \equiv \begin{pmatrix} -2 & 1 & 0 & - & - & - & 0 \\ 1 & -2 & 1 & - & - & - & 0 \\ & 1 & -2 & 1 & & 0 & \\ & & \ddots & \ddots & \ddots & & \\ & & & \ddots & \ddots & \ddots & \\ & 0 & & & 1 & -2 & 1 \\ & & & & & 1 & -2 \end{pmatrix}. \tag{4.3c}$$

The double extended system now has the form

$$\begin{Bmatrix} l\phi - 1 \\ Ax + E(\lambda, \mu, x) \\ [A + E_x(\lambda, \mu, x)]\phi \\ E_{xx}(\lambda, \mu, x)\phi\phi + [A + E_x(\lambda, \mu, x)]v \\ lv \end{Bmatrix} = 0. \tag{4.4}$$

We choose l so that $l\phi = \phi_m$, $lv = v_m$, where $m = (n+1)/2$. (Of course we must choose n odd.) The calculation of each Newton's step requires solving nine $n \times n$ systems with the *same* coefficient matrix. The results of computation are given in tables 1, 2 and 3. They show good agreement with the results in [10].

REFERENCES

[1] N. W. BAZLEY AND G. C. WAKE, *The disappearance of criticality in the theory of thermal ignition*, Z. Angew. Math. Phys., 29 (1979), pp. 971-976.

[2] T. BODDINGTON, P. GRAY AND C. ROBINSON, *Thermal explosions and the disappearance of criticality at small activation energies: exact results for the slab*, Prov. Roy. Soc. London A., 368 (1979), pp. 441-468.

[3] D. W. FRADKIN AND G. C. WAKE, *The critical explosion parameter of thermal ignition*, J. Inst. Math. Appl., 20 (1977), pp. 471-484.

[4] J. P. KEENER AND H. B. KELLER, *Perturbed bifurcation theory*, Arch. Rational Mech. Anal., 50 (1973), pp. 159-175.

[5] H. B. KELLER, *Numerical solution of bifurcation and nonlinear eigenvalue problems*, in Application of Bifurcation Theory, P. H. Rabinowitz, ed., Academic Press, New York, 1977, pp. 359-384.

[6] H. B. KELLER AND R. K.-H. SZETO, *Calculation of flows between rotating disks*, Computing Methods in Applied Sciences and Engineering, Proc. 4th International Symposium, Versailles, 1979, North-Holland, Amsterdam, 1980, pp. 51-61.

[7] G. MOORE AND A. SPENCE, *The calculation of turning points of nonlinear equations*, SIAM J. Numer. Anal., 17 (1980), pp. 567-576.

[8] W. H. RAY, *Bifurcation and stability problems in astrophysics*, in Application of Bifurcation Theory, P. H. Rabinowitz, ed., Academic Press, New York, 1977, pp. 285–315.

[9] R. SEYDEL, *Numerical computation of branches in nonlinear equations*, Numer. Math., 33 (1979), pp. 339–352.

[10] A. SPENCE AND B. WERNER, *Non-simple turning points and cusps*, IMA J. Numer. Anal., 2 (1982) pp. 413–427.

[11] A. JEPSON AND A. SPENCE, *Folds in solutions of two parameter systems and their calculation: Part* I., SIAM J. Numer. Anal., 22 (1985), pp. 347–368.

SIAM J. SCI. STAT. COMPUT.
Vol. 7, No. 2, April 1986

002

A NOTE ON THE ALGORITHM OF ALEFELD AND PLATZÖDER FOR SYSTEMS OF NONLINEAR EQUATIONS*

J. M. SHEARER† AND M. A. WOLFE†

Abstract. A modification (MAP) of the algorithm (AP) of Alefeld and Platzöder (SIAM J. Numer. Anal., 20 (1983), pp. 210–219) for the solution of systems of nonlinear algebraic equations which is similar to the modification (MKM) of the Krawczyk–Moore algorithm (KM) which has been suggested by Wolfe (SIAM J. Numer. Anal., 17 (1980), pp. 376–379) is described. Theoretical results for MAP which are similar to those which have been derived by Alefeld and Platzöder for AP are presented. Numerical results comparing the algorithms KM, MKM, AP and MAP are given.

Key words. interval mathematics, nonlinear algebraic equations, algorithm, efficiency index, inner iterations, strategy

AMS(MOS) subject classifications. 65J15, 47H17, 47H07

1. Introduction. The Krawczyk–Moore algorithm [13], [14], [15], [16] for bounding an isolated zero of a given mapping $f: R^n \to R^n$ using interval arithmetic is known to be very effective, especially when combined with a search procedure of the kind which has been advocated by Moore and Jones [16], [11], [12]. Subsequently several authors, among whom are Hansen [7], Wolfe [22], Hansen and Sengupta [9], Qi [19], [20] and Alefeld and Platzöder [3], have suggested modifications of the Krawczyk–Moore algorithm which improve computational efficiency. In this paper, KM and AP denote the Krawczyk–Moore and Alefeld–Platzöder algorithms respectively.

As pointed out by Alefeld and Platzöder, AP requires less computational labor per iteration than does KM, and is capable of more sophisticated implementation than they themselves describe. It is the purpose of this paper to show that a simple modification of AP gives rise to an algorithm MAP which is more efficient computationally than KM, the modification KKM of KM which has been suggested by Wolfe [22], and the algorithm AP in the basic form which has been described by Alefeld and Platzöder [3].

2. Notation. The symbols R^n, $M(R^n)$, $I(R^n)$, and $I(M(R^n))$ denote the sets of $n \times 1$ real vectors, $n \times n$ real matrices, $n \times 1$ interval vectors, and $n \times n$ interval matrices respectively. Elements of R^n, $M(R^n)$, $I(R^n)$, and $I(M(R^n))$ are represented by the lower case italic, upper case italic, lower case bold, and upper case bold letters respectively. The notation which is used in this paper and that which is used in [22] are identical in all other respects.

Let $A \in M(R^n)$ and $\mathbf{b} \in I(R^n)$ be given. The result of applying the Gauss algorithm [1] to the pair $(A, \mathbf{b})$ is represented by $\mathbf{g}(A, \mathbf{b}) \in I(R^n)$. As explained by Alefeld and Platzöder [3], if A^{-1} exists then $\exists M_A \in M(R^n)$ depending only on A, such that

$$w(\mathbf{g}(A, \mathbf{b})) = M_A w(\mathbf{b}), \tag{1}$$

and

$$A^{-1}\mathbf{b} \subseteq \mathbf{g}(A, \mathbf{b}). \tag{2}$$

3. Preliminaries. This section contains some results which are required in the proof of the theorem which is presented in § 4. Let $\mathbf{K}_N: I(R^n) \times I(M(R^n)) \times M(R^n) \to$

*Received by the editors July 10, 1984, and in revised form January 14, 1985.

† Department of Applied Mathematics, University of St. Andrews, St. Andrews, Fife, KY16 9SS, Scotland.

$I(R^n)$ be defined by

$$\mathbf{K}_N(\mathbf{x}, \mathbf{F}, A) = m(\mathbf{x}) - \mathbf{g}(A, \{f(m(\mathbf{x})) - (A - \mathbf{F})(\mathbf{x} - m(\mathbf{x}))\}) \tag{3}$$

where A is nonsingular, $\mathbf{g}: M(R^n) \times I(R^n) \to I(R^n)$ is given by (1), and $f: R^n \to R^n$ is a given mapping.

LEMMA 1. (i) *If* $\mathbf{x}, \mathbf{y} \in I(R^n)$ *and* $\beta > 0$ *are such that* $w(\mathbf{y}) \leqq \beta w(\mathbf{x})$, $\mathbf{F} \in I(M(R^n))$, $A \in M(R^n)$, *and* A^{-1} *exists, then*

$$w(\mathbf{K}_N(\mathbf{y}, \mathbf{F}, A)) \leqq \beta w(\mathbf{K}_N(\mathbf{x}, \mathbf{F}, A)).$$

(ii) *If* $\mathbf{F}, \mathbf{G} \in I(M(R^n))$ *are such that* $\mathbf{G} \subseteq \mathbf{F}$, $\mathbf{x} \in I(R^n)$, $A \in M(R^n)$, *and* A^{-1} *exists then*

$$w(\mathbf{K}_N(\mathbf{x}, \mathbf{G}, A)) \leqq w(\mathbf{K}_N(\mathbf{x}, \mathbf{F}, A)).$$

LEMMA 2. *Suppose that* $f: D \subseteq R^n \to R^n$ *is a given mapping with* $f \in C^1(D)$. *Let* $\mathbf{f}': I(D) \to I(M(R^n))$ *be a continuous inclusion monotonic interval extension of* $f': D \to M(R^n)$. *Suppose that* $\exists \lambda > 0$ *such that* $(\forall \mathbf{x} \in I(D))$

$$\|w(\mathbf{f}'(\mathbf{x}))\| \leqq \lambda \|w(\mathbf{x})\|. \tag{4}$$

Then $\exists \mu > 0$ *such that* $(\forall \mathbf{x} \in I(D))$

$$\|w(\mathbf{K}_N(\mathbf{x}, \mathbf{f}'(\mathbf{x}), m(\mathbf{f}'(\mathbf{x}))))\| \leqq \mu \|w(\mathbf{x})\|^2.$$

The validity of lemmas 1 and 2 follows easily from (1), (2), and from the results which have been established in [3].

4. The Algorithm MAP. Let $f: D \subseteq R^n \to R^n$ be a given mapping with $f \in C^1(\hat{D})$ where $\hat{D} \subseteq D$ is an open convex set. Let $f: I(\hat{D}) \to I(R^n)$ and $\mathbf{f}': I(\hat{D}) \to I(M(R^n))$ be continuous inclusion monotonic interval extensions of $f: \hat{D} \to R^n$ and $f': \hat{D} \to M(R^n)$ respectively. Let $\hat{\mathbf{x}} \in I(\hat{D})$, a sequence of nonnegative integers $p^{(k)}$, and $\alpha \in [0, 1)$ be given. Then the algorithm MAP for bounding an isolated zero of f in $\hat{x}$ is as follows.

1. $\mathbf{x}^{(0)} := \hat{\mathbf{x}}$
2. $\mathbf{F}^{(0)} := \mathbf{f}'(\mathbf{x}^{(0)})$
3. $B^{(0)} := m(\mathbf{F}^{(0)})$
4. $\mathbf{x}^{(0,0)} := \mathbf{x}^{(0)}$
5. **for** $m = 0$ **to** $p^{(0)}$ **do**
   5.1. $\mathbf{x}^{(0,m+1)} := \mathbf{K}_N(\mathbf{x}^{(0,m)}, \mathbf{F}^{(0)}, B^{(0)}) \cap \mathbf{x}^{(0,m)}$
6. $\mathbf{x}^{(1)} := \mathbf{x}^{(0,p^{(0)}+1)}$
7. $k := 1$
8. **while true do**
   8.1. $\mathbf{F}^{(k)} := \mathbf{f}'(\mathbf{x}^{(k)})$
   8.2. $A^{(k)} := m(\mathbf{F}^{(k)})$
   8.3. $\mathbf{u}^{(k)} := \mathbf{K}_N(\mathbf{x}^{(k)}, \mathbf{F}^{(k)}, A^{(k)})$
   8.4. $\mathbf{z}^{(k)} := \mathbf{u}^{(k)} \cap \mathbf{x}^{(k)}$
   8.5. **if** $w(\mathbf{u}^{(k)}) \leqq \alpha w(\mathbf{x}^{(k)})$
   then
      8.5.1. $B^{(k)} := A^{(k)}$
      8.5.2. $\mathbf{x}^{(k,1)} := \mathbf{z}^{(k)}$
      8.5.3. **for** $m = 1$ **to** $p^{(k)}$ **do**
         8.5.3.1. $\mathbf{u}^{(k,m)} := \mathbf{K}_N(\mathbf{x}^{(k,m)}, \mathbf{F}^{(k)}, A^{(k)})$
         8.5.3.2. $\mathbf{x}^{(k,m+1)} := \mathbf{u}^{(k,m)} \cap \mathbf{x}^{(k,m)}$
      8.5.4. $\mathbf{x}^{(k+1)} := \mathbf{x}^{(k,p^{(k)}+1)}$

else

8.5.5. $B^{(k)} := B^{(k-1)}$

8.5.6. $\mathbf{v}^{(k)} := \mathbf{K}_N(\mathbf{x}^{(k)}, \mathbf{F}^{(k)}, B^{(k)})$

8.5.7. $\mathbf{x}^{(k,1)} := \mathbf{v}^{(k)} \cap \mathbf{z}^{(k)}$

8.5.8. **for** $m = 1$ **to** $p^{(k)}$ **do**

8.5.8.1. $\mathbf{v}^{(k,m)} := \mathbf{K}_N(\mathbf{x}^{(k,m)}, \mathbf{F}^{(k)}, A^{(k)})$

8.5.8.2. $\mathbf{x}^{(k,m+1)} := \mathbf{v}^{(k,m)} \cap \mathbf{x}^{(k,m)}$

8.5.9. $\mathbf{x}^{(k+1)} := \mathbf{x}^{(k,p^{(k)}+1)}$

8.6. $k := k+1$

If $p^{(k)} = 0$ $(\forall k \geqq 0)$ then MAP and AP are identical. Otherwise $\mathbf{f}'(\mathbf{x}^{(k)})$ is re-used $p^{(k)}$ times in iteration k.

THEOREM. *Let $f: D \subseteq R^n \to R^n$ be a given mapping, with $f \in C^1(\hat{D})$ where $\hat{D} \subseteq D$ is an open convex set. Let $\mathbf{f}: I(\hat{D}) \to I(R^n)$ and $\mathbf{f}': I(\hat{D}) \to I(M(R^n))$ be inclusion monotonic interval extensions of $f: \hat{D} \to R^n$ and $f': \hat{D} \to M(R^n)$ respectively. Suppose that $\mathbf{x}^{(0)} \in I(\hat{D})$ is such that $w(\mathbf{x}^{(0)}) > 0$, that $B^{(0)} = m(\mathbf{f}'(\mathbf{x}^{(0)}))$ is nonsingular, and that for some $\alpha \in [0, 1)$*

$$w(\mathbf{K}_N(\mathbf{x}^{(0)}, \mathbf{f}'(\mathbf{x}^{(0)}), B^{(0)})) \leqq \alpha w(\mathbf{x}^{(0)}). \tag{5}$$

If $\mathbf{x}^{(0)}$ contains a zero x^* of f, then the sequence $(\mathbf{x}^{(k)})$ which is generated from MAP is well defined and $\mathbf{x}^{(k)} \to x^*$ $(k \to \infty)$. If also (4) holds, then for each $p^{(k)} \geqq 0$ $\exists \mu^{(k)} > 0$ such that

$$\| w(\mathbf{x}^{(k+1)}) \| \leqq \mu^{(k)} \| w(\mathbf{x}^{(k)}) \|^{2+p^{(k)}} \qquad (k \geqq 0). \tag{6}$$

If MAP terminates because an empty intersection occurs in one of the steps 5.1, 8.4, 8.5.3.2, 8.5.7, *or* 8.5.8.2, *then there is no zero of f in $x^{(0)}$. Finally if, instead of* (5),

$$\mathbf{K}_N(\mathbf{x}^{(0)}, \mathbf{f}'(\mathbf{x}^{(0)}), B^{(0)}) \subset \operatorname{int}(\mathbf{x}^{(0)}) \tag{7}$$

then (5) *holds for some $\alpha \in [0, 1)$ and $\exists x^* \in \mathbf{x}^{(0)}$ such that $f(x^*) = 0$.*

Proof. The proof follows closely that of the theorem which has been proved for AP by Alefeld and Platzöder [3], and therefore only an outline will be given.

By Lemma 2 [3], $A^{(k)-1}$ exists $(\forall k \geqq 1)$ so the sequence $(\mathbf{x}^{(k)})$ generated from MAP is well defined provided no empty intersections occur. A simple inductive argument shows that $(x^* \in \mathbf{x}^{(0)} \wedge f(x^*) = 0) \Rightarrow (x^* \in \mathbf{x}^{(k)} (\forall k \geqq 0))$. Another inductive argument using Lemma 1 (i), (ii) shows that $(\forall k \geqq 0)$

$$w(\mathbf{x}^{(k+1)}) \leqq \alpha^{k+1} w(\mathbf{x}^{(0)}).$$

So $w(\mathbf{x}^{(k)}) \to 0$ $(k \to \infty)$, whence, because $x^* \in \mathbf{x}^{(k)} (\forall k \geqq 0)$, $\mathbf{x}^{(k)} \to x^*$ $(k \to \infty)$. The result (6) is a consequence of Lemma 2. The remainder of the proof follows closely that of the theorem which is given in [3]. □

The theoretical results given by Wolfe [22] for MKM are valid with $p^{(k)}$ replacing p.

5. Methods for determining the $\mathbf{p^{(k)}}$. Computational experience with both MKM and MAP has shown that the choice of the number of inner iterations, $p^{(k)}$, greatly affects the efficiency of both MKM and MAP. Wolfe [22] has considered choosing $p^{(k)} = p$, $(\forall k \geqq 0)$, for some fixed integer p in the implementation of MKM, and has suggested that the optimal value for p might be estimated using techniques similar to those described by Brent [5]. This is, however, difficult to implement in practice. Hansen and Greenberg [8] suggest re-using the Jacobian matrix while the width of successive iterates is being sufficiently reduced. Unfortunately it is difficult to determine

what constitutes a sufficient width reduction since this varies greatly from problem to problem.

In this section two methods which automatically select $p^{(k)}$, independently of the size and complexity of the system of equations, are presented. Assume that the iterate $x^{(k,m)}$ $(k \geqq 0, m \geqq 1)$ has been computed, and that a reliable estimate of the relative efficiencies of computing a new outer iterate, $\mathbf{x}^{(k+1,1)}$ and that of computing a new inner iterate, $\mathbf{x}^{(k,m+1)}$ can be obtained. Then, whichever iterate is expected to be more efficient could be computed and the decision process could be repeated. Efficiency indices ρ_I and ρ_0 corresponding to the computation of $\mathbf{x}^{(k,m+1)}$ and $\mathbf{x}^{(k+1,1)}$ respectively are given by

$$\rho_I = -\ln (\| w(\mathbf{x}^{(k,m+1)}) \| / \| w(\mathbf{x}^{(k,m)}) \|) / T_I$$

and

$$\rho_0 = -\ln (\| w(\mathbf{x}^{(k+1,1)}) \| / \| w(\mathbf{x}^{(k,m)}) \|) / T_0,$$

where T_I and T_0 are the CPU times required to compute $\mathbf{x}^{(k,m+1)}$ and $\mathbf{x}^{(k+1,1)}$ respectively from $\mathbf{x}^{(k,m)}$. Experience with both MKM and MAP has shown that the CPU time required for each outer iteration does not vary greatly with k. Therefore the CPU time for the next outer iteration could be estimated by the CPU time required for the previous one. Similarly the CPU time for the next inner iteration could be estimated by the CPU time for the previous inner iteration. The theory corresponding to both MKM and MAP suggests that if $\mathbf{x}^{(k+1,0)} = \mathbf{x}^{(k,m)}$ then

$$\| w(\mathbf{x}^{(k+1,1)}) \| \approx M^{(k)} \| w(\mathbf{x}^{(k,m)}) \|^2$$

where

$$M^{(k)} = \| w(\mathbf{x}^{(k,1)}) \| / \| w(\mathbf{x}^{(k,0)}) \|^2,$$

and that

$$\| w(\mathbf{x}^{(k,m+1)}) \| \approx N^{(k,m)} \| w(\mathbf{x}^{(k,m)}) \|$$

where

$$N^{(k,m)} = \| w(\mathbf{x}^{(k,m)}) \| / \| w(\mathbf{x}^{(k,m-1)}) \|.$$

The two strategies for deciding whether or not to compute another inner iteration use some or all of the above approximations, and are as follows.

Strategy 1. 1. Compute the efficiency index ρ_I for an inner iteration from

$$\rho_I = -\ln (\| w(\mathbf{x}^{(k,m)}) \| / \| w(\mathbf{x}^{(k,m-1)}) \|) / T^{(k,m)}$$

where $T^{(k,m)}$ is the time required to compute $\mathbf{x}^{(k,m)}$ from $\mathbf{x}^{(k,m-1)}$.

2. Estimate the efficiency index ρ_0 which would have been obtained if instead $\mathbf{x}^{(k+1,1)}$ had been computed with $\mathbf{x}^{(k+1,0)} = \mathbf{x}^{(k,m-1)}$ from

$$\rho_0 = -\ln ((M^{(k)} \| w(\mathbf{x}^{(k,m-1)}) \|^2) / \| w(\mathbf{x}^{(k,m-1)}) \|) / T^{(k,1)}.$$

3. If $\rho_0 > \rho_I$ then recompute the Jacobian. Otherwise re-use the Jacobian.

Strategy 2. 1. Estimate the efficiency index ρ_I for the inner iterate $\mathbf{x}^{(k,m+1)}$ from

$$\rho_I = -\ln ((N^{(k,m)} \| w(\mathbf{x}^{(k,m)}) \|) / \| w(\mathbf{x}^{(k,m)}) \|) / T^{(k,m)}.$$

2. Estimate the efficiency index ρ_0 for computing the outer iterate $\mathbf{x}^{(k+1,1)}$ with $\mathbf{x}^{(k+1,0)} = \mathbf{x}^{(k,m)}$ from

$$\rho_0 = -\ln ((M^{(k)} \| w(\mathbf{x}^{(k,m)}) \|^2) / \| w(\mathbf{x}^{(k,m)}) \|) / T^{(k,1)}.$$

3. If $\rho_0 > \rho_I$ then recompute the Jacobian. Otherwise re-use the Jacobian.

Although strategy 1, by the nature of its design, should always give one inner iteration too many, it uses fewer approximations than strategy 2. In practice there is little difference between the results obtained from the two strategies when they are applied to MAP.

6. Numerical results. The algorithms MKM and MAP have been implemented in Triplex *S*-algol [4], [6], [17] using strategies 1 and 2 to determine the $p^{(k)}$ and have been used to solve several systems of equations on a VAX-11/780 computer. Numerical results for five examples are given in this section. Examples 1 and 2 are of the form

$$Ax + d(x) + c = 0 \tag{8}$$

where $A \in M(R^n)$ is tridiagonal, $d: R^n \to R^n$ is a diagonal operator and $c \in R^n$. Example 1 is the same as Example 2 in [1] and Example 2 is the same as Example 2 in [3].

Examples 3 and 4 are of the form

$$Ax + Td(x) + c = 0$$

where $A, T \in M(R^n)$ are tridiagonal, $d: R^n \to R^n$ is a diagonal operator and $c \in R^n$. In Example 3 the system of equations is obtained by using a discretization, described by Henrici [10], to solve the boundary value problem

$$u''(t) = 2(u(t) - t/2 + 1)^2, \qquad u(0) = u(1) = 0.$$

Example 4 is obtained by using the same discretization to solve the boundary value problem

$$u''(t) = \tfrac{1}{2}(u(t))^3, \qquad u(0) = 1, \quad u(1) = 2.$$

Example 5 is described by Rall [21, pp. 66, 183] and the corresponding system of nonlinear algebraic equations, which is obtained using a discretization described by Ortega and Rheinboldt, [18, pp. 14-16], has the form (8) where $A \in M(R^n)$, $d: R^n \to R^n$ is a diagonal operator and $c \in R^n$. The numerical results which are presented in this section are obtained using the initial interval $\mathbf{x}^{(0)} = (\mathbf{x}_i^{(0)})_{n\times 1}$ where for $i = 1, \ldots, n$ $\mathbf{x}_i^{(0)}$ has the value [0, 5], [50, 100], [0, 4], [0, 10], and [1, 7] for Examples 1-5 respectively.

For each example, (5) is satisfied with

$$\alpha = \max\{(w(\mathbf{K}_N(\mathbf{x}^{(0)}, \mathbf{f}'(\mathbf{x}^{(0)}), B^{(0)})))_i / (w(\mathbf{x}^{(0)}))_i \mid i = 1, \cdots, n\}$$

and (7) is also satisfied. The convergence criterion for each example is $\| w(\mathbf{x}^{(k)}) \| \leqq 10^{-10}$. Tables 1-5 give the CPU times, in seconds, which are required by the algorithm MKM

TABLE 1

n	Strategy 1	Strategy 2	$p=0$	$p=1$	$p=2$	$p=3$	$p=4$
5	46.55	52.64	52.68	47.48	48.53	50.29	52.27
10	153.74	153.84	174.98	141.87	136.75	144.44	145.35
20	605.96	520.16	772.83	598.44	540.51	445.11	434.73

TABLE 2

n	Strategy 1	Strategy 2	$p=0$	$p=1$	$p=2$	$p=3$	$p=4$
5	30.06	30.66	40.30	30.02	32.63	29.80	29.58
10	76.17	84.00	101.93	73.82	81.30	72.45	72.83
20	380.82	289.93	478.24	348.59	370.79	285.99	287.18

to converge in Examples 1-5 respectively. Tables 6-10 give the CPU times, in seconds, for the algorithm MAP to converge in Examples 1-5 respectively.

TABLE 3

n	Strategy 1	Strategy 2	$p=0$	$p=1$	$p=2$	$p=3$	$p=4$
5	21.93	23.77	24.49	22.50	23.44	25.19	23.93
10	82.21	75.61	83.96	73.38	71.94	78.77	74.77
20	495.43	450.03	922.34	688.09	501.41	535.31	466.16

TABLE 4

n	Strategy 1	Strategy 2	$p=0$	$p=1$	$p=2$	$p=3$	$p=4$
5	22.22	22.48	25.36	22.15	23.41	23.98	25.39
10	77.09	73.18	91.03	74.77	78.79	76.93	78.49
20	510.70	542.44	960.83	626.15	566.86	506.66	488.94

TABLE 5

n	Strategy 1	Strategy 2	$p=0$	$p=1$	$p=2$	$p=3$	$p=4$
9	37.92	44.20	44.79	36.33	43.96	38.56	41.92
16	202.59	224.07	296.36	197.98	211.50	181.76	183.00
25	532.40	498.73	1048.32	669.48	702.19	517.96	407.83

TABLE 6

n	Strategy 1	Strategy 2	$p=0$	$p=1$	$p=2$	$p=3$	$p=4$
5	48.92	49.35	54.13	48.87	50.72	50.42	54.21
10	123.58	125.41	135.28	118.00	120.71	129.18	130.25
20	391.47	383.09	439.00	387.50	378.80	385.29	390.74

TABLE 7

n	Strategy 1	Strategy 2	$p=0$	$p=1$	$p=2$	$p=3$	$p=4$
5	30.82	34.20	39.21	30.60	33.72	29.58	30.09
10	75.01	84.49	98.19	74.63	84.75	74.00	72.44
20	240.38	252.06	311.07	228.94	268.73	215.00	211.39

TABLE 8

n	Strategy 1	Strategy 2	$p=0$	$p=1$	$p=2$	$p=3$	$p=4$
5	21.72	22.09	23.89	21.43	21.65	24.88	22.78
10	78.40	74.19	84.09	74.29	72.05	79.70	75.67
20	326.00	336.55	408.51	343.42	305.88	342.54	309.03

TABLE 9

n	Strategy 1	Strategy 2	$p=0$	$p=1$	$p=2$	$p=3$	$p=4$
5	21.78	22.10	25.05	22.16	22.79	22.95	24.84
10	76.17	74.63	90.68	76.61	78.03	78.56	79.73
20	333.72	344.45	459.76	350.47	358.12	359.13	340.37

TABLE 10

n	Strategy 1	Strategy 2	$p=0$	$p=1$	$p=2$	$p=3$	$p=4$
9	35.17	37.98	41.92	35.86	41.42	37.69	41.10
16	122.91	126.68	160.11	127.34	138.57	122.38	133.19
25	320.14	335.36	428.28	317.32	350.63	321.89	331.65

7. Conclusions. The results given in Tables 1-10 illustrate quite clearly that the algorithm MAP is more efficient than the algorithm MKM and the increase in efficiency becomes greater as n increases. The results also show that although the two strategies to select the number of inner iterations do not always choose the optimal value for $p^{(k)}$, they do always produce a very significant increase in efficiency over the algorithm AP, and this improvement increases with n. Thus of the four algorithms KM, AP, MKM, and MAP, MAP, using either strategy to determine the number of inner iterations $p^{(k)}$, is the most efficient.

REFERENCES

[1] G. ALEFELD, *Über die Existenz einer eindeutigen Lösung bei einer Klasse nichtlinearer Gleichungssysteme und deren Berechnung mit Iterationsverfahren*, Aplikace Matematiky, 17 (1972), pp. 329-340.

[2] G. ALEFELD AND J. HERZBERGER, *Einführung in die Intervallrechnung*, Reihe Informatik 12, Bibliographisches Institut, Mannheim, 1974.

[3] G. ALEFELD AND L. PLATZÖDER, *A quadratically convergent Krawczyk-like algorithm*, SIAM J. Numer. Anal., 20 (1983), pp. 210-219.

[4] P. J. BAILEY, A. J. COLE AND R. MORRISON, *Triplex User Manual* CS/82/5, Department of Computational Science Univ. St. Andrews, North Haugh St. Andrews, Fife, KY16 9SX, Scotland.

[5] R. P. BRENT, *Some efficient algorithms for solving systems of nonlinear equations*, SIAM J. Numer. Anal., 10 (1973), pp. 327-344.

[6] A. J. COLE AND R. MORRISON, *An Introduction to Programming in S-algol*, Cambridge Univ. Press, Cambridge, 1982.

[7] E. HANSEN, *Interval forms of Newton's method*, Computing, 20 (1978), pp. 153-163.

[8] E. HANSEN AND R. I. GREENBERG, *An interval Newton method*, Appl. Math. Comput., 12 (1983), pp. 89-98.

[9] E. HANSEN AND S. SENGUPTA, *Bounding solutions of systems of equations using Interval analysis*, BIT, 21 (1981), pp. 203-211.

[10] P. HENRICI, *Discrete Variable Methods in Ordinary Differential Equations*, John Wiley, New York, London, 1962.

[11] S. T. JONES, *Searching for solutions of finite nonlinear systems-An interval approach*, PhD thesis, Univ. Wisconsin-Madison, 1978.

[12] ———, *Locating safe starting regions for iterative methods: A heuristic algorithm*, in Interval Mathematics 1980, K.L.E. Nickel ed., Academic Press, New York, 1980.

[13] R. KRAWCZYK, *Newton-Algorithmen zur Bestimmung von Nullstellen mit Fehlerschranken*, Computing, 4 (1969), pp. 187-201.

[14] R. E. MOORE, *A test for existence of solutions to nonlinear systems*, SIAM J. Numer. Anal., 14 (1977), pp. 611-615.

[15] ———, *A computational test for convergence of iterative methods for nonlinear systems*, SIAM J. Numer. Anal., 15 (1978), pp. 1194-1196.

[16] R. E. MOORE AND S. T. JONES, *Safe starting regions for iterative methods*, SIAM J. Numer. Anal., 14 (1977), pp. 1051-1065.

[17] R. MORRISON, A. J. COLE, P. J. BAILEY, M. A. WOLFE AND J. M. SHEARER, *Experience in using a high level language which supports interval arithmetic*, read at ARITH6, the sixth symposium on computer arithmetic, Aarhus, Denmark, 1983.

[18] J. M. ORTEGA AND W. C. RHEINBOLDT, *Iterative Solution of Nonlinear Equations in Several Variables*, Academic Press, New York, 1970.

[19] L. QI, *A generalization of the Krawczyk-Moore algorithm*, in Interval Mathematics 1980, K. L. E. Nickel, ed., Academic Press, New York, 1980.

[20] L. QI, *Interval boxes of solutions of nonlinear systems*, Computing, 27 (1981), pp. 137-144.

[21] L. B. RALL, *Computational Solution of Nonlinear Operator Equations*, John Wiley, New York, 1969.

[22] M. A. WOLFE, *A modification of Krawczyk's algorithm*, SIAM J. Numer. Anal., 17 (1980), pp. 376-379.

SIAM J. SCI. STAT. COMPUT.
Vol. 7, No. 2, April 1986

CONVERSION OF DECIMAL NUMBERS TO IRREDUCIBLE RATIONAL FRACTIONS*

R. PYZALSKI† AND M. VALA‡

Abstract. A simple method for the conversion of decimal numbers to irreducible rational fractions is formulated using a continued fraction expansion. A condition has been found such that the conversion may be performed in a single-valued way for any set of decimal numbers. The algorithm has been applied to a modified computer program and used to recover lost precision in certain computer calculations.

Key words. decimal number, conversion to fractions

1. Introduction. In many computations it is convenient to give the final results as irreducible fractions instead of decimal numbers. Matrix elements in quantum mechanics, in spectroscopic calculations, and in group theory are examples. Moreover, an algorithm for the conversion can be used for the recovery of precision in numbers being used as intermediate results in some computations. Many algorithms for such a conversion have been used without publication. In the Statistical Analysis System (SAS) there exists a machine language subroutine for the conversion of output numbers called the FRACT format [7]. A common feature of these algorithms is the lack of control of correctness of the conversion in relation to the precision of the numbers to be converted. Without this, the conversion is not single-valued, i.e., the same number can be converted to several fractions according to an arbitrarily selected difference between the number and the fraction.

The conversion of decimal numbers to irreducible fractions is a simple example of the more general mathematical problem concerning the approximation of algebraic numbers by rational numbers. Although an old problem (cf. [1], [2], [6]) for which good approximations are known [1], the application of the results to a computer conversion of a set of numbers is not straightforward.

In this paper a method for the conversion of decimal numbers belonging to the interval (0, 1) to irreducible rational fractions is described assuming that the maximum expected denominator of the fractions to be obtained is known. This assumption allows us to find a simple equation for the precision of the numbers to be converted. The conversion is performed with the same precision. The conversion has an important feature. If the number to be converted is computed from a rational function in an approximate way, then one can calculate the necessary precision of the number to get the same fraction as from exact computation. A brief outline of the method was published earlier [5]. Its generalization for any decimal number greater than one is straightforward.

The method is based on a continued fraction expansion of a rational number. The expansion and some of its properties are given in § 2. The conversion of an approximate number (i.e., with a value different than the expected fraction) and the propagation of errors of the decimal number through the different levels of a continued fraction is described in § 3. In § 4 a condition is formulated for the conversion to be

* Received by the editors October 4, 1983, and in revised form June 13, 1984. This work was supported by the National Science Foundation under grant CHE-8206142.

† Department of Chemistry, University of Florida, Gainesville, Florida 32611. Present address, Theoretical Chemistry Institute, University of Wisconsin, Madison, Wisconsin 53706.

‡ Department of Chemistry, University of Florida, Gainesville, Florida 32611.

single-valued for a set of decimal numbers. A summary and the conclusions are given in § 5. The Appendix contains a Fortran subroutine for the conversion.

2. Continued fraction expansion of a rational number. Consider a rational number u_1 which is greater than zero and smaller than one,

$$0 < u_1 < 1. \tag{1}$$

Expressing u_1 as a finite continued fraction

$$\begin{aligned} u_1 &= 1/(n_1 + u_2) = 1/(n_1 + 1/(n_2 + u_3)) \\ &= 1/(n_1 + 1/(n_2 + \cdots + 1/n_k)) \end{aligned} \tag{2}$$

where the ith partial denominator is the nearest small integer of $1/u_i$

$$n_i = [1/u_i] \tag{3}$$

and

$$u_i = 1/u_{i-1} - n_{i-1}. \tag{4}$$

Following Knuth [4] a set of the partial denominators n_i for $i = 1, 2, \cdots, k$ in the expansion (2) will be called the Euclidean representation (ER) of the number u_1. The number of elements of the set will be called a dimension of the ER. The continued fraction (2) can be converted to a rational fraction l/m by the recurrence equations [9]

$$l_i = l_{i-1} n_i + l_{i-2}, \tag{5}$$

$$m_i = m_{i-1} n_i + m_{i-2}, \tag{6}$$

where

$$l = l_k, \quad m = m_k, \quad l_0 = 0, \quad m_0 = 1. \tag{7}$$

The sequence of l_i/m_i fractions gives all the best approximations of u_1 by fractions whose denominators do not exceed the value m_i [1].

If the following inequality

$$0 < u_i < 1 \tag{8}$$

is satisfied for $i = 2, 3, \cdots, k$, then the expansion (2) is a single-valued one [3], [9], i.e., the rational number u_1 has one and only one ER.

If instead of (3) for $i = k$, we allow

$$n'_k = [1/u_k] - 1, \tag{9}$$

then we get another ER with $k+1$ dimensions. In this case

$$u_1 = 1/(n_1 + 1/(n_2 + \cdots 1/(n'_k + 1/1))). \tag{10}$$

So now $n_{k+1} = 1$ and $u_{k+1} = 1$. One can easily check that the recurrence equations (5) and (6) lead to the same fraction l/m for both ERs. Thus the two ERs are equivalent. The method of introducing the second ER looks artificial. However, both are necessary for a description of the conversion of the approximate numbers, i.e., greater or smaller in comparison to $u_1 = l/m$.

3. Conversion of an approximate number. If the expansion (2) together with the recurrence equations (5) and (6) are applied to a decimal number, they give an exact conversion where l/m is equal to the converted number. Assume that instead of the

u_1, we want to convert an approximate decimal number $u = u_1 + \Delta_1$ with

$$0 < u < 1 \tag{11}$$

(similar to (1)). When can this number be converted to the l/m fraction, i.e., converted the same as the exact u_1 number? From our previous consideration we conclude that it may be done only when the number u can be represented by an ER whose k or $k+1$ initial n_i numbers have the same values as one of the two ER for the exact u_1 number. Thus what condition must be imposed upon Δ_1 to get this ER? The condition will arise from properties of the expansion of the approximate number.

The reciprocal of u can always be written as

$$1/u = n_1 + u_2 + \Delta_2 \tag{12}$$

where n_1 and u_2 are the same as for the exact number u_1. If n_1 is to be obtained from u by the same equation as (3) i.e.,

$$n_1 = [1/u], \tag{13}$$

then the condition which must be satisfied is

$$0 < u_2 + \Delta_2 < 1. \tag{14}$$

Otherwise the n_1 number will be different than for the u_1 and the ER will not be preserved. The $u_2 + \Delta_2$ can be decomposed analogously to $u_1 + \Delta_1$ in (12) and the whole expansion to a continued fraction may be carried on as for the exact number. For each level of the continued fraction we obtain the fractional parts $u_i + \Delta_i$ instead of u_i. So the error Δ_1 introduced to the exact number propagates through levels of the continued fraction. It can be shown that the errors are connected and that

$$\Delta_{i+1} = -\Delta_i/(u_i(u_i + \Delta_i)). \tag{15}$$

In order to get the same ER as for the exact number it is sufficient to take the expansion up to $i = k$ or $i = k+1$ and for each i assume (as in (14)) that

$$0 < u_i + \Delta_i < 1. \tag{16}$$

If this condition is satisfied for each $i = 2, 3, \cdots, k$, the expansion of u has the form

$$\begin{aligned} u = u_1 + \Delta_1 &= 1/(n_1 + u_2 + \Delta_2) \\ &= 1/(n_1 + 1/(n_2 + u_3 + \Delta_3)) \\ &= 1/(n_1 + 1/(n_2 + \cdots + 1/(n_k + \Delta_{k+1}))) \end{aligned} \tag{17}$$

where it was assummed that a k-dimensional ER is necessary. The second possibility will be given below.

The appearance of Δ_{k+1} in (17) changes this expansion from the exact number expansion given by (2). The value of Δ_{k+1} cannot be arbitrary. The limitation of only positive values arises in the same way as the conditions (14) and (16) did

$$0 \leqq \Delta_{k+1} < 1. \tag{18}$$

Keeping in mind that the same fraction l/m may be reached from a $(k+1)$-dimensional ER, we get a limitation on the negative values of the Δ_{k+1}. The formal transition from

a k-dimensional ER to $k+1$ can be written as

$$\begin{aligned} \cdots 1/(n_k+\Delta_{k+1}) &= \cdots 1/(n'_k+1+\Delta_{k+1}) \\ &= \cdots 1/(n'_k+u_{k+1}) \\ &= \cdots 1/(n'_k+1/(n_{k+1}+\Delta_{k+2})) \end{aligned} \tag{19}$$

where

$$n'_k = n_k - 1, \tag{20}$$

$$n_{k+1} = 1+\Delta_{k+1}, \tag{21}$$

$$n_{k+1} = [1/u_{k+1}], \tag{22}$$

$$\Delta_{k+2} = 1/u_{k+1} - n_{k+1}. \tag{23}$$

Because u_{k+1} in the $(k+1)$-dimensional ER must satisfy the condition

$$0 < u_{k+1} \leqq 1, \tag{24}$$

its introduction by (21) makes sense only when the Δ_{k+1} has a negative value and satisfies the condition that

$$-1 < \Delta_{k+1} \leqq 0. \tag{25}$$

Since in the $(k+1)$-dimensional ER, n_{k+1} is equal to one, so, from (22), we have

$$\tfrac{1}{2} < u_{k+1} \leqq 1. \tag{26}$$

This condition limits (25) to

$$-\tfrac{1}{2} < \Delta_{k+1} < 0. \tag{27}$$

Combining (18) and (27) we get a general condition

$$-\tfrac{1}{2} < \Delta_{k+1} < 1 \tag{28}$$

obeying both ERs.

This condition is of great importance in our problem. It enables us to calculate minimum and maximum values of the approximate number u which can be converted to the same fraction l/m. These limiting values could have been calculated by using the recurrence relation (15), but this is inconvenient for our purposes. We notice that the recurrence equations (5) and (6) are correct even if the partial denominators in (2) are not integral and they can be applied to the continued fraction (17). This leads to

$$u = (l_{k-1}\tilde{n}_k + l_{k-2})/(m_{k-1}\tilde{n}_k + m_{k-2}) \tag{29}$$

where

$$\tilde{n}_k = n_k + \Delta_{k+1}. \tag{30}$$

The remaining numbers in (29) are calculated in the same way as for the exact number u_1. In order to get the two limiting values of the approximate number u it suffices to substitute into (29) two limiting values of the Δ_{k+1} from the inequality (28). Because it is more convenient to discuss the error Δ_1 we write

$$\Delta_1 = u - u_1 = u - \frac{l}{m}. \tag{31}$$

Its two limiting values are

$$\Delta_1(-\tfrac{1}{2}) = -(l_{k-1}m - lm_{k-1})/(2m^2 - mm_{k-1}), \tag{32}$$

$$\Delta_1(1) = (l_{k-1}m - lm_{k-1})/(m^2 + mm_{k-1}). \tag{33}$$

These values confine the error of the numbers to be converted to the fraction l/m. Moreover, they also give limiting values of a difference between the approximate number u and the fraction l/m to be obtained. This result gives an important contribution to the algorithm for the conversion. In order to terminate the expansion process of the number u in the place where $l_i = l$ and $m_i = m$, it is necessary to compare the difference $u - l_i/m_i$ for each i with values not exceeding (32) and (33). But direct application of this simple procedure is not effective. First of all, if only the number to be converted is known and not the final fraction, the allowed interval cannot be calculated. The problem may be removed by decreasing the allowed error interval of the numbers to be converted. This will be done in § 4 for the conversion of a set of numbers.

4. Conversion of a set of approximate numbers. Assume that we want to convert numbers to various fractions with denominators from two to a maximum value M. Because the characteristic interval of error cannot be evaluated for each number, we will find one universal interval for the whole set. We take this interval to be the smallest one belonging to the fractions of interest. The negative bound of the interval is given by (32) for the case where

$$l = M - 1, \qquad m = M. \tag{34}$$

In this case

$$m_{k-1} = l_{k-1} = 1 \tag{35}$$

and $k = 2$. Then (32) becomes

$$\Delta_1(-\tfrac{1}{2}) = -1/(2M^2 - M) \equiv -\Delta. \tag{36}$$

A positive bound is also given by (32) for the case when

$$l = 1, \qquad m = M. \tag{37}$$

In this case $k = 1$ and

$$m_{k-1} = m_0 = 1, \qquad l_{k-1} = l_0 = 0. \tag{38}$$

So the (32) becomes

$$\Delta_1(-\tfrac{1}{2}) = +\Delta. \tag{39}$$

Therefore, the maximum common interval is determined by the two fractions, the left bound by $(M-1)/M$ and the right one by $1/M$. The interval can be treated as a universal one for all fractions of interest. All numbers belonging to both the interval $(0, 1)$ and one of the intervals

$$(l/m - \Delta, l/m + \Delta), \tag{40}$$

where Δ is given by (36), and $m = 2, 3, \cdots, M$ and $l = 1, 2, \cdots, M-1$, can be converted to fractions with denominators from two to M. The differences between the fractions and the converted numbers never exceed Δ

$$\left| u - \frac{l}{m} \right| < \Delta. \tag{41}$$

Moreover, the absolute value of the difference between another fraction with a denominator smaller than M and the number u cannot be smaller than Δ. This means that with $i<k$ for a k-dimensional ER or with $i<k+1$ for a $(k+1)$-dimensional ER we have

$$\begin{aligned} |u-l_i/m_i| &= |l/m-l_i/m_i+\Delta_1| \\ &> |\Delta-|l/m-l_i/m_i|| \geqq |\Delta-1/Mm_i| \\ &\geqq |\Delta-1/(M^2-M)| \geqq |\Delta-2\Delta| = \Delta. \end{aligned} \tag{42}$$

The inequalities (41) and (42) determine the termination of the expansion (17) for an approximate number.

5. Summary and conclusions. The expansion (17) together with the recurrence equations (5) and (6) and the inequality (42) form the basis of the algorithm for the conversion of a set of approximate numbers to fractions. If the maximum M of the expected denominator is known, the allowed precision of the converted numbers is

$$\Delta = \pm 1/(2M^2-M). \tag{43}$$

This means that each number to be converted has to satisfy the condition (41). The numbers which do this are exact or approximate in comparison with the fraction l/m to be obtained. They may be called accurate and their conversion single-valued.

The algorithm can be written in the following way:

1. $\Delta = 1/(2M^2-M)$; $v=u$
2. $l_{-1}=1$; $m_{-1}=0$; $l_0=0$; $m_0=1$; $i=0$
3. $i=i+1$; $n_i=[1/v]$
4. $l_i=l_{i-1}n_i+l_{i-2}$; $m_i=m_{i-1}n_i+m_{i-2}$
5. if $|u-l_i/m_i|<\Delta$ then $l=l_i$ and $m=m_i$; end
6. $v=1/v-n_i$; go to step 3.

If the conversion described by the above algorithm is applied to other numbers belonging to the $(0, 1)$ interval but do not satisfy (41), the resulting fractions always have denominators greater than M. These numbers may be called inaccurate. The appearance of a denominator greater than M is a convenient feature when the conversion is applied to a modification of a computer output and indicates the lack of precision of the converted number.

Finally, we discuss the method of forming the ER for a number. It has been assumed that the ER can be obtained by the successive application of (3) and (4). The full solution of this problem gives the Knuth-Schönhage algorithm (see [4], [8], and references therein).

It is worth noting some additional properties of the present conversion. The fractions are always irreducible. This feature is a result of the continued fraction method applied to the conversion [9]. The number of steps for the conversion (which is equal to the dimensions of the ER) is small. The recurrence formulas (5) and (6) show that the number of iterations which are necessary to convert a decimal number to a rational fraction is larger the smaller the partial denominators (3). The greatest number of steps occurs for fractions whose ER contains only ones. The sequence of the numerators and denominators of the rational fractions represented by these ERs with increasing dimension can be obtained immediately from (5) and (6). Each element of the sequence is a sum of two previous ones

$$l_i = l_{i-1}+l_{i-2}, \qquad m_i = m_{i-1}+m_{i-2}. \tag{44}$$

Therefore both sequences are the Fibonacci ones. The fractions l_i/m_i form the sequence which converges to $(\sqrt{5}-1)/2$. The elements of this sequence belong to the interval $\langle \frac{1}{2}, \frac{2}{3} \rangle$. The decimal numbers with values from this interval are converted to the rational fractions which may (but do not need to) have the biggest ERs among all fractions with denominators not exceeding a given value. Thus, these numbers may require the largest number of iterations. From the above consideration one can find easily the maximum number of iterations for a given denominator. It is given by the number (diminished by one) of the elements in the Fibonacci sequence with a value not exceeding the maximum denominator. For example, when the maximum denominator is 100 then the maximum number of iterations is 10. An increase in the denominator by one order of magnitude gives an increase in the maximum number of steps by 4 or 5. Numerical tests have shown that most fractions (from a set of all fractions with denominators not exceeding a given value) have significantly smaller ERs in comparison to the maximum one.

The application of this conversion to the modification of a computer output is not complicated. A subroutine in Fortran language which may be applied for this purpose is included in the Appendix. We have applied the conversion to the matrix elements and coefficients of wave functions in calculations of atomic properties. Having the values expressed as fractions permits them to be manipulated more easily without any accumulated rounding errors.

In some computations intermediate results are calculated from rational functions. Because of rounding (and other reasons) these numbers are only approximate. If the maximum denominator is known and the precision of the intermediate numbers is high enough to make the conversion in a single-valued way, the conversion and, subsequently, the division of the numerator by the denominator result in a higher overall precision. The improvement of the precision is dependent on a difference between the precision of the numbers to be converted and computer precision which limits the final division precision. Situations where this procedure can be applied exist in many computations in physics and chemistry.

Appendix. The presented subroutine was tested on the Amdahl 470 computer. A systematic test was done for fractions with denominators smaller than 1,000. Selected numbers were converted to fractions with denominators up to 10^7. Fractions with larger denominators require the numbers with precision comparable to or higher than available in double precision calculations.

```
      SUBROUTINE CONV (D, UU, NUM, DEN)
C       D       MAXIMUM DENOMINATOR       UU    NUMBER TO BE CONVERTED
C       NUM     NUMERATOR                 DEN   DENOMINATOR
      REAL*8 P, Q, U, UU
      INTEGER D, NUM, DEN
      P=1D0/(2D0*D*D-D)
      U=UU
      L0=1
      M0=0
      L1=0
      M1=1
10    Q=1D0/U
      N=Q
      NUM=L1*N+L0
      DEN=M1*N+M0
      IF(DABS(UU*DEN-NUM).LT.P*DEN)RETURN
      L0=L1
      L1=NUM
```

```
M0 = M1
M1 = DEN
U = Q - N
GOTO 10
END
```

Acknowledgments. The authors are grateful to Marek Gutowski for many helpful discussions and J. D. Murchland for the letter with several comments and additional references concerning the problem.

REFERENCES

[1] J. W. S. CASSELS, *An introduction to Diophantine approximation*, Cambridge Tracts in Math. and Math. Physics, NR 45, Cambridge Univ. Press, Cambridge, 1967, pp. 1-17.

[2] H. DAVENPORT AND K. F. ROTH, *Rational approximations to algebraic numbers*, Mathematika, 2 (Sarajevo) (1955), pp. 160-167.

[3] G. H. HARDY AND E. M. WRIGHT, *An Introduction to the Theory of Numbers*, 4th edition, Clarendon Press, Oxford, 1968.

[4] D. E. KNUTH, *The analysis of algorithms*, Proc. Internat. Congress Math., Nice 1970, Vol. 3, Gauthier-Villars, Paris, 1971, pp. 269-274.

[5] R. PYZALSKI AND M. VALA, ACM Signum Newsletter #4, 18, 22 (1983).

[6] K. F. ROTH, *Rational approximations to algebraic numbers*, Mathematika, 2 (Sarajevo) (1955), pp. 1-20.

[7] SAS *User's Guide*: Basic, 1982 Edition, SAS Institute, Inc., Cary, North Carolina, p. 403.

[8] V. STRASSEN, *The computational complexity of continued fractions*, SIAM J. Comput., 12 (1983), pp. 1-27.

[9] H. S. WALL, *Theory of Continued Fractions*, Van Nostrand, New York, 1967.

SIAM J. SCI. STAT. COMPUT.
Vol. 7, No. 2, April 1986

004

THE USE OF ITERATIVE LINEAR-EQUATION SOLVERS IN CODES FOR LARGE SYSTEMS OF STIFF IVPs FOR ODEs*

TONY F. CHAN† AND KENNETH R. JACKSON‡

Abstract. Systems of linear algebraic equations must be solved at each integration step in all commonly used methods for the numerical solution of systems of stiff IVPs for ODEs. Frequently, a substantial portion of the total computational-work and storage required to solve stiff IVPs is devoted to solving these linear algebraic systems, particularly if the systems are large. Over the past decade, several efficient iterative methods have been developed to solve large sparse (nonsymmetric) systems of linear algebraic equations. We study the use of a class of these iterative methods in codes for stiff IVPs. Our theoretical estimates and preliminary numerical results show that the use of iterative linear-equation solvers in stiff-ODE codes improves the efficiency—in terms of both computational-work and storage—with which a significant class of stiff IVPs having large sparse Jacobians can be solved.

Key words. iterative linear-equation solvers, inexact Newton methods, stiff-ODE solvers, initial value problems

1. Introduction.

As Gear [36], [37] and many others have noted, a major open problem in scientific computing is the efficient solution of large systems of stiff initial-value problems (IVPs) for ordinary differential equations (ODEs) of the form

$$\dot{y}(t) = f(t, y(t)), \qquad y(t_0) = y_0. \tag{1.1}$$

These problems arise either directly in models of physical systems (such as chemical kinetics or electrical networks) or indirectly as a step in the solution of another problem (such as the application of the method-of-lines to a system of parabolic partial differential equations [61]). Consequently, the efficient solution of large systems of stiff IVPs is of great practical importance.

Although several authors have investigated techniques for avoiding implicitness in the numerical solution of stiff IVPs, most workers in the field still agree with Stetter's comment [82] that "all reasonable methods for stiff systems of ODEs have to be implicit," except, possibly, for special classes of problems. That is, a system of linear or nonlinear algebraic equations must be solved at each step of the numerical integration. Moreover, it seems that a Newton-like method must be used to solve the nonlinear systems to avoid a severe restriction on the stepsize. Consequently, large systems of linear equations must be solved in this case as well.

As we explain in more detail in § 4, frequently a substantial portion of the total computational-work and storage required to solve large systems of stiff IVPs is devoted to solving systems of linear algebraic equations. Therefore, any improvement in the efficiency with which these linear systems are solved will directly improve the performance of the integrator. Fortunately, the linear algebraic systems that arise in large systems of stiff IVPs are usually sparse and this property can be exploited to great advantage.

* Received by the editors March 1, 1984, and in final revised form November 15, 1984. This work was supported in part by the U.S. Air Force under grants AFOSR-81-0193 and AFOSR-83-0097, U.S. Department of Energy under grant DE-AC02-81ER10996, and the Natural Sciences and Engineering Research Council of Canada under grants U0133 and A2521.

† Department of Computer Science, Yale University, New Haven, Connecticut 06520.

‡ Department of Computer Science, University of Toronto, Toronto, Ontario, Canada M5S 1A4.

Over the past decade, several efficient iterative methods have been developed to solve large sparse systems of linear algebraic equations. The Krylov subspace methods, of which the conjugate gradient method [44] is a well-known example, have proven to be particularly effective for solving the linear systems that arise in the numerical solution of elliptic and parabolic partial differential equations. (See, for example, [2], [11], [14], [15], [17], [18], [25], [43], [55], [57], [59], [62], [63], [70], [71], [72], [84], [86], [89] and the references therein.) Therefore, it is natural to consider the use of iterative linear-equation solvers in codes for large systems of stiff IVPs for ODEs. Not only are iterative methods faster than direct solvers for many systems of linear algebraic equations, but also they require significantly less storage than direct solvers in most cases. In addition, the use of iterative methods will ease some of the restrictions on the stepsize- and order-selection strategies used in stiff-ODE codes; we believe that this may improve the efficiency of these codes as well.

The outline of the remainder of this paper is as follows. In § 2, we review the numerical solution of the implicit formulas used in many of the most popular stiff-ODE codes, emphasizing the relationship between the user specified error tolerance for the IVP and the accuracy with which the implicit formulas must be solved. We also show that a large class of stiff IVPs have properties that make the associated systems of linear algebraic equations amenable to solution by iterative methods. We then review the "inexact Newton methods" in which the systems of linear equations that arise in Newton's method are solved approximately, rather than exactly. Again, we emphasize the relationship between the accuracy with which the implicit formulas and associated linear algebraic systems must be solved.

In § 3, we review iterative linear-equation solvers with particular emphasis on two Krylov subspace methods: the preconditioned conjugate residual method for symmetric positive-definite systems and preconditioned Orthomin(k) for nonsymmetric positive-real systems.[1] We also point out how these iterative linear-equation solvers can be used in a stiff-ODE code that does not explicitly compute or store the Jacobian associated with the IVP, and, in particular, how the linear systems can be preconditioned in this case.

In § 4, we develop theoretical estimates of the computational work and storage required to solve the spatially-discretized two- and three-dimensional Heat Equation using a stiff-ODE solver that employs either direct or iterative linear-equation solvers. In § 5, we present numerical results for the solution of the spatially-discretized two- and three-dimensional Heat and Convection-Diffusion Equations as well as the 30 Stiff Detest Problems [28], [30] using stiff-ODE solvers based upon either direct or iterative linear-equation solvers. Both the theoretical and numerical results look quite promising.

Finally, in § 6, we present our conclusions.

This paper complements the work of Miranker and Chern [66], Gear and Saad [38], and Brown and Hindmarsh [3], who also studied the use of iterative linear-equation solvers in stiff-ODE codes. We believe our development of the properties of the linear algebraic systems that arise in stiff-ODE solvers that make these linear systems amenable to solution by iterative linear-equation solvers is new, as is our analysis of the relationship between the three tolerances required in a stiff-ODE code employing an iterative linear-equation solver. In addition, our theoretical estimates

[1] A real square matrix A is a positive-real with respect to a real inner-product $(\cdot, \cdot)$ if and only if $(x, Ax) > 0$ for all nonzero real vectors x. Typically, the usual Euclidean inner-product, $(x, y) = \sum x_i \cdot y_i$, is used.

and numerical results extend the work of the authors referenced above, and, in particular, show the importance of preconditioning in the solution of some large systems of stiff IVPs.

Although their point-of-view is distinctly different, the predictor-corrector methods developed and analyzed by van der Houwen and Sommeijer [52] are related to the stiff-ODE methods discussed in this paper and those referenced in the preceding paragraph.

2. Implicit formulas. Many numerical methods have been developed during the past few decades for the solution of systems of stiff IVPs for ODEs, and this continues to be an active area of research. Most of these methods can be classified as being in one of three families: linear multistep (multiderivative) methods, implicit Runge–Kutta methods, and extrapolation methods. Of these, the linear multistep methods have so far proven to be the most successful [28], [30], with the most widely used codes being DIFSUB [34], [35], GEAR [45], EPISODE [5], and LSODE [49], each of which is based upon the Backward Differentiation Formulas (BDFs) popularized by Gear [35]. Therefore, in our discussion of implicit formulas below, we concentrate on the BDFs, although much of what we say applies to stiff methods in general.

A k-step BDF for the solution of (1.1) can be written in the form

$$y_n = \alpha_n^1 y_{n-1} + \cdots + \alpha_n^k y_{n-k} + h_n \beta_n f(t_n, y_n). \tag{2.0.1}$$

Tables of coefficients for these formulas may be found in [35]. To advance the numerical solution from t_{n-1} to $t_n = t_{n-1} + h_n$, (2.0.1) is solved for the approximation y_n to $y(t_n)$ using the previously computed approximations $\{y_{n-i}\}$. Because (2.0.1) is implicit in y_n, an equation of the form

$$F(y_n) = y_n - h_n \beta_n f(t_n, y_n) + c_n = 0 \tag{2.0.2}$$

must be solved at each step of the integration, where c_n contains the terms in (2.0.1) that do not depend upon y_n.

2.1. Accuracy requirement for (2.0.2). In general, (2.0.2) is nonlinear and cannot be solved exactly. Shampine [76], [77] discusses accuracy requirements for this equation. He notes that most stiff-ODE codes attempt to compute an approximate solution, $\tilde{y}_n$, to (2.0.2) satisfying

$$\|y_n - \tilde{y}_n\| \leqq c_1 \cdot \text{TOL}, \tag{2.1.1}$$

where TOL is the user specified error tolerance for the IVP and c_1 is a positive constant (usually less than 1). Shampine [77] presents a convincing argument that, for a stiff-ODE solver, a more appropriate criterion is to accept $\tilde{y}_n$ if

$$\|F(\tilde{y}_n)\| \leqq c_1 \cdot \text{TOL}. \tag{2.1.2}$$

Not only is this criterion more easily related to the accuracy requirement for the IVP, but also it is simpler to implement. Furthermore, Shampine gives an intuitive argument that suggests that, for most stiff problems,

$$\|y_n - \tilde{y}_n\| \leqq \|F(\tilde{y}_n)\|. \tag{2.1.3}$$

However, Houbak, Norsett, and Thomsen [51] demonstrate that it often takes more computational work to satisfy (2.1.2) than (2.1.1) with little or no gain in the accuracy of the numerical solution of the associated IVP. Although we do not address the interesting question of which of these stopping criterion is more appropriate in a stiff-ODE solver, we do develop a bound on $\|y_n - \tilde{y}_n\|$ in terms of $\|F(\tilde{y}_n)\|$ similar to

one given by Williams [87], but using a somewhat different (and possibly simpler) derivation. This bound and some of the relations used in its derivation are important to our discussion of inexact Newton methods and iterative linear-equation solvers below.

The validity of (2.1.3) is intimately related to the stability of the associated IVP. Assume that the IVP satisfies

$$(f(t, y)-f(t, z), y-z) \leqq \gamma(y-z, y-z) \tag{2.1.4}$$

for all (t, y) and (t, z) in the domain of interest, where y and z are real vectors, γ is a real (possibly negative) constant, and $(\cdot, \cdot)$ is a real inner-product. This assumption is frequently made in studying the nonlinear stability of formulas for stiff IVPs as it ensures the stability of the IVP (1.1) in the following sense. Let $y(t)$ be a solution of (1.1) and let $z(t)$ satisfy the same differential equation but have a different initial value, $z(t_0)$. If (2.1.4) is satisfied in a domain containing both $y(t)$ and $z(t)$, then

$$\|y(t)-z(t)\| \leqq e^{\gamma(t-t_0)}\|y(t_0)-z(t_0)\|, \tag{2.1.5}$$

where $\|\cdot\|$ is the norm associated with the inner-product in (2.1.4). We say that the IVP is dissipative if and only if $\gamma<0$. In this case, the IVP is asymptotically stable in the sense that the distance between $y(t)$ and any neighbouring solution of the differential equation, $z(t)$, decreases exponentially with t.

Inequality (2.1.4) can also be used to bound $\|y_n-\tilde{y}_n\|$ in terms of $\|F(\tilde{y}_n)\|$. Assume that (2.1.4) holds at $t=t_n$ in a domain containing both y_n and $\tilde{y}_n$. By (2.0.2) and (2.1.4),

$$(1-h_n\beta_n\gamma)(y-z, y-z) \leqq (F(y)-F(z), y-z). \tag{2.1.6}$$

Hence, if $1-h_n\beta_n\gamma>0$, then, by (2.1.6) and the Cauchy-Schwarz inequality,

$$\|y-z\| \leqq \frac{1}{1-h_n\beta_n\gamma}\|F(y)-F(z)\|.$$

Taking $y=y_n$ and $z=\tilde{y}_n$, we get

$$\|y_n-\tilde{y}_n\| \leqq \frac{1}{1-h_n\beta_n\gamma}\|F(\tilde{y}_n)\|, \tag{2.1.7}$$

from which it follows that, if $\gamma \leqq 0$, then (2.1.3) holds for any $h_n>0$, since $\beta_n>0$ for the BDFs. Note also that, if $1-h_n\beta_n\gamma>0$, then (2.1.7) ensures that any solution of $F(y_n)=0$ is unique in the domain for which (2.1.4) holds. Moreover, if (2.1.4) holds at $t=t_n$ for all real vectors y and z and if $1-h_n\beta_n\gamma>0$, then, by the Uniform Monotonicity Theorem [69], the unique solution of $F(y_n)=0$ exists.

Now we consider in more detail for which class of stiff IVPs we can expect the condition $1-h_n\beta_n\gamma>0$ to hold throughout the course of the numerical integration. First, note that, if the Jacobian $f_y(t, y)$ exists and is continuous, then the algebraically smallest γ for which (2.1.4) holds is

$$\gamma=\max\frac{(f_y(t, y)v, v)}{(v, v)}, \tag{2.1.8}$$

where the maximum is taken over all nonzero real vectors v and all (t, y) in the domain of interest. Hence, $F_y(y_n)=I-h_n\beta_n f_y(t_n, y_n)$ is positive-real if and only if $1-h_n\beta_n\gamma>0$.

It is easy to show that

$$\max\{\mathrm{Re}(\lambda): \lambda \text{ an eigenvalue of } f_y(t, y)\} \leqq \gamma,$$

where $\mathrm{Re}(\lambda)$ is the real part of λ. If $f_y(t, y)$ is symmetric, then equality holds in the

last inequality, but, if $f_y(t, y)$ is nonsymmetric, then the inequality may be strict, as the example below demonstrates. However, the proof of [40, Thm. 1] can be adapted easily to show that, for any fixed (t, y) and any $\varepsilon > 0$, there exists a real inner-product and an associated γ satisfying (2.1.8) for which

$$\gamma - \varepsilon \leqq \max\{\mathrm{Re}(\lambda): \lambda \text{ an eigenvalue of } f_y(t, y)\} \leqq \gamma. \tag{2.1.9}$$

For IVPs having a symmetric Jacobian, it is quite reasonable to expect $1 - h_n\beta_n\gamma > 0$. In fact, for a large subclass of these problems, all the eigenvalues of the Jacobian $f_y(t, y)$ are nonpositive, from which it follows that $\gamma \leqq 0$, whence $1 - h_n\beta_n\gamma \geqq 1$, since $h_n\beta_n > 0$. Thus, (2.1.3) holds. On the other hand, if $\gamma > 0$, then the Jacobian must have a positive eigenvalue arbitrarily close to γ in the domain of interest. Consequently, the differential equation has solutions whose components may grow like $e^{\gamma t}$ in a neighbourhood of the solution of the IVP. Hence, it is reasonable to expect a stiff-ODE solver to choose a stepsize h_n for which $1 - h_n\beta_n\gamma > 0$ to control the accuracy in the potentially growing components of the solution. In fact, if $\gamma > 0$, one would expect $1 - h_n\beta_n\gamma$ to be close to 1, at least if the error tolerance is sufficiently stringent. Moreover, the numerical solution of (2.0.2) requires that $F_y(y_n)$ be "numerically" nonsingular. This effectively ensures that $1 - h_n\beta_n\gamma > 0$, provided that $h_n\beta_n$ changes by small increments, since this inequality holds initially for h_n sufficiently small and, as $1 - h_n\beta_n\gamma$ is essentially the algebraically smallest eigenvalue of $F_y(y_n) = I - h_n\beta_n f_y(t_n, y_n)$, $1 - h_n\beta_n\gamma$ can never approach or pass through zero. Thus, for IVPs having a symmetric Jacobian, it is reasonable to expect that $F_y(y_n)$ will be positive-definite and (2.1.7) will hold in the event that (2.1.3) does not.

If the Jacobian $f_y(t, y)$ is nonsymmetric, then the assumption that $1 - h_n\beta_n\gamma > 0$ is somewhat more problematic, since, for a given inner-product, the associated γ given by (2.1.8) may be much larger than the real part of any of the eigenvalues of $f_y(t, y)$. For example, consider the differential equation $\dot{y} = Ay$, where

$$A = \begin{bmatrix} -1 & 2a \\ 0 & -1 \end{bmatrix}.$$

The eigenvalues of A are both -1, but, for the usual Euclidean inner-product,

$$\gamma = \max \frac{(x, Ax)}{(x, x)} = -1 + |a|$$

can be arbitrarily large even though the eigenvalues of A are fixed. In particular, if $|a| > 1$, then A is not negative-real. Moreover, if a stiff-ODE solver is used to integrate $\dot{y} = Ay$ over a long time interval with absolute error control, the numerical solution will decay exponentially outside of an initial transient region and h_n will become large. Hence, for large t, it is reasonable to expect that $1 - h_n\beta_n\gamma \ll 0$ and $F_y(y_n) = I - h_n\beta_n A$ will not be positive-real. Furthermore, for the usual Euclidean norm, the smallest constant c that ensures that

$$\|y - z\| \leqq c\|F(y) - F(z)\|$$

is $\|(I - h_n\beta_n A)^{-1}\|$, which may be larger than $\sqrt{8a/27}$. Thus, for large a, the residual is not a good estimate of the error for this problem in the usual Euclidean norm. This is not to say that, for the usual Euclidean inner-product and all IVPs having nonsymmetric Jacobians, the condition $1 - h_n\beta_n\gamma > 0$ will be violated and $F_y(y_n)$ will not be positive-real or that the residual will be a poor estimate of the error in (2.0.2), but this is the case for some problems.

On the other hand, even if the Jacobian $f_y(t_n, y_n)$ is nonsymmetric, by an argument similar to the one used in the symmetric case, it follows that it is reasonable to expect that the stepsize, h_n, in a stiff-ODE solver will be restricted by accuracy considerations to the extent that $1 - h_n\beta_n \operatorname{Re}(\lambda) > 0$ for all eigenvalues λ of $f_y(t_n, y_n)$. In fact, for many stiff IVPs, $\operatorname{Re}(\lambda) \leqq 0$ for all the eigenvalues of $f_y(t_n, y_n)$, whence $1 - h_n\beta_n \operatorname{Re}(\lambda) \geqq 1$ without any restriction on the stepsize h_n. In any event, if $1 - h_n\beta_n \operatorname{Re}(\lambda) > 0$ for all eigenvalues λ of $f_y(t_n, y_n)$, then, by (2.1.9), there exists a real inner-product with respect to which $F_y(y_n)$ is positive-real, although this inner-product may depend upon (t_n, y_n). For example, for the matrix A above, if we use the real inner-product $(x, y)_T = (x, Ty)$, where $T = \operatorname{diag}(\delta^2, 1)$ and $(\cdot, \cdot)$ is usual Euclidean inner-product, then

$$\gamma = \max \frac{(x, Ax)_T}{(x, x)_T} = -1 + |a\delta|$$

which, for $\delta = \varepsilon/a$, is within ε of -1. Hence, for $0 < \delta < 1/|a|$, A is negative-real, whence $I - h_n\beta_n A$ is positive-real with respect to the $(\cdot, \cdot)_T$ inner-product for any $h_n > 0$. Although these observations may not be of any practical importance in the selection of an error control strategy for a stiff-ODE solver, we believe that they may be of significance for the implementation of iterative linear-equation solvers, as will become evident in § 3. Moreover, as we explain in that section, the iterative solvers that we consider are guaranteed to converge if $F_y(y_n)$ is positive-real, but may break-down otherwise. Hence, their break-down gives a warning that inequality (2.1.7) is violated.

2.2. Numerical solution of (2.0.2). For nonstiff-ODE solvers, it is common to use functional iteration to solve (2.0.2) or to employ an implicit formula, such as (2.0.1), as the corrector in a predictor-corrector method. However, for stiff-ODE solvers, the use of either of these techniques severely restricts the stepsize and it is exactly this type of restriction that must be avoided for stiff problems. Therefore, in most stiff-ODE solvers, a chord-Newton method is used to solve (2.0.2) at each step in the integration. That is, given an initial approximation y_n^0 to the solution y_n of (2.0.2), the system of linear equations

$$W_n^k(y_n^{k+1} - y_n^k) + F(y_n^k) = 0 \tag{2.2.1}$$

is solved repeatedly until an acceptable approximation y_n^k is computed, where W_n^k is an approximation to the Newton iteration matrix

$$F_y(y_n^k) = I - h_n\beta_n f_y(t_n, y_n^k). \tag{2.2.2}$$

Frequently, W_n^k is just the Newton iteration matrix retained from an earlier iteration on the current or a previous step. Of course, if (1.1) is linear and the exact Newton iteration matrix (2.2.2) is used at each step, then (2.2.1) gives the solution to (2.0.2) in one iteration.

With the exception of GEARBI [48], all the "production" codes for stiff IVPs known to the authors employ direct methods to solve the system of linear algebraic equations (2.2.1). For example, GEAR, EPISODE, and LSODE each use Gaussian Elimination (GE) with partial pivoting, while DIFSUB computes the inverse of W_n^k explicitly. For large systems of stiff IVPs, great savings in both computational-work and storage can be achieved by taking advantage of the sparsity of the Jacobian. This observation led to the development of codes that employ either banded GE (such as GEARB [46], GEARIB [47], and LSODE) or sparse GE (such as GEARS [80] and LSODES [50]).

Furthermore, much of the consideration in choosing the formulas, strategies, and heuristics in a stiff-ODE solver is directed towards solving (2.0.2) as efficiently as possible. To this end, most stiff methods evaluate the Jacobian and refactor W_n^k as seldom as possible, since, as explained in more detail in § 4, the cost of these two operations may dominate the computation. Hence, in most stiff-ODE solvers, W_n^k remains unchanged for several consecutive integration steps.

The desire to avoid refactoring W_n^k also affects the choice of stepsize- and order-selection strategies in a stiff-ODE solver. If the stepsize or order is changed from one step to the next, then at least one of the terms h_n or β_n occurring in the Newton iteration matrix (2.2.2) is changed as well. Therefore, unless W_n^k is updated and refactored, it may be a poor approximation to (2.2.2). As a result, the chord-Newton iteration (2.2.1) may fail to converge or converge too slowly. (Note that this observation applies to linear as well as nonlinear IVPs.) Consequently, the stepsize- and order-selection strategies in most current stiff-ODE solvers are restricted by this consideration. For example, EPISODE changes stepsize and/or order only after a failed step or when it estimates that it can increase its stepsize on the next step by a factor of at least 1.3. In addition to forcing the method to take more steps and function evaluations to integrate a problem than might otherwise be required, this constraint on the order- and stepsize-selection strategies reduces the "smoothness" of the dependence of the actual error committed by the code in solving a problem on the user specified error tolerance; it is generally agreed [37] that such "smoothness" is a very desirable property for an ODE solver to possess.

The choice of variable-stepsize implementation of a multistep formula is also affected by the consideration of how this choice will effect the efficiency of the Newton iteration. The two commonly used implementations are the fixed-coefficient implementation (FCI) of Nordsieck [67], which is used in DIFSUB, GEAR, and LSODE, and the variable-coefficient implementation (VCI), which is used in EPISODE. For nonstiff problems, both theoretical considerations and numerical testing have shown VCI to be superior to FCI for the Adams formulas. (See, for example, [29], [39], [53], [78] and the references therein.)

However, this clear superiority of one implementation over the other for Adams codes does not extend to stiff methods based upon the BDFs. The reason for this seems to be that, when VCI is used with a k-step BDF, the coefficient β_n in (2.2.2) continues to change on each of the $k-1$ steps following a stepsize change. Therefore, unless W_n^k is updated and refactored on each of these steps, it may be a poor approximation to the Newton iteration matrix (2.2.2). On the other hand, FCI does not share this disadvantage, since, for this implementation, β_n is a constant that depends only upon the formula being used. We believe that it is primarily for this reason that the numerical results in [28], [30] indicate that GEAR is more efficient than EPISODE. On the other hand, the numerical results in [6], [7] suggest that EPISODE is more robust than GEAR. This empirical observation is supported by the theoretical results in [39], which show that VCI is more stable than FCI for the BDFs. (See [54] for a more detailed discussion of this topic.)

The cost of solving the implicit equation (2.0.2) also affects the choice of formulas used in a stiff-ODE solver. For example, although A-stable for arbitrarily high orders, the classical implicit Runge–Kutta formulas (IRKFs) [4] suffer the major disadvantage that the implicit system of equations associated with an S-stage formula is S times as large as the corresponding system (2.0.2) for the BDFs.

There has been a considerable effort during the past decade to alleviate some of the difficulties discussed above associated with solving an implicit equation of the form

(2.0.2) at each step of the integration of a stiff ODE. However, one approach that has only recently begun to be investigated actively is the use of iterative methods to solve (2.2.1) [3], [38], [66].

For parabolic PDEs, iterative methods have been popular since the early days of computing: SOR and ADI have been used effectively for several decades [83]. More recently, the conjugate gradient method [2], [17], [18], [57] has received a considerable amount of attention. We believe that the use of iterative methods in stiff-ODE codes should be investigated as well. It appears that these methods offer a great potential for reducing the cost—in terms of both computational work and storage—of solving large systems of stiff IVPs having sparse Jacobians. Furthermore, as discussed in more detail below, the use of iterative methods may alleviate some of the constraints on the stepsize- and order-selection strategies discussed above.

2.3. Inexact Newton methods. To begin, note that the linear equation (2.2.1) is solved only to obtain an approximate solution to the nonlinear equation (2.0.2): there is no reason why a direct linear-equation solver must be used in a stiff-ODE code to solve (2.2.1). Moreover, Sherman [79] and Dembo, Eisenstat, and Steihaug [16] show that it is only necessary to approximate the solution of these linear equations "sufficiently accurately" to obtain a quadratic rate of convergence for the Newton iteration.

More specifically, consider the class of inexact Newton methods [16]. Given an initial guess y_n^0, any such method computes a sequence of values $\{y_n^k\}$ satisfying the recursion

$$\| F_y(y_n^k)(y_n^{k+1} - y_n^k) + F(y_n^k) \| \leqq \eta_k \| F(y_n^k) \|, \tag{2.3.1}$$

where $\eta_k \leqq \eta_{\max} < 1$. In the next section, we discuss the use of iterative methods to compute $(y_n^{k+1} - y_n^k)$ satisfying (2.3.1), but, independently of how y_n^{k+1} is determined, Dembo, Eisenstat, and Steihaug [16] prove that, under the usual assumptions for Newton's method, $y_n^k \to y_n$ with a rate of convergence that is at least linear.

Even though it is not necessary to factor or invert $F_y(y_n^k)$ in an inexact Newton method, it is necessary to evaluate the Jacobian of the IVP, $f_y(t_n, y_n^k)$, to compute $F_y(y_n^k)$ on each iteration. For large problems, the evaluation of the Jacobian may be very expensive, and, consequently, should be avoided whenever possible. Therefore, we consider the class of inexact chord-Newton methods for which (2.3.1) is replaced by

$$\| W_n^k(y_n^{k+1} - y_n^k) + F(y_n^k) \| \leqq \eta_k \| F(y_n^k) \|, \tag{2.3.2}$$

where, as in the previous subsection, W_n^k is an approximation to $F_y(y_n^k)$. But in this case, if an iterative method is used to solve (2.3.2), there is little additional cost associated with using the current value of the scalar $h_n\beta_n$ in W_n^k, although the Jacobian may remain unchanged from one inexact chord-Newton iteration to the next. In any case, the proof of Theorem 2.3 in [16] can be adapted easily to show that $y_n^k \to y_n$ linearly for an inexact chord-Newton method if in addition we assume that W_n^k is a good approximation to $F_y(y_n)$ in the sense that

$$\| W_n^k - F_y(y_n) \| \leqq \gamma \quad \text{and} \quad \| (W_n^k)^{-1} - F_y(y_n)^{-1} \| \leqq \gamma,$$

where γ is the constant appearing in the similar inequalities (2.3) and (2.4), respectively, of [16].

Like a chord-Newton method, the rate of convergence of an inexact chord-Newton method is not superlinear in general. This together with the convergence results quoted above suggest that an appropriate choice for η_k is a constant $\eta < 1$.

In choosing a value for η, it is useful to note that, in many stiff-ODE solvers such as GEAR, EPISODE, and LSODE, y_n^0 is normally a very good initial approximation to y_n in the sense that both $\|y_n^0 - y_n\|$ and $\|F(y_n^0)\|$ are close to TOL, the user specified error tolerance for the IVP, since y_n^0 is computed by an explicit formula of the same order as the implicit corrector. As a result, usually only one or two iterations of (2.2.1) are required to compute y_n^k satisfying either (2.1.1) or (2.1.2). To avoid an excessive number of evaluations of $F(y_n^k)$ when using an inexact chord-Newton method to solve (2.0.2), we also require that, on most steps, only one or two iterations of (2.3.2) be used to compute an acceptable y_n^k. Furthermore, note that

$$F(y_n^{k+1}) \approx F_y(y_n^k)(y_n^{k+1} - y_n^k) + F(y_n^k) \approx W_n^k(y_n^{k+1} - y_n^k) + F(y_n^k).$$

Hence, if $\|F(y_n^0)\| \approx \text{TOL}$ and we want y_n^1 to satisfy (2.1.2), then a reasonable value for η is $r \cdot c_1$, where $r < 1$ is a positive constant and c_1 is the constant appearing in (2.1.2). An alternative is to replace (2.3.2) by

$$\| W_n^k(y_n^{k+1} - y_n^k) + F(y_n^k)\| \leqq r \cdot c_1 \cdot \text{TOL}, \tag{2.3.3}$$

since we require only that y_n^k satisfy the acceptance criterion (2.1.2) and not that y_n^k ultimately converges to y_n.

Based upon the relationship between $\|y_n^k - y_n\|$ and $\|F(y_n^k)\|$ developed in § 2.1, it also seems appropriate to use either (2.3.2) with $\eta = r \cdot c_1$ or (2.3.3) as the acceptance criterion for an inexact chord-Newton method when the acceptance criterion for the implicit equation (2.0.2) is (2.1.1) rather than (2.1.2), although the justification is more tenuous in this case. However, our numerical tests reported in § 5, based upon a modified version of LSODE which employs LSODE's acceptance criterion of the form (2.1.1) for (2.0.2) and the acceptance criterion (2.3.3) for the inexact chord-Newton method, show that this heuristic works quite well in practice.

A stopping criterion of the form (2.3.3) for the inexact Newton method is also used by Brown and Hindmarsh [3] in their modified version of LSODE. In addition, they prove a result about the iterates y_n^k, which, although apparently not tight, suggests that the stopping criterion (2.3.3) is appropriate for stiff-ODE solvers.

Finally, we note that the accuracy of the approximation y_n^k to y_n affects not only the accuracy and stability of the underlying implicit ODE formula [60] but also other formulas, strategies, and heuristics used in the ODE solver. For example, in our preliminary numerical tests with a modified version of LSODE, we found that $y_n^1 = y_n^0$ often satisfied (2.3.3), particularly on the initial steps of the integration. However, accepting $y_n^1 = y_n^0$ has a deleterious effect upon the code, since the error estimate in LSODE is based upon the difference between y_n^0 and the accepted y_n^k and, moreover, the stepsize- and order-selection strategies are based upon the magnitude of the error estimate. Hence, the error may be grossly underestimated and too large a stepsize selected for the next step. We were able to avoid this difficulty in part by taking $-F(y_n^k)$, rather than 0, as the initial guess for $y_n^{k+1} - y_n^k$ in the iterative solution of (2.3.3). Moreover, as this initial guess corresponds to the usual corrector in a predictor-corrector method, it produces a good initial approximation to the nonstiff components of (2.3.3). The choice of a good initial guess for $y_n^{k+1} - y_n^k$ is discussed in more detail for linear systems of IVPs in [66], where, in our notation, they consider initial guesses for $y_n^{k+1} - y_n^k$ of the form $-(I + A + \cdots + A^j)F(y_n^k)$ where $A = I - W_n^k$. Other polynomial approximations to $(I - W_n^k)^{-1}$ could be considered also. However, the effect of the accuracy of the approximation y_n^k to y_n on the formulas, strategies, and heuristics of an ODE solver clearly requires much more study, not only for methods employing inexact Newton methods, but also for all methods based upon implicit formulas.

3. Iterative linear-equations solvers. In this section, we discuss the choice of iterative methods for solving the systems of linear equations that arise in inexact chord-Newton methods (2.3.2). Because these iterative methods must function as a component of a general purpose stiff-ODE solver, it is essential that they perform effectively for general sparse systems of linear equations and are not dependent upon any special matrix properties such as those, for example, associated with the five-point operator for the two-dimensional Laplacian. This consideration immediately eliminates PDE-related methods such as ADI or multi-grid. Moreover, even for the application of the method-of-lines to parabolic problems, many of these PDE-related methods are unsuitable because they require specific information about the PDE itself (e.g., grid structure or operator splittings) which is not usually available to a general-purpose stiff-ODE solver.

Although the classical iterative methods, such as Jacobi, Gauss–Seidel, and SOR, are not restricted to PDE-related problems, they may not converge for general linear systems. Moreover, even when they do converge, these methods are often slow when used on their own and are, therefore, frequently coupled with an acceleration technique to improve their convergence rate. For example, the Symmetric SOR (SSOR) method [88] may be accelerated by either the Chebyshev semi-iteration method or Richardson's second-order method [41]. One undesirable feature of these acceleration techniques is the need to estimate parameters to make them effective. Typically, these parameters depend upon the eigenvalues of the coefficient matrix, which are generally not known to the user a priori. However, adaptive Chebyshev methods, which automatically estimate these parameters, have been developed recently [62], [63] for both symmetric and nonsymmetric problems. These methods may be particularly effective for time-dependent problems, since the coefficient matrix W_n^k of (2.3.2) (and, hence, the associated optimal Chebyshev parameters also) change slowly from step to step throughout the numerical integration of (1.1). Moreover, the Chebyshev iteration is guaranteed to converge if the required parameters are chosen "correctly" and if the real part of each of the eigenvalues of W_n^k is positive. As we argued in the last section, if this last condition is not satisfied, then the stepsize h_n is almost surely too large and should be reduced until this condition is satisfied to ensure a reliable numerical integration. However, to date we have not investigated in depth the use of adaptive Chebyshev methods in stiff-ODE solvers.

The Conjugate Gradient (CG) method [44] is possibly the most well-known example of another class of iterative methods that has received considerable attention recently since being re-introduced by Reid [70] as an iterative method for large sparse systems of linear equations. It has proven to be very effective in this role for a wide range of problems arising from, for example, geophysical applications [74], elliptic PDEs [11], [12], [14], [15], [75], and time-dependent PDEs [2], [17], [18], [57]. We believe that this class of iterative methods is also suitable for solving the linear systems that arise in stiff-ODE solvers. In particular, we consider the Preconditioned Conjugate Residual method [11], [25] and one of its generalizations for nonsymmetric problems, Preconditioned Orthomin [21], [25], [84]. In the remainder of this section, we give a brief description of these methods.

3.1. The preconditioned conjugate residual method. Throughout this subsection, let A be a symmetric positive-definite matrix. To solve the system of linear equations

$$Ax = b, \tag{3.1.1}$$

the Conjugate Residual (CR) method, like CG, requires only that the user supply a

routine to compute the matrix-vector Av for any given vector v. Thus, CR can take full advantage of the sparsity of A. However, the effectiveness of CR can often be improved dramatically by applying CR to the equivalent preconditioned system

$$\hat{A}\hat{x} = \hat{b} \tag{3.1.2}$$

instead of (3.1.1), where $\hat{A} = S^{-1}AS^{-t}$ is a symmetric positive-definite matrix since A is, $\hat{x} = S^t x$, $\hat{b} = S^{-1}b$, and $Q = SS^t$ is "close" to A (in a sense to be made more precise below), but substantially less "expensive" than A to invert. We refer to CR applied to (3.1.2) as the Preconditioned Conjugate Residual (PCR) method and Q as the preconditioner.

One of the several equivalent forms of PCR is given in Fig. 3.1.1. Although any inner-product can be used with PCR, the usual Euclidean inner-product is most often used in practice.

Choose x_0.
Set $r_0 = b - Ax_0$.
Solve $Q\tilde{r}_0 = r_0$.
Set $p_0 = \tilde{r}_0$.
FOR $i = 0$ STEP 1 UNTIL convergence DO
    Solve $Qq_i = Ap_i$.
    $a_i = (\tilde{r}_i, A\tilde{r}_i)/(Ap_i, q_i)$,
    $x_{i+1} = x_i + a_i p_i$,
    $\tilde{r}_{i+1} = \tilde{r}_i - a_i q_i$,
    $b_i = (\tilde{r}_{i+1}, A\tilde{r}_{i+1})/(\tilde{r}_i, A\tilde{r}_i)$,
    $p_{i+1} = \tilde{r}_{i+1} + b_i p_i$.
END FOR

FIG. 3.1.1. *The Preconditioned Conjugate Residual* (PCR) *method.*

If $Q = I$, then PCR reduces to CR. Both methods require the same amount of storage, but, for $Q \neq I$, PCR requires one additonal solve of the form $Qu = v$ per iteration. Note, though, that the matrix S associated with (3.1.2) is not required explicitly. Also, if $Q \neq I$, only the residual $\tilde{r}_i = Q^{-1}(b - Ax_i) = Q^{-1}r_i$ is available in this implementation of PCR; if the residual r_i for (3.1.1) is required also, then either one additional matrix-vector product of the form Qu must be computed per iteration or one additional vector must be stored.

It is well known [1], [11] that PCR is an optimal polynomial-based method in the sense that the ith iterate x_i computed by PCR minimizes

$$\|\hat{r}_i\| = (\hat{r}_i, \hat{r}_i)^{1/2} = (r_i, Q^{-1}r_i)^{1/2} = \|r_i\|_{Q^{-1}} \tag{3.1.3}$$

over the translated preconditioned Krylov subspace

$$x_0 + \langle Q^{-1}r_0, (Q^{-1}A)Q^{-1}r_0, \cdots, (Q^{-1}A)^{i-1}Q^{-1}r_0 \rangle \tag{3.1.4}$$

where $r_i = b - Ax_i$ is the residual for (3.1.1) associated with x_i, for $\hat{x}_i = Sx_i$, $\hat{r}_i = \hat{b} - \hat{A}\hat{x}_i = S^{-1}r_i$ is the corresponding residual for (3.1.2), x_0 is the initial guess for the solution of (3.1.1), and $r_0 = b - Ax_0$ is the associated residual.

The Preconditioned Conjugate Gradient (PCG) method can be implemented in a similar way, but we believe that, for our application, PCR is more appropriate than

PCG. First, note that the inexact chord-Newton method requires that the residual of (2.2.1) satisfy (2.3.2). Therefore, for this problem, CR is the optimal unpreconditioned Krylov subspace method in the sense that it minimizes the norm of the residual over the Krylov subspace (3.1.4) with $Q = I$. On the other hand, CG minimizes the A-norm of the error

$$\|x - x_i\|_A = (x - x_i,\ A(x - x_i))^{1/2} = (r_i,\ A^{-1}r_i)^{1/2} = \|r_i\|_{A^{-1}}, \tag{3.1.5}$$

rather than the residual itself, over the same space (3.1.4). However, this advantage is partially lost for the preconditioned methods, since PCR minimizes $\|r_i\|_{Q^{-1}}$ while PCG minimizes $\|r_i\|_{A^{-1}}$ over (3.1.4). Second, PCR can be generalized more easily than PCG for nonsymmetric problems, partly because (3.1.3) defines a norm for any nonsingular matrix A, provided Q is positive-real, while (3.1.5) does not. Moreover, the preconditioned Krylov subspace methods discussed in the next subsection which extend PCR to nonsymmetric systems are capable of minimizing the residual associated with (3.1.1) provided the preconditioning is applied "on the right". Therefore, we consider PCR only throughout the remainder of this subsection, although similar results hold for PCG.

Since x_i is a member of the affine space (3.1.4), the residual $\hat{r}_i$ associated with (3.1.2) satisfies

$$\hat{r}_i = (I - \hat{A}P_{i-1}(\hat{A}))\hat{r}_0 = R_i(\hat{A})\hat{r}_0, \tag{3.1.6}$$

where P_{i-1} is a polynomial of degree $i-1$ and R_i is a polynomial of degree i that satisfies $R_i(0) = 1$. One can derive from (3.1.3) and (3.1.6) the bound [11]

$$\|\hat{r}_i\| \leqq \left[\min_{R \in \Pi_i} \max_{1 \leqq j \leqq M} |R(\hat{\lambda}_j)|\right] \|\hat{r}_0\| \tag{3.1.7}$$

where Π_i is the set of polynomials of degree i or less that satisfy $R(0) = 1$ and $\{\hat{\lambda}_j\}$ are the eigenvalues of $\hat{A}$, which are also the eigenvalues of $Q^{-1}A$ since these two matrices are similar. Using the ith Chebyshev polynomial as a particular choice for R, one can derive the following bound [1], [11]

$$\|\hat{r}_i\| \leqq 2\left[\frac{1 - 1/\sqrt{K(\hat{A})}}{1 + 1/\sqrt{K(\hat{A})}}\right]^i \|\hat{r}_0\| \tag{3.1.8}$$

where $K(\hat{A}) = \lambda_{\max}(\hat{A})/\lambda_{\min}(\hat{A})$ is the spectral condition number of $\hat{A}$. Since $\|\hat{r}_i\| = \|r_i\|_{Q^{-1}}$, inequalities (3.1.7) and (3.1.8) hold with $\|\hat{r}_i\|$ and $\|\hat{r}_0\|$ replaced by $\|r_i\|_{Q^{-1}}$ and $\|r_0\|_{Q^{-1}}$, respectively.

If A is well-conditioned or the eigenvalues of A are clustered, then CR reduces the error in the initial approximation very rapidly. Therefore, this method can be expected to perform very effectively on the linear equations that arise in mildly-stiff IVPs or in large IVPs for which the eigenvalues of the associated Jacobian form a few clusters. In particular, CR is well suited for problems with a few stiff components only. (See [31], [85] and the references therein for a more detailed discussion of this latter class of problems.) On the other hand, if A is ill-conditioned with its eigenvalues spread throughout a very large interval, then these bounds suggest that CR may require a great many more iterations than PCR to generate an acceptable approximation to the solution of (3.1.1). Since such linear algebraic systems arise during the numerical solution of many large systems of stiff IVPs (in particular, those that arise from the spatial discretization of time-dependent PDEs), we believe that it is necessary to consider effective preconditionings for use with iterative linear-equation solvers in

codes for stiff ODEs. The importance of preconditioning is demonstrated in the next two sections.

Typically, stiff IVPs are nonstiff during an initial transient phase where the accuracy requirements force the stepsize h_n to be small relative to the size of the eigenvalues associated with the problem. This is followed by mildly-stiff and stiff phases as the transient decays and the stepsize increases. In some problems, mildly-stiff or nonstiff phases re-occur. To take advantage of this behaviour, several authors have developed codes that use inexpensive functional iteration during the nonstiff phases of the integration and more expensive Newton-like methods during the mildly-stiff and stiff phases. An important advantage of using either CR or PCR together with an inexact Newton iteration is that the transition from an inexpensive method using only a few (P)CR iterations during the nonstiff and mildly-stiff phases of an IVP to a more expensive method using many (P)CR iterations during the stiff phases takes place automatically and "continuously" as the stiffness of the problem varies.

Among the more popular preconditionings for symmetric positive-definite systems are SSOR [43], [88], the Incomplete Cholesky (IC) factorization [65], and the Modified Incomplete LU (MILU) factorization [42], a generalization of the Dupont-Kendall-Rachford (DKR) factorization [19]. Each of these preconditionings can be written in the form

$$Q = LL^t = A + E,$$

where L is a lower triangular matrix and E is an error matrix. In this paper, we always choose L to have the same sparsity structure as A, although allowing more fill-in in L may yield a more accurate incomplete factorization. Hence, the preconditionings that we consider do not require more storage than the original matrix A and, if implemented carefully [20], may require substantially less. Furthermore, to solve a system $Qu = v$ or to compute Qu for any of the preconditionings that we consider does not require more computational work than multiplying a vector by A and, when embedded in PCR, may require substantially less [20].

As explained in § 2.1, if the Jacobian $f_y(t, y)$ is symmetric, then it is reasonable to expect that the chord-Newton iteration matrix W_n^k (2.2.1) will be symmetric positive-definite. If this is not the case, then the stepsize h_n is almost surely too large for the IVP and should be reduced until W_n^k is positive-definite to ensure a reliable numerical integration. For W_n^k symmetric positive-definite, the SSOR preconditioning is well defined and both the IC and MILU incomplete factorizations can usually be formed [58], [64], [65].

3.2. Preconditioned orthomin. Although both PCG and PCR have proven to be very effective methods for solving symmetric positive-definite systems of linear algebraic equations, only recently have they been extended to solve more general systems effectively. As explained in the previous subsection, the solution of symmetric indefinite systems is not of great importance for stiff-ODE solvers. Therefore, we consider the solution of nonsymmetric systems only in this subsection.

An obvious way to extend either PCG or PCR to solve a nonsymmetric system $Ax = b$ is to apply either of these methods to the symmetric positive-definite normal equations $A^tAx = A^tb$ or to the related system $AA^ty = b$, $x = A^ty$. In either case, though, for the badly conditioned systems that arise in stiff-ODE solvers, this approach is not attractive because it frequently leads to a slow rate of convergence.

Recently, several effective Krylov subspace methods have been developed which extend PCG and/or PCR to nonsymmetric systems. For example, Concus and Golub [13] and Widlund [86] developed a technique known as the Generalized Conjugate

Gradient (GCG) method which uses the symmetric part of A, $S=\frac{1}{2}(A+A^t)$, as a preconditioning. GCG is particularly effective if a "fast" solver exists for S. Although this may be the case for many parabolic problems, this method is not well suited for use in a general-purpose stiff-ODE solver, since there is no guarantee that systems of the form $Sx=b$ can be solved cheaply.

We chose to base our investigation of the use of iterative linear-equation solvers in codes for stiff IVPs upon the Preconditioned Orthomin (k) (POR (k)) method [21], [25], [84], an extension of PCR to nonsymmetric systems, partly because Elman's codes [22], [26] were available to us and partly because, like PCR, POR (k) minimizes the residual associated with the preconditioned system over a subspace described in more detail below. One of several other alternative Krylov subspace methods is discussed by Gear and Saad [38] and Brown and Hindmarsh [3].

In this subsection, we briefly outline POR (k) and the related Preconditioned Generalized Conjugate Residual (PGCR) method from which it is derived; a more detailed discussion of these methods can be found in [21], [25].

Like PCR, the effectiveness of POR (k) can often be improved dramatically by an appropriate choice of preconditioning. However, since POR (k) is applicable to nonsymmetric systems, there is more flexibility in the choice of preconditioning for POR (k) than there is for PCR. More specifically, the preconditioned system associated with (3.1.1) may be of the form

$$\hat{A}\hat{x}=\hat{b} \tag{3.2.1}$$

where $\hat{A}=Q_1^{-1}AQ_2^{-1}$, $\hat{x}=Q_2x$, $\hat{b}=Q_1^{-1}b$, Q_1 and Q_2 are substantially less "expensive" to invert than A, and the preconditioning matrix $Q=Q_1Q_2$ is "close" to A (in a sense to be made more precise below). In this formulation, the preconditioning (3.1.2) used with PCG or PCR is equivalent to a *symmetric positive-definite split preconditioning* having $Q_2=Q_1^t$. Two other particular forms of preconditioning (3.2.1) are worth noting: *preconditioning on the left only* with $Q_2=I$ and *preconditioning on the right only* with $Q_1=I$.

The prototype of the PGCR family of methods from which POR (k) is derived is shown in Fig. 3.2.1. The expression for a_i used there is mathematically equivalent to the expression given in Fig. 3.1.1, but Elman [25] believes that the former is less sensitive to roundoff error for nonsymmetric problems.

Choose x_0.

Set $r_0=b-Ax_0$.

Compute $\hat{r}_0=Q_1^{-1}r_0$.

Compute $p_0=Q_2^{-1}\hat{r}_0$.

FOR $i=0$ STEP 1 UNTIL convergence DO

$a_i=(\hat{r}_i,\ Q_1^{-1}Ap_i)/(Q_1^{-1}Ap_i,\ Q_1^{-1}Ap_i)$,

$x_{i+1}=x_i+a_ip_i$,

$\hat{r}_{i+1}=\hat{r}_i-a_iQ_1^{-1}Ap_i$.

Compute p_{i+1}.

END FOR

FIG. 3.2.1. *The prototype of the Preconditioned Generalized Conjugate Residual* (PGCR) *family of methods.*

The two-term recurrence

$$p_{i+1} = \tilde{r}_{i+1} + b_i p_i, \qquad b_i = \frac{(\tilde{r}_{i+1}, A\tilde{r}_{i+1})}{(\tilde{r}_i, A\tilde{r}_i)}, \tag{3.2.2}$$

used in PCR generates an $\hat{A}^t\hat{A}$-orthogonal sequence of search directions $\{p_i\}$ provided $\hat{A}$ is symmetric positive-definite. However, to obtain such a sequence for $\hat{A}$ nonsymmetric, it appears to be necessary to explicitly orthogonalize p_{i+1} against all previous search directions p_j in general. The recurrence recommended by Elman [21], [25] for PGCR is

$$p_{i+1} = Q_2^{-1}\hat{r}_{i+1} + \sum_{j=0}^{i} b_j^i p_j, \qquad b_j^i = -\frac{(Q_1^{-1}AQ_2^{-1}\hat{r}_{i+1}, Q_1^{-1}Ap_j)}{(Q_1^{-1}Ap_j, Q_1^{-1}Ap_j)}. \tag{3.2.3}$$

PGCR consists of the prototype method given in Fig. 3.2.1 together with these last two equations to compute p_{i+1}.

The recurrence (3.2.3) requires the storage of all past search directions p_j as well as far more computation than (3.2.2). This may be prohibitively expensive for large problems. In POR (k), the truncated recurrence

$$p_{i+1} = Q_2^{-1}\hat{r}_{i+1} + \sum_{j=j_i}^{i} b_j^i p_j, \qquad j_i = \max(0, i-k+1), \tag{3.2.4}$$

is used instead, where b_j^i is computed as in (3.2.3). That is, p_{i+1} is orthogonalized against the past k search directions only. Hence, POR (k) requires the storage of at most k past search directions and the recurrence (3.2.4) is significantly cheaper to compute than (3.2.3).

If $\hat{A}$ is symmetric positive-definite, then $b_i = b_i^i$ and $b_i^j = 0$ for $j < i$ [21], [25]. Consequently, both PGCR and POR (k), $k \geq 1$, are mathematically equivalent to PCR in this case. However, if $\hat{A}$ is not symmetric, then $b_i \neq b_i^i$ in general. Consequently, PCR and POR (1) generate different search directions p_i and different solution vectors x_i in general.

The work per iteration for these preconditioned Krylov subspace methods is the same as for the unpreconditioned versions except that $Q_1^{-1}AQ_2^{-1}\hat{r}_{i+1}$ must be computed in place of Ar_{i+1}. In computing the former product, the intermediate result $Q_2^{-1}\hat{r}_{i+1}$ can be used to compute p_{i+1}, and $Q_1^{-1}Ap_{i+1}$ can be computed without any additional matrix-vector multiples provided that $Q_1^{-1}Ap_j$ is saved instead of p_j. Moreover, for SSOR and several of the incomplete factorizations [42], [65], $Q_1^{-1}AQ_2^{-1}\hat{r}_{i+1}$ can be computed very efficiently using Eisenstat's technique [20].

If $Ax = b$ is preconditioned on the left only ($Q_2 = I$), then each of the PGCR family of methods requires the same amount of storage as its corresponding unpreconditioned version. Otherwise, each preconditioned method requires one more vector of storage than its corresponding unpreconditioned version. However, the residual $\hat{r}_i$ calculated in this implementation of the PGCR family of methods is the residual associated with the preconditioned system (3.2.1). If the residual $b - Ax_i = r_i = Q_1\hat{r}_i$ associated with (3.1.1) is required, then the storage advantage of preconditioning on the left only is lost: in this case, each preconditioned method requires one more vector of storage than its corresonding unpreconditioned version.

If $\hat{A}$ is positive-real, then both PGCR and POR (k), $k \geq 0$, are convergent descent methods in the sense that $\|\hat{r}_i\| \to 0$ as $i \to \infty$ and $\|\hat{r}_{i+1}\| < \|\hat{r}_i\|$ for $\hat{r}_i \neq 0$ [21], [25]. More specifically, PGCR, like PCR, minimizes $\|\hat{r}_i\|$ over the translated Krylov subspace

(3.1.4). Hence, in this case also, the residual $\hat{r}_i$ at the ith PGCR iteration satisfies

$$\|\hat{r}_i\| \leqq \min_{R \in \Pi_i} \|R(\hat{A})\| \, \|\hat{r}_0\|, \tag{3.2.5}$$

where Π_i is the set of polynomials of degree i or less satisfying $R(0) = 1$. Using (3.2.5), one can prove [21], [25] that

$$\|\hat{r}_i\| \leqq \left[1 - \frac{\lambda_{\min}(\hat{S})^2}{\lambda_{\max}(\hat{A}^t\hat{A})}\right]^{i/2} \|\hat{r}_0\|, \tag{3.2.6}$$

where $\hat{S} = \frac{1}{2}(\hat{A} + \hat{A}^t)$, the symmetric part of $\hat{A}$, is positive-definite since $\hat{A}$ is positive-real by assumption.[2] This bound, though, is not nearly as strong as (3.1.8) even though PGCR and PCR compute identical iterates x_i if A is symmetric positive-definite. If A has a complete set of eigenvectors, then

$$\|\hat{r}_i\| \leqq K(\hat{T})\hat{M}_i \|\hat{r}_0\|, \tag{3.2.7}$$

where $K(\hat{T}) = \|\hat{T}\| \, \|\hat{T}^{-1}\|$ is the condition number of the matrix $\hat{T}$ that diagonalizes $\hat{A}$,

$$\hat{M}_i = \min_{R \in \Pi_i} \max_{1 \leqq j \leqq M} |R(\hat{\lambda}_j)|, \tag{3.2.8}$$

and $\{\hat{\lambda}_j\}$ are the eigenvalues of $\hat{A}$. Note, if $\hat{A}$ is normal, then $K(\hat{T}) = 1$.

For $i > k$, the ith iterate x_i computed by POR (k) minimizes $\|\hat{r}_i\|$ over the affine space

$$x_{i-k+1} + \langle p_{i-k+1}, \cdots, p_{i-1} \rangle$$

rather than the full translated Krylov subspace (3.1.4). However, in this case also, (3.2.6) holds for any k.

In all of the bounds listed above, $\hat{r}_i$ may be replaced by $Q_1^{-1} r_i$, since these two vectors are equal. Thus, one advantage of preconditioning on the right only ($Q_1 = I$) is that, in this case, the PGCR family of methods minimizes the residual r_i associated with the unpreconditioned problem $Ax = b$ at each iteration, since $\hat{r}_i = r_i$. As explained in the previous subsection, this seems to be the most appropriate measure of the error to be minimized by an iterative method embedded in an inexact Newton iteration.

Provided that the associated chord-Newton iteration matrix W_n^k is positive-real, these bounds indicate that, like CR, the (unpreconditioned) GCR family of methods is very effective for mildly-stiff IVPs and for stiff IVPs for which the eigenvalues of the associated Jacobian form a few clusters. On the other hand, if the eigenvalues of W_n^k are spread throughout a large domain, then, as is demonstrated in the next two sections, the effectiveness of POR (k) may be improved dramatically by an appropriate choice of preconditioning. Also, like PCR, an important advantage of using either OR (k) or POR (k) together with an inexact Newton iteration is that the transition from an inexpensive method using only a few (P)OR (k) iterations during the nonstiff and mildly-stiff phases of an IVP to a more expensive method using many (P)OR (k) iterations during the stiff phases takes place automatically and "continuously" as the stiffness of the problem varies.

However, care must be taken in choosing a preconditioning since, for the more general preconditionings considered in this subsection, $\hat{A}$ may fail to be positive-real even though A is. One advantage of using the symmetric positive-definite split preconditioning ($Q_2 = Q_1^t$) is that $\hat{A}$ is positive-real if and only if A is. Furthermore, if A is

[2] $0 < (x, \hat{A}x) = (x, \hat{S}x)$ for all real nonzero vectors x, since $\hat{A} = \hat{S} + \hat{N}$ is positive-real and $(x, \hat{N}x) = 0$ for all real x because $\hat{N} = \frac{1}{2}(\hat{A} - \hat{A}^t)$ is skew-symmetric.

"nearly" symmetric, then so is $Q_1^{-1}AQ_1^{-t}$, and, for symmetric problems, the iterates x_i computed by POR (k), $k \geqq 1$, are identical to the iterates computed by PCR and PGCR. Intuitively, if $Q_1^{-1}AQ_1^{-t}$ is "nearly" symmetric, then we expect the convergence rate of POR (k) to be close to that of PGCR. On the other hand, even if A and $Q = Q_1Q_2$ are both "nearly" symmetric, $Q_1^{-1}AQ_2^{-1}$ need not be, and the convergence rate of POR (k) may be significantly slower than that of PGCR.

Some popular preconditionings for nonsymmetric systems are SSOR [43], [88], the Incomplete LU (ILU) factorization [65], and the Modified Incomplete LU ($MILU$) factorization [42]. Each of these preconditionings can be written in the form

$$Q = LU = A + E,$$

where L and U, respectively, are lower and upper triangular matrices having the same sparsity pattern as A. With these factorizations, it is possible to precondition on the left or right or to use a split preconditioning with $Q_1 = L$ and $Q_2 = U$.

POR (k) with these preconditionings has proven to be very effective for solving the systems of linear algebraic equations associated with discretized nonself-adjoint elliptic PDEs. Obviously, the smaller k is the more efficient these methods are in terms of storage required. Elman [25] also found that these methods are most efficient in terms of computational work for $k \leqq 5$, with $k = 1$ often requiring the least amount of work.

As mentioned in § 2.1, W_n^k is positive-real with respect to a given inner-product for any $h_n > 0$ for a large class of stiff IVPs, including all problems that are dissipative with respect to that inner-product. But, for any given inner-product, there are stiff IVPs for which W_n^k is not positive-real with respect to that inner-product for a reasonable choice of stepsize, h_n. In the latter case, any member of the PGCR family of methods based upon that inner-product may either compute an acceptable numerical solution or may "break-down" during the computation.

On the other hand, if all the eigenvalues of the Jacobian $f_y(t, y)$ lie either in the left-half complex plane or on the imaginary axis, then, without any restriction on the stepsize h_n, all the eigenvalues of W_n^k lie strictly in the right-half complex plane. Moreover, as discussed in § 2.1, even if some of the eigenvalues of $f_y(t, y)$ lie in the right-half complex plane, it is reasonable to expect the stepsize h_n to be constrained by the accuracy requirements to the extent that all the eigenvalues of W_n^k will lie strictly in the right-half complex plane. In either case, it follows from (2.1.9) that there exists a real inner-product with respect to which W_n^k is positive-real. We hope to find a computationally effective way to utilize this result to dynamically choose an appropriate inner-product whenever W_n^k is not positive-real with respect to the usual Euclidian inner-product.

3.3. Jacobian-free stiff-ODE solvers. As several authors have noted, it is possible to avoid explicitly computing and storing the Newton iteration matrix W_n^k when solving nonlinear equations by an inexact Newton method coupled with a Krylov subspace method. To implement such a Newton–Krylov method, it is only necessary to be able to compute Jv for any given vector v, where J is an approximation to the Jacobian $f_y(t_n, y_n^k)$. In many stiff-ODE solvers, divided differences are used to form J. But, since J is not needed explicitly, the directional difference

$$[f(t_n, y_n^k + \delta v) - f(t_n, y_n^k)]/\delta$$

can be used to calculate an approximation to $f_y(t_n, y_n^k)v$ directly, where δ is a scalar constant.

Garg and Tapia [33] and O'Leary [68] recently investigated a similar idea for the application of CG to minimization problems. O'Leary shows that, in addition to saving storage, the Newton–CG method employing directional differences requires less computational work than the traditional discrete-Newton method for large problems.

Furthermore, the test results of Brown and Hindmarsh [3] based on the code developed by Gear and Saad [38] demonstrate that the use of directional derivatives to approximate matrix-vector products in a Newton–Krylov iteration is very effective for the spatially-discretized nonlinear parabolic problems that they considered.

All of the preconditionings referenced in the preceding two subsections require an explicit representation of the matrix J. Polynomial preconditionings [56], [73], though, using directional derivative approximations to matrix-vector products, could be implemented without the explicit computation or storage of the matrix J. However, as Saad notes [73], the effectiveness of polynomial preconditionings for CR, GCR, and OR(k) for in-core computations on serial machines is questionable, since these Krylov subspace methods are, in a sense, optimal polynomial methods. Alternatively, Chan and Jackson [8] recently developed a class of nonlinear preconditionings, including a variant of SSOR, that does not require J explicitly and so can be used effectively with Newton–Krylov methods employing directional differences. Moreover, for their test problems, the nonlinear SSOR preconditioning was as efficient as the standard explicit SSOR preconditioning.

Since computing and storing Jacobians is a major source of expense in solving large stiff IVPs, the possibility of avoiding this computation seems very attractive, particularly for those problems for which we can expect a Krylov subspace method to converge very rapidly, such as those IVPs for which the eigenvalues of the associated Jacobian form a few clusters.

4. Theoretical results for the heat equation. The theoretical results in the last section can be adapted easily to show that the use of Krylov subspace methods in stiff-ODE solvers is very effective for a large class of IVPs. As a particular example, in this section, we compare the computational-work and storage required to solve the spatially-discretized Heat Equation by five stiff-ODE solvers each based upon the BDFs but using one of the following methods to solve the systems of linear algebraic equations that arise at each step in the numerical integration: (1) full Gaussian Elimination (GE), (2) band GE, (3) sparse GE, (4) the Conjugate Residual (CR) method, or (5) the Preconditioned Conjugate Residual (PCR) method with either the SSOR [43], [88] or MILU [42] preconditioning. Although we do not advocate using these methods to solve the Heat Equation in practice, the spatially-discretized Heat Equation is a good test problem from a theoretical point-of-view because it is representative of a class of large stiff IVPs with sparse Jacobians and it can be analyzed thoroughly.

Consider the Heat Equation in one dimension (1-D) with homogeneous Dirichlet boundary conditions:

$$
\begin{aligned}
& u_t(t, x) = u_{xx}(t, x) \quad \text{for } (t, x) \in (t_0, t_f] \times (0, 1), \\
& u(t, 0) = u(t, 1) = 0 \quad \text{for } t \in (t_0, t_f), \\
& u(t_0, x) = u_0(x) \quad \text{for } x \in [0, 1].
\end{aligned} \tag{4.1}
$$

Applying the method of lines with the usual centered-difference approximation with stepsize $\Delta = 1/(m+1)$ to the spatial derivative of (4.1) gives the linear system of $M = m$ ODEs $\dot{y} = A_1 y$ for $t \in (t_0, t_f]$ with initial conditions $y_i(t_0) = u(t_0, i\Delta)$ for $i = 1, \cdots, m$, where $y_i(t)$ is an approximation to $u(t, i\Delta)$ and $A_1 = \Delta^{-2}$ diag $(1, -2, 1)$. It is well-known

that the eigenvalues of A_1 are

$$\{2\Delta^{-2}[\cos(i\Delta\pi)-1]: i=1,\cdots,m\}. \tag{4.2}$$

All the eigenvalues are negative and are distributed throughout an interval from approximately $-\pi^2$ to approximately $-4\Delta^{-2}$. As the spatial discretization becomes finer, the resulting system of ODEs becomes both larger and stiffer, but the eigenvalues of the associated matrix A_1 do not cluster.

Also consider a similar spatial discretization of the Heat Equation in two dimensions (2-D) and three dimensions (3-D), each with homogeneous Dirichlet boundary conditions. For the 2-D problem, the matrix A_2 associated with the resulting linear system of $M=m^2$ ODEs $\dot{y}=A_2y$ is $A_2=\Delta^{-2}\,\mathrm{diag}\,(I_1, T_1, I_1)$, where I_1 is the $m\times m$ identity matrix and $T_1=\mathrm{diag}\,(1,-4,1)$. Hence, the eigenvalues of A_2 are

$$\{\lambda_i+\lambda_j: i=1,\cdots,m, j=1,\cdots,m\},$$

where λ_i and λ_j are eigenvalues (4.2) of the 1-D problem. Similarly, for the 3-D problem, the matrix A_3 associated with the resulting linear system of $M=m^3$ ODEs $\dot{y}=A_3y$ is $A_3=\Delta^{-2}\,\mathrm{diag}\,(I_2, B_2, I_2)$, where I_2 is the $m^2\times m^2$ identity matrix, $B_2=\mathrm{diag}\,(I_1, T_2, I_1)$, and $T_2=\mathrm{diag}\,(1,-6,1)$. Hence, the eigenvalues of A_3 are

$$\{\lambda_i+\lambda_j+\lambda_k: i=1,\cdots,m,\ j=1,\cdots,m,\ k=1,\cdots,m\},$$

where λ_i, λ_j, and λ_k are eigenvalues (4.2) of the 1-D problem.

We compare the computational-work per step required by each of the five stiff-ODE solvers considered above to integrate the spatially-discretized 1-D, 2-D, and 3-D Heat Problems. The numerical results presented in the next section show that, for any given problem in this class, each solver requires essentially the same number of steps throughout the numerical integration. Thus, for each solver, the computational-work per step is representative of the total computational-work required. Moreover, implicit in our comparison is the assumption that each stiff-ODE solver requires the same number of Newton iterations per step. The validity of this assumption is supported by the numerical results also.

The computational-work per step can be divided into three components: (1) the work to factor W_n^k for the GE variants or to compute a preconditioning for PCR (if W_n^k is refactored or the preconditioning is recomputed on that step), (2) the work to solve (2.2.1) using either the LU factorization for the GE variants or the (P)CR method, and (3) all the remaining work per step, which is termed the computational-work overhead. We measure the computational-work for each operation in terms of the number of arithmetic operations required to perform it.

Similarly, the storage required by each solver can be divided into two components: (1) the storage required to solve the system of linear algebraic equations and (2) all the remaining storage, which is termed the storage overhead.

The computational-work and storage required for each of the five stiff-ODE solvers is summarized in Table 4.1. For each of the five stiff-ODE solvers considered, both the computational-work and storage overheads are proportional to M, the size of the system of ODEs solved. Moreover, in both cases, the overhead is identical for each solver. For full GE, we assume that no advantage is made of the sparsity of the matrices A_1, A_2, and A_3.

In determining the computational-work and storage required for sparse GE, we assume that the asymptotically optimal ordering of the variables is used, although, frequently, this is not the case in practice for the 2-D and 3-D problems (cf. [23], [24]).

TABLE 4.1

*The principal asymptotic term for the storage for and the computational work per step required to factor and solve the linear algebraic systems that arise during the numerical integration of the spatially-discretized Heat Problem, as well as the overhead of all the remaining storage and computational-work per step required by the stiff-*ODE *solver.*

		full GE	band GE	sparse GE	CR	PCR
1-D	factor	m^3	m	m	-	m
	solve	m^2	m	m	m^2	$m^{1\frac{1}{2}}$
	storage	m^2	m	m	m	m
	overhead	m	m	m	m	m
2-D	factor	m^6	m^4	m^3	-	m^2
	solve	m^4	m^3	$m^2 \log m$	m^3	$m^{2\frac{1}{2}}$
	storage	m^4	m^3	$m^2 \log m$	m^2	m^2
	overhead	m^2	m^2	m^2	m^2	m^2
3-D	factor	m^9	m^7	m^6	-	m^3
	solve	m^6	m^5	m^4	m^4	$m^{3\frac{1}{2}}$
	storage	m^6	m^5	m^4	m^3	m^3
	overhead	m^3	m^3	m^3	m^3	m^3

As stated in § 2.3, to compute a sufficiently accurate solution for the inexact chord-Newton method, it is generally necessary to reduce the initial residual associated with (2.3.3) by a constant factor η only, where η is typically about .1. From (4.2), the spectral condition-number of the Newton iteration matrix $I - h_n\beta_n A_i$, $i = 1, 2, 3$, increases with h_n from 1 at $h_n = 0$ to $4\pi^{-2}\Delta^{-2}$ as $h_n \to \infty$. Hence, from (3.1.8), the number of CR iterations required to reduce the initial residual by a factor of η is at most $\lceil \log(2/\eta)\pi^{-1}\Delta^{-1} \rceil$. In addition, because of the sparsity of A_1, A_2, and A_3, the number of arithmetic operations required for each CR iteration is proportional to M, the dimension of the matrix. Thus, for each of these matrices, the computational-work required to compute a sufficiently accurate solution to (2.3.3) is at most asymptotically proportional to m^2, m^3, and m^4, respectively, and the storage required is asymptotically proportional to m, m^2, and m^3, respectively.

For either the SSOR [43], [88] or MI*LU* [42] preconditioning (with the appropriate choice of scalar parameters), the spectral condition-number of the preconditioned Newton iteration matrix increases with h_n from 1 at $h_n = 0$ to $c\Delta^{-1}$ (for some constant c) as $h_n \to \infty$ [11]. Hence, the number of PCR iterations required to reduce the initial residual by a factor of η is at most asymptotically proportional to $\Delta^{-1/2}$. In addition, because of the sparsity of each Newton iteration matrix and its associated preconditioning, the number of arithmetic operations required for each PCR iteration is proportional to M, the dimension of the system. Thus, in each case, the computational-work required by PCR to compute a sufficiently accurate solution to (2.3.3) is at most proportional to $m^{1\,1/2}$, $m^{2\,1/2}$, and $m^{3\,1/2}$, respectively,[3] and the storage required remains proportional to m, m^2, and m^3, respectively. Moreover, in each case, the work required to compute the MI*LU* factorization is proportional to the number of nonzeros in the matrix, m, m^2, and m^3, respectively, while no work at all is required to "compute" the traditional form of the SSOR "factorization".

[3] For the 1-D problem, PCR preconditioned by MI*LU* (with the associated MI*LU* parameter $\alpha = 0$) converges in one iteration with computational-work proportional to m, since, in this case, the MI*LU* factorization is actually the exact LU factorization of $I - h_n\beta_n A_1$. This result holds for several other incomplete factorizations as well.

The computational-work estimates given in Table 4.1 are biased in favour of the GE variants. During the initial transient for each problem, the stepsize h_n is "small", and the problem is either nonstiff or mildly-stiff. During these phases, the condition number of the Newton iteration matrix $I - h_n\beta_n A_i$, $i = 1, 2, 3$, or the associated preconditioned matrix is not large, and, as a result, the computational work per step for CR and PCR is much smaller than the estimates given above indicate: these estimates are accurate for h_n "large" only. On the other hand, the computational-work required by the GE variants to factor and solve the Newton systems is independent of the stepsize h_n. Although a cheaper fixed-point iteration may be used in the nonstiff phase of the IVP, it is hard to avoid the more expensive Newton iteration in the mildly-stiff phase.

In addition, even for h_n "large", the computational-work estimates for PCR do not appear to be optimal, whereas the estimates for the three GE variants discussed above are optimal. For example, Chan, Jackson, and Zhu [10] show that there is strong evidence that, if the "AD-DKR" preconditioning is used for the 2-D problem, then the condition number of the preconditioned Newton iteration matrix is asymptotically proportional to $\Delta^{-2/3}$ and that the number of PCR iterations required to reduce the initial residual by a constant factor of η is asymptotically proportional to $\Delta^{-1/3}$. Again, because of the sparsity of A_2 and the associated AD-DKR preconditioning, the number of arithmetic operations required for each PCR iteration is proportional to M, the dimension of the system. Thus, there is strong evidence that, for the 2-D problem, the computational-work required by PCR to compute a sufficiently accurate solution to (2.3.3) is at most asymptotically proportional to $m^{21/3}$, rather than $m^{21/2}$. Moreover, both the storage required for PCR and the computational-work needed to compute the AD-DKR incomplete factorization remain asymptotically proportional to m^2.

Table 4.1 shows that, for the 1-D problem, band (sparse) GE is the most effective method. On the other hand, for the 2-D problem, both CR and PCR require asymptotically less storage than any of the GE variants and are asymptotically faster than either full or band GE. However, it is not clear which of PCR or sparse GE is asymptotically faster. The answer to this question depends on how frequently the linear systems must be refactored as the stepsize increases during the course of the numerical integration when sparse GE is used as well as the proportion of steps taken during the mildly-stiff phase of the IVP, where h_n is "small" and PCR requires less computational-work per step than the estimates in Table 4.1 indicate. Numerical results concerning these two questions are given in the next section. For the 3-D problem, though, PCR is asymptotically faster than any of the other methods and requires significantly less storage than any of the GE variants.

5. Numerical results. We have replaced the direct linear-equation solvers in LSODE [49] by PCGPACK, a collection of preconditioned Krylov subspace methods implemented by Elman [22], [26]. We refer to the resulting experimental code as LSODCG. In this section, we report some preliminary numerical experiments with LSODCG to test the effectiveness of iterative linear-equation solvers in codes for large systems of stiff IVPs for ODEs. In particular, we compare the performance of LSODCG and LSODES[4] [50] on two pairs of spatially-discretized two- and three-dimensional linear parabolic problems as well as the performance of LSODCG and LSODE on the thirty Stiff Detest Problems [28], [30]. Although most of the Stiff Detest Problems are not large, they do test the robustness of the inexact chord-Newton method and

[4] LSODES is a variant of LSODE incorporating the Yale Sparse Matrix Package [23], [24] to solve the systems of linear algebraic equations by a sparse direct method.

the associated iterative linear-equation solvers used in LSODCG. These preliminary test results look quite promising.

5.1. LSODE, LSODES, and LSODCG. We developed two variants of LSODCG; LSODCG.V1 and LSODCG.V2. In the former, we did not make any modifications to the formulas, strategies, or heuristics used in LSODE other than those modifications that were necessary to interface LSODE and PCGPACK, such as changing the data structure for storing matrices in LSODCG to the sparse "*IA-JA-A*" representation used in PCGPACK and many other sparse linear-equation solvers. In LSODCG.V2, we made one additional modification to LSODE in hope of reducing the number of inexact chord-Newton iterations and associated function evaluations throughout the course of the numerical integration: each time $h_n\beta_n$ is changed in LSODCG.V2, this scalar factor is updated in the Newton iteration matrix $I - h_n\beta_n J$ without re-evaluating J, and, if a preconditioner is being used in PCGPACK, it is recomputed. Since the Jacobian approximation J is not re-evaluated, these updates are relatively cheap compared to solving the associated linear algebraic equations.[5] In LSODE, LSODES, and LSODCG.V1, on the other hand, the Newton iteration matrix is updated only when the magnitude of the relative change in $h_n\beta_n$ is greater than CCMAX, a constant set to 3. Whenever the Newton iteration matrix is updated, either it is refactored in LSODES, or, if a preconditioner is being used in PCGPACK, the preconditioner is recomputed in LSODCG.V1. In LSODE and LSODCG.V1, the Jacobian approximation J is re-evaluated whenever the Newton iteration matrix is updated; in LSODES, the Jacobian is re-evaluated only when it is estimated to be a poor approximation to the current Jacobian.

In all four codes, the acceptance criterion for the Newton iteration is of the form (2.1.1) with $c_1 = \text{CONIT} = .5/(NQ+2)$, where NQ is the order of the BDF in use. In both variants of LSODCG, we use a stopping criterion for the iterative linear-equation solver in the inexact chord-Newton method of the form (2.3.3). Our numerical experiments show that any r in the range $[.1, .5]$ is quite satisfactory: smaller values of r in this range lead to more PCGPACK iterations per inexact chord-Newton iteration, but frequently lead to fewer inexact chord-Newton iterations resulting in fewer function evaluations. Some numerical results along this line are reported in the third subsection. Also, as mentioned in § 2.3, we take $-F(y_n^k)$, rather than 0, as an initial guess for $y_n^{k+1} - y_n^k$ in (2.3.3) for both variants of LSODCG.

5.2. Spatially-discretized linear parabolic problems. Consider the Heat Equation in two dimensions (2-D)

$$u_t = u_{xx} + u_{yy} \tag{5.2.1}$$

and three dimensions (3-D)

$$u_t = u_{xx} + u_{yy} + u_{zz} \tag{5.2.2}$$

and the Convection-Diffusion Equation in 2-D

$$u_t = u_{xx} + u_x + u_{yy} + u_y \tag{5.2.3}$$

and 3-D

$$u_t = u_{xx} + u_x + u_{yy} + u_y + u_{zz} + u_z \tag{5.2.4}$$

[5] Updating the Newton iteration matrix would be even cheaper if LSODCG.V2 stored $(1/h_n\beta_n)I - J$ rather than $I - h_n\beta_n J$. This change is easy to implement.

each with homogeneous Dirichlet boundary-conditions either on the unit square $[0, 1]\times[0, 1]$ for the 2-D problems or on the unit cube $[0, 1]\times[0, 1]\times[0, 1]$ for the 3-D problems and initial conditions for $t\in[0, 10.24]$

$$u(0, x, y) = 16x(1-x)y(1-y)$$

for the 2-D problems and

$$u(0, x, y, z) = 64x(1-x)y(1-y)z(1-z)$$

for the 3-D problems. As described in § 4, applying the method of lines to the Heat Equation with $m+1$ evenly spaced grid points in each dimension ordered in the usual left-to-right bottom-to-top manner and using the usual three-point second-order centered-difference approximation to the second-order spatial derivatives with stepsize $\Delta = 1/(m+1)$ yields a system of stiff ODEs of the form

$$\dot{y}(t) = Ay(t), \tag{5.2.5}$$

where A is a constant symmetric negative-definite matrix with $A = A_2$ of dimension $M = m^2$ for the 2-D problem and $A = A_3$ of dimension $M = m^3$ for the 3-D problem. Applying the method of lines to the Convection-Diffusion Equation in a similar way, but with the addition of the two-point second-order centered-difference approximation to the first-order spatial derivatives, also yields a system of stiff ODEs of the form (5.2.5), where again A is a constant matrix of dimension $M = m^2$ for the 2-D problem and of dimension $M = m^3$ for the 3-D problem. In this case, though, A is a nonsymmetric negative-real matrix for both the 2-D and 3-D problems.

The eigenvalues and eigenvectors of the matrix A associated with the spatially-discretized Heat Equation (5.2.5) are well known. Therefore, the exact solution of the associated IVP can be calculated easily for any t. For the Convection-Diffusion Problem, we used EISPACK [32], [81] in double precision on an IBM 3033 to calculate the eigenvalues and eigenvectors of the matrix A associated with the spatially-discretized 1-D problem of the form (5.2.5). Since the solution of the spatially-discretized 2-D and 3-D problems can be written as the tensor product of solutions of the associated 1-D problems, the exact solution of the spatially-discretized 2-D and 3-D Convection-Diffusion Problems can be computed easily for any t also.

We used LSODES, LSODCG.V1, and LSODCG.V2 on an IBM 3033 computer in double precision to compute numerical solutions of the 2-D problems for $m = 5$, 10, 15, 20, 25, 30 and the 3-D problems for $m = 3$, 5, 7, 9. In each case, we used the BDFs with exact Jacobians (MF = 21) and an absolute local error tolerance of ATOL = 10^{-3} (ITOL = 1 and RTOL = 0). We integrated from the initial point $t = 0$ to the output points $T = 2^i/100$, for $i = 0, 1, 2, \cdots, 10$, using the continuation option (ISTATE = 2) to integrate from one intermediate output point to the next. Because we did not require the output points to be hit exactly (ITASK = 1), the solution vector is computed by interpolation and, on occasion, more than one solution vector is computed per integration step, as can be seen in some of the numerical results presented below. No optional input (IOPT = 0) was used.

We used the PCGPACK implementation of the Preconditioned Conjugate Residual (PCR) method [26] and the Preconditioned Orthomin (k) (POR (k)) method [22], [26] for $k = 1, 3, 5$ to solve the linear algebraic equations in LSODCG. For each of these methods, we used one of the three PCGPACK preconditionings:

1. NOPRE—no preconditioning,

2. TCSSOR—the two-cyclic implementation [20] of the SSOR preconditioning, or
3. TCDKR—the two-cyclic implementation [20] of the DKR [19] incomplete factorization, more generally referred to as the Modified Incomplete LU (MILU) factorization [42].

For the TCSSOR preconditioning, we used $\omega = 2/[1+\sin(\pi\Delta/2)]$, where $\Delta = 1/(m+1)$ is the spatial stepsize. This value of ω is "near optimal" [88] for the spatially-discretized 2-D and 3-D Heat Equations. Although this value of ω may not be "near optimal" for the spatially-discretized Convection-Diffusion Equation, it is appropriate in this case as well, since, in practice, an optimal value of ω for the problem to be solved is typically not known. For the TCDKR preconditioning, we used $\alpha = 0$ for all problems, as recommended by Chandra [11]. In Orthomin (k), the preconditioning was applied on the right as described in § 3.2. In both variants of LSODCG, we used a stopping criterion of the form (2.3.3) with $r = .5$ for each iterative linear-equation solver. However, we also set the maximum PCGPACK iterations permitted to solve any one linear system to max $(100, 10m)$.

5.2.1. Detailed numerical results for one problem. Detailed results for the numerical solution of the spatially-discretized 2-D Convection-Diffusion Problem with $m = 30$ using LSODES, LSODCG.V1, and LSODCG.V2, respectively, are given in Tables 5.2.1.1, 5.2.1.2, and 5.2.1.3. The linear-equation solver used in LSODCG is POR(1) preconditioned by TCDKR. These numerical results are representative of the performance of these three codes on the problems considered in this subsection.

In each table,

—T is the output point,

—ERROR is the root-mean-square norm[6] of the difference between the numerical and exact solutions to the problem at T,

—HU and NQU, respectively, are the stepsize and order used by the BDF in the last step taken to reach T, and

—NST, NFE, and NJE, respectively, are the total number of steps, function evaluations, and Jacobian evaluations used from the initial point $t = 0$ to the current output point T.

Note also that NFE − 1 is the number of Newton iterations used from the initial point $t = 0$ to the current output point T, since all but the first function evaluation is associated with a Newton iteration. For LSODES, NLU, MLTFAC, and MLTSLV, respectively,

TABLE 5.2.1.1

LSODES *solution of the spatially-discretized* 2-D *Convection-Diffusion Problem on an* $m \times m$ *grid with* $m = 30$.

T	ERROR	HU	NQU	NST	NFE	NJE	NLU	MLTFAC	MLTSLV	MLTTOT
0.010	0.826D-03	0.278D-02	2	8	11	1	3	611382	203220	814602
0.020	0.570D-03	0.482D-02	3	11	15	1	4	815176	284508	1099684
0.040	0.454D-03	0.738D-02	3	14	18	1	5	1018970	345474	1364444
0.080	0.413D-03	0.137D-01	3	19	24	1	6	1222764	467406	1690170
0.160	0.101D-03	0.186D-01	4	24	29	1	6	1222764	569016	1791780
0.320	0.166D-03	0.301D-01	3	30	36	1	7	1426558	711270	2137828
0.640	0.483D-05	0.131D+00	1	35	41	1	9	1834146	812880	2647026
1.280	0.127D-05	0.123D+01	1	36	42	1	10	2037940	833202	2871142
2.560	0.455D-07	0.123D+01	1	37	43	1	10	2037940	853524	2891464
5.120	0.280D-08	0.123D+02	1	38	44	1	11	2241734	873846	3115580
10.240	0.143D-08	0.123D+02	1	38	44	1	11	2241734	873846	3115580

Storage required by YSMP: STRMAT = 4380, STRFAC = 20322, STRTOT = 59306.

[6] The root-mean-square norm on an n-vector x is $\|x\| = ((1/n)\sum_{i=1}^{n} x_i^2)^{1/2}$.

TABLE 5.2.1.2
LSODCG.V1 *solution of the spatially-discretized* 2-D *Convection-Diffusion Problem on an* $m \times m$ *grid with* $m = 30$.

T	ERROR	HU	NQU	NST	NFE	NJE	NPRE	ITSTOT	ITSMAX	MLTTOT
0.010	0.830D-03	0.279D-02	2	8	11	3	3	16	3	257483
0.020	0.568D-03	0.482D-02	3	11	15	4	4	24	3	381455
0.040	0.463D-03	0.742D-02	3	14	19	5	5	33	3	519649
0.080	0.414D-03	0.137D-01	3	19	25	6	6	52	5	803421
0.160	0.106D-03	0.188D-01	4	24	30	6	6	71	5	1082034
0.320	0.115D-03	0.275D-01	3	30	37	7	7	115	10	1723035
0.640	0.462D-05	0.120D+00	1	35	42	9	9	157	10	2335714
1.280	0.139D-05	0.108D+01	1	36	43	10	10	175	18	2596869
2.560	0.418D-07	0.108D+01	1	37	44	10	10	188	18	2783434
5.120	0.113D-07	0.108D+02	1	38	45	11	11	211	23	3115699
10.240	0.509D-08	0.108D+02	1	38	45	11	11	211	23	3115699

Storage required by PCGPACK: STRMAT = 4380, STRPRE = 900, STRTOT = 15963.

TABLE 5.2.1.3
LSODCG.V2 *solution of the spatially-discretized* 2-D *Convection-Diffusion Problem on an* $m \times m$ *grid with* $m = 30$.

T	ERROR	HU	NQU	NST	NFE	NJE	NPRE	ITSTOT	ITSMAX	MLTTOT
0.010	0.835D-03	0.277D-02	2	8	10	1	4	15	3	245062
0.020	0.575D-03	0.488D-02	3	11	13	1	5	20	3	324689
0.040	0.467D-03	0.738D-02	3	14	16	1	6	28	3	446982
0.080	0.414D-03	0.138D-01	3	19	21	1	7	44	4	686409
0.160	0.872D-04	0.187D-01	4	24	27	2	9	67	5	1030549
0.320	0.177D-03	0.304D-01	3	30	33	2	10	97	6	1470763
0.640	0.402D-05	0.134D+00	1	35	38	2	12	135	11	2026554
1.280	0.110D-05	0.134D+01	1	36	39	2	13	155	20	2316153
2.560	0.483D-07	0.134D+01	1	37	40	2	13	169	20	2516940
5.120	0.342D-07	0.134D+02	1	38	41	2	14	194	25	2877649
10.240	0.188D-07	0.134D+02	1	38	41	2	14	194	25	2877649

Storage required by PCGPACK: STRMAT = 4380, STRPRE = 900, STRTOT = 15963.

are the total number of
—*LU* factorizations used,
—multiplies used in the *LU* factorizations, and
—multiplies used in forward and backward substitutions
by the Yale Sparse Matrix Package to solve the linear equations that arise in the numerical integration from the initial point $t = 0$ to the output point T. For LSODCG, NPRE and ITSTOT, respectively, are the total number of
—preconditionings computed, and
—iterations used by the linear-equation solvers
to integrate from the initial point $t = 0$ to the output point T. ITSMAX is the maximum number of iterations used to solve any one system of linear equations in integrating from the initial point $t = 0$ to the output point T. For each ODE solver, MLTTOT is the total number of multiplies used to solve the linear equations from the initial point $t = 0$ to the output point T; for LSODES, MLTTOT = MLTFAC + MLTSLV.

Also shown in these tables is the storage required by each of the three ODE solvers. In each case, STRMAT is the number of nonzeros in the matrix A associated with the ODE (5.2.5). For LSODES, STRFAC is the number of nonzeros in the LU factorization computed by YSMP. For LSODCG, STRPRE and STRMTH, respectively, are the number of nonzeros required to store the preconditioning (M for TCSSOR or TCDKR and 1 for NOPRE) and the additional storage used in the iterative method ($[4+2k]M+2k$ for POR (k) and $4M$ for PCR). For both LSODES and LSODCG, STRTOT is the total number of storage locations required for the linear equation

solvers.[7] For LSODES, STRTOT = 2*STRMAT + 2*STRFAC + 11*M + 2, while, for LSODCG, STRTOT = 2*STRMAT + STRMTH + STRPRE + M + 1.

The values of ERROR, HU, NQU, and NST are very similar for all three codes. From this we deduce that, for this class of problem at least, the error-control, stepsize-selection, and order-selection strategies in LSODE are not significantly affected by the use of an iterative linear-equation solver. Although NFE also is similar for all three codes, it is 7–10% smaller for LSODCG.V2 than for either of the other two codes indicating that the use of the current value of $h_n\beta_n$ in the Newton iteration matrix $I - h_n\beta_n J$ reduces slightly the total number of Newton iterations required throughout the integration.

The difference in NJE for LSODES and LSODCG.V1 demonstrates the superiority, for this class of problem at least, of the strategy used in LSODES over the one used in LSODCG.V1 (taken without modification from LSODE) for determining when a Jacobian re-evaluation is required. LSODCG.V2 uses two, rather than one, Jacobian evaluations because we did not alter LSODEs strategy that forces a Jacobian re-evaluation every MSBP ($=20$) steps. If this requirement were removed from LSODCG.V2, then it too would use only one Jacobian evaluation throughout the course of the integration, as it should for this class of problem.

The sequence of values of NLU for LSODES and both NJE and NPRE for LSODCG.V1 are identical indicating that both codes had the same number of "significant" changes in $h_n\beta_n$ between each pair of output points, where by a "significant" change we mean that the magnitude of the relative change in $h_n\beta_n$ is greater than CCMAX ($=3$). The values of NPRE for LSODCG.V2 are slightly larger than those for LSODCG.V1. Thus, assuming that the stepsize sequences in all three codes were similar, there were some changes of $h_n\beta_n$ in LSODES and LSODCG.V1 that were not "significant". Consequently, on some steps in LSODES and LSODCG.V1, the factor $h_n\beta_n$ in the Newton iteration matrix $I - h_n\beta_n J$ was not equal to the value of $h_n\beta_n$ used in the BDF on that step. On the other hand, LSODCG.V2 updates the factor $h_n\beta_n$ in the Newton iteration matrix whenever $h_n\beta_n$ changes. This may explain why LSODCG.V2 used fewer Newton iterations (NFE − 1) than either of the other two codes. As a result, MLTTOT is smaller for LSODCG.V2 than LSODCG.V1 at each output point even though LSODCG.V2 re-computed the TCDKR preconditioner more frequently than LSODCG.V1 did; the reduction in the number of Newton iterations and associated linear-system solves more than offset the additional preconditioner computations.

The final value of MLTTOT is approximately the same for all three codes. However, during the initial nonstiff and mildly-stiff phases of the integration, which last until approximately $T = 0.160$ and require more than half of the steps used throughout the integration, MLTTOT for the two variants of LSODCG is significantly less than for LSODES. For these steps, h_n is relatively small and, consequently, only a few POR iterations (ITSMAX) are required to solve each linear system. However, as the integration proceeds and h_n grows, the spectrum of $I - h_n\beta_n J$ expands and more iterations are required to solve each linear system. However, for $h_n > 1$, ITSMAX does not grow significantly with h_n, since, as a rule of thumb, it is the relative size of the eigenvalues

[7] We count each double precision and integer variable as one storage location although, on the IBM 3033, each double precision variable requires two words of storage whereas each integer variable requires only one. However, this makes little difference in the comparison of the storage required by iterative and direct linear-equation solvers since, for a given problem, both techniques use approximately the same proportion of integer to double precision variables.

to one another, rather than the absolute size of the eigenvalues, that determines the rate of convergence of most Krylov subspace methods, and the relative size of the eigenvalues does not change significantly with h_n for $h_n > 1$. For LSODES, MLTFAC is approximately two-thirds of MLTTOT, and this factor grows as the grids become finer.

Although, for this problem, each code requires approximately the same amount of computational-work, STRTOT for LSODES is about four times the value of STRTOT for either variant of LSODCG. Moreover, this factor grows as the grids become finer. In addition, note that, for LSODES, STRFAC is about five times as large as STRMAT. On the other hand, for LSODCG with POR(1) preconditioned by TCDKR (or TCSSOR), STRPRE is about one eighteenth of STRTOT, since the TCDKR (or TCSSOR) preconditioner requires only one M-vector of storage.

5.2.2. A summary of the numerical results for all the test problems. We present below a summary of the numerical results for LSODES and LSODCG.V2 using the PCGPACK linear-equation solvers PCR and POR (k), $k = 1, 3, 5$, preconditioned by NOPRE, TCSSOR, and TCDKR for the four spatially-discretized parabolic problems on $m \times m$ grids, $m = 5, 10, 15, \cdots, 30$, for the 2-D problems, and $m \times m \times m$ grids, $m = 3, 5, 7, 9$, for the 3-D problems. Somewhat more detailed numerical results are presented in [9]. Since the numerical results for LSODCG.V2 are similar to, but generally better than, those for LSODCG.V1, we have not included a summary of the numerical results for the latter code in either paper.

The computational-work required by LSODES and LSODCG.V2 to solve the linear algebraic systems throughout the integration is presented in Tables 5.2.2.1 and 5.2.2.2. In these tables, we actually list the total number of multiplies, MLTTOT, divided by 1,000 and rounded to the nearest integer for a subrange of m-values: $m = 10$, 20, 30 for the 2-D problems and $m = 5$, 7, 9 for the 3-D problems. Graphs of m against MLTTOT on a log-log scale for LSODES and LSODCG.V2 with POR (1) preconditioned by NOPRE and TCDKR are given in Plots 5.2.2.1 to 5.2.2.4 for all four problems for their full range of m-values.

The total storage, STRTOT, required by the linear-equation solvers in LSODES and LSODCG.V2 for the 2-D and 3-D problems is given in Table 5.2.2.3 for the same subrange of m-values. (Each linear equation solver requires the same amount of storage for both of the 2-D problems as well as the same amount of storage for both of the 3-D problems.) Graphs of m against STRTOT on a log-log scale for LSODES and LSODCG.V2 with POR (1) preconditioned by NOPRE and TCDKR (or TCSSOR) are given in Plots 5.2.2.5 and 5.2.2.6 for the 2-D and 3-D problems for their full range of m-values.

An entry of "*" in place of a number in these tables indicates that, during the course of the integration, the associated iterative linear-equation solver failed to converge in the maximum number of iterations allowed, max (100, $10m$). Only PCR with no preconditioning failed to converge, and it failed on the spatially-discretized 2-D Convection-Diffusion Problem with $m = 10$ and 15 only. It is in fact surprising that PCR did not fail on more of the Convection-Diffusion Problems, since the linear systems associated with these problems are nonsymmetric and PCR is not (in theory at least) applicable to such systems.

Consider the results for LSODCG.V2 first. For these test problems, POR (1) is the most effective of the four basic PCGPACK methods considered. For a given problem and preconditioning, MLTTOT for POR (k), $k = 1, 3, 5$, generally increases with k even though the total number of PCGPACK iterations required often decreases with k: the reduction in the number of iterations is more than offset by the additional

work required per iteration as k increases. As mentioned above, PCR failed on two problems and is not guaranteed to converge for any nonsymmetric linear system. Furthermore, for the symmetric Heat Problems, PCR is not significantly more efficient than POR (1). On the contrary, when preconditioned, PCR frequently requires more multiplies than POR (1) since, even though PCR may require fewer iterations, fewer multiplies are required per iteration to precondition POR (1) on the right than to precondition PCR symmetrically, as is required for the latter method.

Of the three preconditionings, TCDKR is nearly always the most effective in terms of both multiplies and iterations required. The effectiveness of preconditioning is much more pronounced for the nonsymmetric Convection-Diffusion Problems than for the symmetric Heat Problems. In fact, for the latter class of problems, MLTTOT for TCSSOR is frequently larger than for NOPRE for the same basic PCGPACK method, since the additional work required to precondition is not offset by a sufficient reduction in the number of PCGPACK iterations used throughout the integration.

This is not the case for TCDKR. Although NOPRE required fewer multiplies than TCDKR on some coarse grid problems, the difference is never significant. On the other hand, TCDKR is frequently substantially more effective than NOPRE in terms of both multiplies and iterations required by PCGPACK, and, moreover, Plots 5.2.2.1 and 5.2.2.2 show that this difference grows with m.

Although the maximum number of PCGPACK iterations required for any one linear-system solver during the course of the integration by TCDKR and TCSSOR are frequently close, the total number of PCGPACK iterations required by TCDKR is usually significantly less than that required by TCSSOR, indicating that TCDKR is substantially more efficient than TCSSOR on the large number of linear algebraic systems for which h_n is small and the spectrum of $I - h_n\beta_n J$ is clustered around 1.

Now compare LSODES and LSODCG.V2 with POR (1) preconditioned by TCDKR. Tables 5.2.2.1 and 5.2.2.2 and Plots 5.2.2.1 to 5.2.2.4 reveal that LSODES requires fewer multiplies than LSODCG.V2 for the 2-D problems, except on the finest

TABLE 5.2.2.1

The computational-work required by LSODES *and* LSODCG.V2 *to solve the linear algebraic systems throughout the numerical integration of the spatially-discretized* 2-D *and* 3-D *Heat Problem on a* $m \times m$ *and* $m \times m \times m$ *grid, respectively, measured in terms of the total number of multiplies,* MLTTOT, *divided by* 1,000 *and rounded to the nearest integer.*

Method	2-D Problem			3-D Problem		
	m			m		
	10	20	30	5	7	9
LSODES	105	865	3116	271	2413	14236
PCR NOPRE	303	1849	6486	144	605	1890
PCR TCSSOR	335	2355	7609	255	1072	2764
PCR TCDKR	238	1412	4367	206	795	1951
POR K=1 NOPRE	329	1999	7035	151	639	2009
POR K=1 TCSSOR	190	1619	5200	154	672	1717
POR K=1 TCDKR	163	968	3091	135	509	1256
POR K=3 NOPRE	430	2719	10004	174	781	2545
POR K=3 TCSSOR	236	1921	6585	170	742	1927
POR K=3 TCDKR	177	1102	3624	141	541	1366
POR K=5 NOPRE	498	3308	12559	183	869	2902
POR K=5 TCSSOR	246	2119	7419	175	777	2026
POR K=5 TCDKR	183	1148	3885	143	563	1418

TABLE 5.2.2.2

The computational-work required by LSODES *and* LSODCG.V2 *to solve the linear algebraic systems throughout the numerical integration of the spatially-discretized* 2-D *and* 3-D *Convection-Diffusion Problem on a* $m \times m$ *and* $m \times m \times m$ *grid, respectively, measured in terms of the total number of multiplies,* MLTTOT, *divided by* 1,000 *and rounded to the nearest integer.*

Method	2-D Problem m			3-D Problem m		
	10	20	30	5	7	9
LSODES	105	865	3116	255	2401	14236
PCR NOPRE	*	2433	8475	213	839	2141
PCR TCSSOR	325	2559	7772	242	1063	2539
PCR TCDKR	234	1508	4265	205	756	1971
POR K=1 NOPRE	348	2656	10573	209	797	2427
POR K=1 TCSSOR	223	1670	5143	154	649	1746
POR K=1 TCDKR	155	924	2878	129	474	1206
POR K=3 NOPRE	448	3768	13349	237	831	3027
POR K=3 TCSSOR	229	1982	7308	165	730	1739
POR K=3 TCDKR	164	1044	3679	134	513	1304
POR K=5 NOPRE	544	4754	16730	255	935	3523
POR K=5 TCSSOR	238	2449	7675	169	754	1796
POR K=5 TCDKR	167	1100	3979	136	531	1368

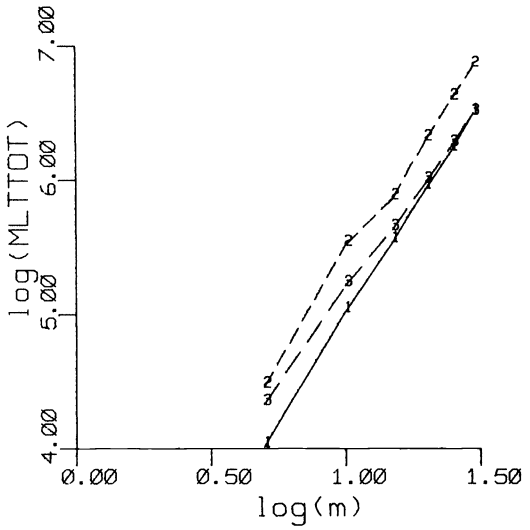
TABLE 5.2.2.3

Total storage, STRTOT, *required by the linear-equation solvers in* LSODES *and* LSODCG.V2 *for the* 2-D *and* 3-D *problems on a* $m \times m$ *and* $m \times m \times m$ *grid, respectively.*

Method	2-D Problem m			3-D Problem m		
	10	20	30	5	7	9
LSODES	4450	22606	59306	7909	32415	97425
PCR NOPRE	1422	5842	13262	2077	5931	12881
PCR TCDKR	1521	6241	14161	2201	6273	13609
POR K=1 NOPRE	1624	6644	15064	2329	6619	14341
POR K=1 TCDKR	1723	7043	15963	2453	6961	15069
POR K=3 NOPRE	2028	8248	18668	2833	7995	17261
POR K=3 TCDKR	2127	8647	19567	2957	8337	17989
POR K=5 NOPRE	2432	9852	22272	3337	9371	20181
POR K=5 TCDKR	2531	10251	23171	3461	9713	20909

grid ($m = 30$). However, the difference decreases with m and an extrapolation of the graphs in Plots 5.2.2.1 and 5.2.2.3 suggests that LSODCG.V2 with POR (1) preconditioned by TCDKR would become increasingly more efficient than LSODES for these two 2-D problems on finer grids. For the two 3-D problems with $m = 9$, LSODES requires more than ten times as many multiplies as LSODCG.V2 to solve the linear algebraic equations throughout the integration. Moreover, from Plots 5.2.2.2 and 5.2.2.4, it is clear that this factor grows with m.

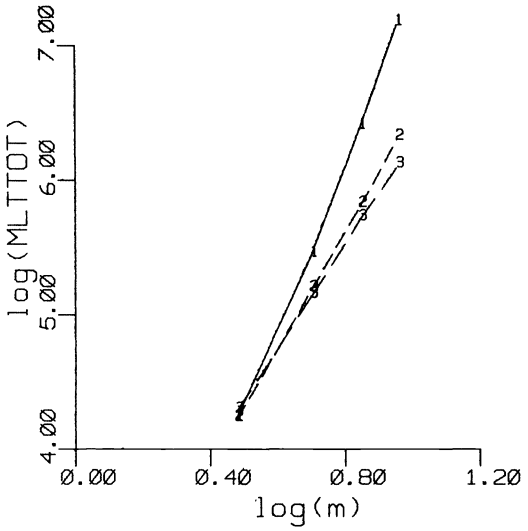
This supports and extends our earlier observation based on theoretical estimates of the computational-work in § 4 that iterative methods are significantly more efficient than direct methods for solving the spatially-discretized 3-D Heat Problem.



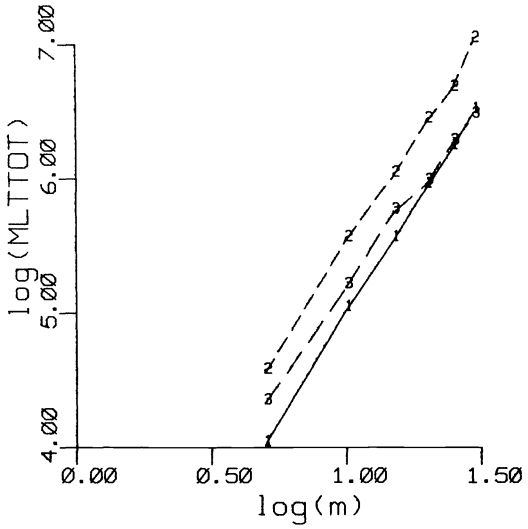
PLOT 5.2.2.1. *Graphs of m against* MLTTOT *for* (1) LSODES *and for* LSODCG.V2 *with* POR (1) *preconditioned by* (2) NOPRE *and by* (3) TCDKR *for the spatially-discretized* 2-D *Heat Problem.*

Table 5.2.2.3 and Plots 5.2.2.5 and 5.2.2.6 show that, for both the 2-D and 3-D problems, STRTOT is significantly larger for LSODES than LSODCG.V2: for the 2-D problems with $m = 30$, LSODES requires approximately 3.7 times as much storage as LSODCG.V2 and, for the 3-D problems with $m = 9$, LSODES requires approximately 6.4 times as much storage as LSODCG.V2. Moreover, for both the 2-D and 3-D problems, this factor grows with m.

5.3. Stiff Detest Problems. We used LSODE, LSODES, LSODCG.V1, and LSODCG.V2 on an IBM 3033 computer in double precision to solve the 30 Stiff Detest Problems [28], [30]. Although these problems are not large, they do test the robustness



PLOT 5.2.2.2. *Graphs of m against* MLTTOT *for* (1) LSODES *and for* LSODCG.V2 *with* POR (1) *preconditioned by* (2) NOPRE *and by* (3) TCDKR *for the spatially-discretized* 3-D *Heat Problem.*

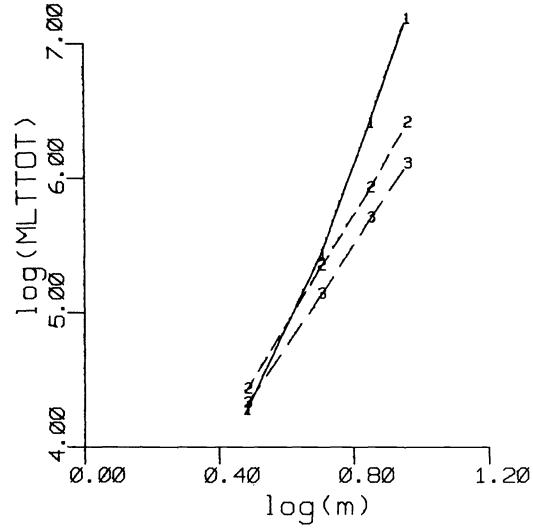


PLOT 5.2.2.3. *Graphs of m against* MLTTOT *for* (1) LSODES *and for* LSODCG.V2 *with* POR (1) *preconditioned by* (2) NOPRE *and by* (3) TCDKR *for the spatially-discretized* 2-D *Convection-Diffusion Problem.*

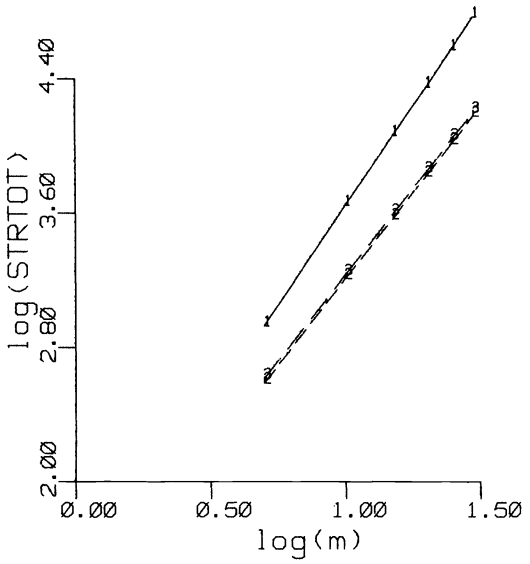
of the inexact chord-Newton method and the associated iterative linear-equation solvers in the two variants of LSODCG.

For each of the four codes, we solved the Stiff Detest Problems using the BDFs with exact Jacobians (MF = 21) to an absolute local error tolerance of ATOL = 10^{-2}, 10^{-4}, 10^{-6}, and 10^{-8} (RTOL = 0 and ITOL = 1).

For LSODCG.V1 and LSODCG.V2, we used POR (5), the PCGPACK [22], [26] implementation of Orthomin (5) [21], [25], to solve the linear algebraic systems of equations that arise in the inexact chord-Newton method. We did not precondition



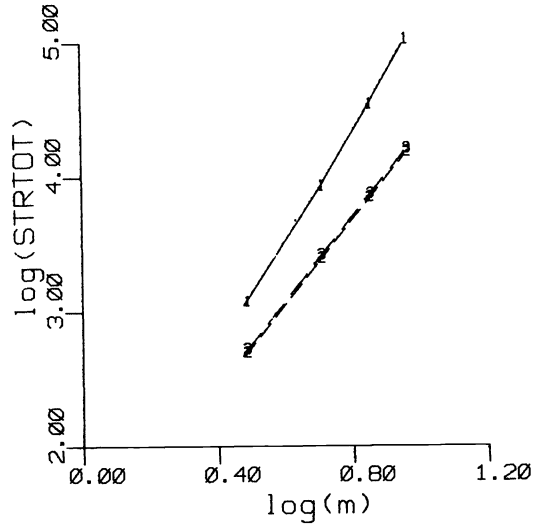
PLOT 5.2.2.4. *Graphs of m against* MLTTOT *for* (1) LSODES *and for* LSODCG.V2 *with* POR (1) *preconditioned by* (2) NOPRE *and by* (3) TCDKR *for the spatially-discretized* 3-D *Convection-Diffusion Problem.*



PLOT 5.2.2.5. *Graphs of m against* STRTOT *for* (1) LSODES *and for* LSODCG.V2 *with* POR (1) *preconditioned by* (2) NOPRE *and by* (3) TCDKR (*or* TCSSOR) *for the two spatially-discretized* 2-D *problems.*

POR (5) because, for many of the Stiff Detest Problems, an incomplete factorization would actually yield the exact factorization of the associated Newton iteration matrix and, consequently, POR (5) would generate the exact solution to the linear algebraic equations in one iteration.

We used a stopping criterion of the form (2.3.3) with $r = .1$, .25, and .5 for the solution of the linear algebraic systems arising in the inexact chord-Newton method. Since the Stiff Detest Problems are small and the tolerance for the linear algebraic systems is lax, we allowed a maximum of 50 POR (5) iterations to solve each linear algebraic system.



PLOT 5.2.2.6. *Graphs of m against* STRTOT *for* (1) LSODES *and for* LSODCG.V2 *with* POR (1) *preconditioned by* (2) NOPRE *and by* (3) TCDKR (*or* TCSSOR) *for the two spatially-discretized* 3-D *problems.*

TABLE 5.3.1
Normalized function evaluations for the Stiff Detest Problems.

Problem	Tol = 1.0D-2				Tol = 1.0D-4				Tol = 1.0D-6			
	LSODE	LSODCG r=.1	LSODCG r=.25	LSODCG r=.5	LSODE	LSODCG r=.1	LSODCG r=.25	LSODCG r=.5	LSODE	LSODCG r=.1	LSODCG r=.25	LSODCG r=.5
A1	53	44	59	71	115	112	129	141	233	218	231	289
A2 #	64	*	*	*	150	148	145	*	307	276	290	-
A3	86	98	94	108	194	207	206	243	399	372	401	464
A4	90	77	89	106	192	182	210	234	374	356	392	453
B1 #	544	651	1417	1691	1017	2151	2533	2747	-	-	-	-
B2	49	49	53	59	106	104	121	133	191	201	219	250
B3	53	49	62	65	121	117	128	156	228	225	267	304
B4	84	80	82	99	190	173	201	218	582	608	441	495
B5	24	1933	327	314	2878	1802	584	666	3502	3439	1430	1489
C1	88	60	64	69	189	141	157	168	271	271	284	418
C2	50	52	61	83	126	139	144	178	265	247	267	410
C3	48	47	50	71	132	124	143	155	254	242	298	378
C4 #	210	513	623	777	362	813	1061	1343	635	-	1524	-
C5 #	361	1270	1368	1541	639	1564	1769	2552	-	-	-	-
D1	97	98	110	110	218	197	218	237	481	375	380	447
D2	85	93	89	94	222	177	186	204	444	341	347	356
D3 #	77	*x	*x	*x	180	160	176	194	348	317	350	378
D4 #	9	22	14	14	17	30	20	21	41	48	39	50
D5	57	60	67	61	120	125	123	138	241	247	250	283
D6	18	25	29	29	46	56	54	60	111	106	98	102
E1 #	-12	138	49	133	41	120	181	184	112	156	217	229
E2	325	335	352	404	575	595	660	704	-	-	-	-
E3	84	101	92	99	244	204	196	205	499	342	365	352
E4 #	248	*x	*x	*x	508	433	530	660	977	890	-	-
E5 #	-	x	x	*x	14	x	x	*x	29	30	33	35
F1 #	305	364	393	449	611	702	730	1050	1156	1347	1289	2091
F2 #	24	35	32	44	66	83	80	76	145	159	161	156
F3	x	x	x	x	x	x	x	x	21	34	36	37
F4	-	-	-	-	-	-	-	-	x	x	x	x
F5 #	x	*	*	*	72	54	69	83	113	117	119	135

We present our results for LSODE and LSODCG.V2 only. As in the previous section, the results for LSODCG.V2 are generally better than those for LSODCG.V1, and the strategies used in LSODCG.V2 are closer to those in LSODE than those in LSODES.

In Tables 5.3.1 and 5.3.2, respectively, we present the "normalized" number of function evaluations and Jacobian evaluations required by LSODE and LSODCG.V2 with $r = .1$, .25, and 5 to solve each of the 30 Stiff Detest Problems to an absolute global error tolerance of Tol $= 10^{-2}$, 10^{-4}, and 10^{-6} at the end-point of the integration; in Table 5.3.3, we present the "normalized" total number of POR (5) iterations required by LSODCG.V2 throughout the integration. These normalized statistics were calculated by a new version of the Stiff Detest Program which, as described in [27], first performs

TABLE 5.3.2
Normalized Jacobian evaluations for the Stiff Detest Problems.

Problem	Tol = 1.0D-2				Tol = 1.0D-4				Tol = 1.0D-6			
	LSODE	LSODCG r=.1	LSODCG r=.25	LSODCG r=.5	LSODE	LSODCG r=.1	LSODCG r=.25	LSODCG r=.5	LSODE	LSODCG r=.1	LSODCG r=.25	LSODCG r=.5
A1	13	2	2	3	17	5	6	6	25	9	11	13
A2 #	14	*	*	*	22	7	6	*	32	12	13	-
A3	16	4	4	4	25	8	9	10	40	17	17	20
A4	20	3	4	5	28	8	9	10	37	16	18	20
B1 #	46	65	159	187	72	108	178	208	-	-	-	-
B2	10	2	2	2	15	4	5	6	18	9	9	11
B3	11	2	2	3	16	5	5	7	21	10	11	13
B4	12	3	4	4	19	7	8	9	36	27	20	22
B5	136	87	14	12	146	83	27	27	180	156	65	66
C1	13	3	3	3	19	7	7	7	26	12	13	18
C2	12	3	3	3	20	6	7	8	30	11	12	18
C3	11	3	3	3	21	6	6	7	24	11	13	16
C4 #	26	21	41	74	35	35	80	84	43	-	87	-
C5 #	36	124	153	178	56	133	152	155	-	-	-	-
D1	12	4	4	3	29	7	8	9	56	15	15	18
D2	13	6	5	5	29	7	7	7	46	14	15	14
D3 #	18	*x	*x	*x	28	6	8	8	40	12	15	17
D4 #	5	1	1	1	5	3	1	1	10	2	2	2
D5	14	2	3	4	19	5	5	6	30	10	10	10
D6	8	2	2	2	8	2	3	3	16	4	3	4
E1 #	-1	15	10	20	12	8	25	26	15	9	19	28
E2	28	12	13	15	38	25	27	29	-	-	-	-
E3	13	3	3	3	30	8	8	8	55	14	16	15
E4 #	33	*x	*x	x	52	19	23	27	72	40	-	-
E5 #	-	x	x	*x	5	x	x	*x	7	2	2	2
F1 #	45	19	21	24	68	29	32	44	115	54	52	84
F2 #	7	2	2	2	10	3	3	3	17	6	6	6
F3	x	x	x	x	x	x	x	x	10	2	2	2
F4	-	-	-	-	-	-	-	-	x	x	x	x
F5 #	x	*	*	*	18	4	4	4	22	5	5	6

TABLE 5.3.3
Normalized PCGPACK *iterations for the Stiff Detest Problems.*

Problem	Tol = 1.0D-2			Tol = 1.0D-4			Tol = 1.0D-6		
	LSODCG r=.1	LSODCG r=.25	LSODCG r=.5	LSODCG r=.1	LSODCG r=.25	LSODCG r=.5	LSODCG r=.1	LSODCG r=.25	LSODCG r=.5
A1	70	60	56	111	103	103	175	160	178
A2 #	*	*	*	322	307	*	511	491	-
A3	139	117	122	148	216	222	398	382	366
A4	307	318	322	607	609	632	1019	1043	1104
B1 #	1196	1521	1588	2214	2340	2281	-	-	-
B2	60	57	50	101	79	73	133	107	101
B3	82	74	67	134	107	98	184	161	137
B4	133	99	87	217	151	145	723	291	223
B5	2954	316	181	2309	318	271	3817	577	500
C1	100	85	70	172	145	139	257	231	240
C2	90	83	103	173	143	148	248	234	237
C3	84	78	81	162	147	154	257	252	266
C4 #	569	602	737	747	892	868	-	1025	-
C5 #	1199	1194	1197	1346	1266	1044	-	-	-
D1	142	147	139	243	240	239	386	355	368
D2	88	83	69	131	126	135	232	208	217
D3 #	*x	*x	*x	97	90	84	123	123	125
D4 #	29	11	11	37	17	17	41	33	40
D5	54	49	42	90	75	75	161	143	144
D6	34	37	29	66	56	62	113	101	101
E1 #	179	77	139	108	194	173	110	143	181
E2	111	64	41	90	35	22	-	-	-
E3	99	80	76	166	142	135	253	247	227
E4 #	*x	*x	x	589	544	595	1009	-	-
E5 #	x	x	*x	x	x	*x	47	42	39
F1 #	875	905	949	1658	1633	2050	2990	2761	3852
F2 #	23	18	18	39	29	27	66	48	44
F3	x	x	x	x	x	x	83	85	83
F4	-	-	-	-	-	-	x	x	x
F5 #	*	*	*	92	114	119	153	148	158

a least squares fit to

$$\sum_{i=1}^{\text{NTOL}} [\log(\text{global error}_i) - \log(C) - E \cdot \log(\text{ATOL}_i)]^2$$

for C and E, where, in this case, $\text{ATOL}_i = 10^{-2}$, 10^{-4}, 10^{-6}, and 10^{-8} and NTOL = 4. The Stiff Detest Program then performs a piecewise linear interpolation on the actual recorded values of the costs to solve the IVP at ATOL_i versus the corresponding expected global accuracy $\text{Tol} = C \cdot \text{ATOL}^E$ to arrive at the normalized costs for an absolute global error tolerance of Tol. (A consequence of this procedure is that the normalized function and Jacobian evaluations are negative for one problem.)

A "—", "*", or "x" may occur as an entry in place of a number in these tables. A "—" indicates that Stiff Detest could not calculate the normalized statistics for this problem and tolerance based upon the actual global errors incurred. A "*" indicates that the method being tested (LSODE or LSODCG.V2) could not solve the problem at that tolerance, and a "x" indicates that Stiff Detest could not solve the problem at that tolerance. In addition, the 12 problems marked with a "#" have a Jacobian that is not negative-real over some subinterval of the range of integration.

From the tables, we see that the number of function and Jacobian evaluations typically increases with r. For the Jacobian evaluations, the increase is generally not significant, but, for the function evaluations, the increase is frequently 10% or more from one value of r to the next. On the other hand, the number of POR (5) iterations typically decreases with r by a factor of 10% or more from one value of r to the next. Hence, if a POR (5) iteration is less expensive than a function evaluation, then, based upon these results, $r = .1$ would usually be the most cost effective of the three values

considered. On the other hand, if a POR (5) iteration is substantially more expensive than a function evaluation (as is the case for the problems in the previous subsection), then, based upon these results, $r = 5$ would usually be the most cost effective of the three values considered. Thus, the choice of r is dependent upon the class of IVPs solved.

Except for problem C5, which has a Jacobian that is not negative-real, LSODCG.V2, with each of the values of r considered, used fewer Jacobian evaluations than LSODE on all problems that were solved successfully by both codes. Moreover, for those IVPs having a negative-real Jacobian, the number of Jacobian evaluations required differs by a factor of 2 to 5. This superiority of LSODCG.V2 over LSODE is a result of the strategy used in LSODCG.V2 described above that permits it to update the scalar factor $h_n\beta_n$ in the Newton iteration matrix $I - h_n\beta_n J$ whenever $h_n\beta_n$ changes without re-evaluating the Jacobian, J. If we had also removed from LSODCG.V2 the requirement inherited from LSODE that the Jacobian be re-evaluated at least once every MSBP ($= 20$) steps, then LSODCG.V2 would have used even fewer Jacobian evaluations.

Now consider the function evaluations required by LSODE and LSODCG.V2 with $r = .1$ to solve the Stiff Detest Problems.

LSODCG.V2 failed to solve four of the Stiff Detest Problems (A2, D3, E4, F5) at $\text{Tol} = 10^{-2}$. Each of these problems has a Jacobian that is not negative-real over some subinterval of the range of integration. However, except for problem F5 at $\text{Tol} = 10^{-6}$, LSODCG.V2 required fewer function evaluations than LSODE for these problems at $\text{Tol} = 10^{-4}$ and 10^{-6}.

Of the remaining problems, LSODCG.V2 with $r = .1$ used substantially fewer function evaluations than LSODE for seven of the Stiff Detest Problems (A1, A4, C1, C3, D1, D2, E3). Again, this may be due to LSODCG.V2's updating the scalar factor $h_n\beta_n$ in the Newton iteration matrix whenever $h_n\beta_n$ changes resulting in a more accurate Newton iteration matrix and a more rapid convergence of the Newton iteration.

LSODE and LSODCG.V2 used approximately the same number of function evaluations on 11 of the Stiff Detest Problems (A3, B2, B3, B4, B5, C2, D5, D6, F2, F3, F5). It is worth noting that the class B problems are of the form $\dot{y} = Ay$, where A is a constant matrix with complex eigenvalues and, consequently, A is far from being symmetric.

LSODE used substantially fewer function evaluations than LSODCG.V2 on seven problems (B1, C4, C5, D4, E1, F1, F2) each of which has a Jacobian that is not negative-real over some subinterval of the range of integration.

Therefore, except for those problems having a Jacobian that is not negative-real, the use of an iterative linear-equation solver did not cause the performance of LSODCG.V2 to deteriorate relative to the unmodified code LSODE which incorporates a direct linear-equation solver. In fact, LSODCG.V2 performs as well as or better than LSODE on all the Stiff Detest Problems for which LSODCG.V2 is applicable.

One final point is worth noting. From Tables 5.3.1 and 5.3.3, we see that, for many of the Stiff Detest Problems, particularly at the more stringent tolerances, an average of less than one PCGPACK iteration is required per inexact chord-Newton iteration. That is, for many of the inexact chord-Newton iterations, the initial guess $-F(y_n^k)$ for $y_n^{k+1} - y_n^k$ satisfies (2.3.3) and no further PCGPACK iterations are required. Hence, when using an iterative linear-equation solver in a stiff-ODE code in this way, we automatically obtain the benefit of the use of an inexpensive predictor-corrector iteration when a more expensive Newton iteration is not required. Moreover, this appears to have no deleterious effect upon the overall performance of the stiff-ODE solver.

6. Conclusions. Both the theoretical and numerical results presented in the preceding two sections show that the use of iterative linear-equation solvers in stiff-ODE codes has the potential to improve the efficiency—in terms of both computational-work and storage—with which a significant class of stiff IVPs having large sparse Jacobians can be solved. Moreover, these results demonstrate the importance of preconditioning for Krylov subspace methods used in stiff-ODE solvers.

The numerical results for both the linear and nonlinear IVPs show that the stopping criterion (2.3.3) for the inexact chord-Newton iteration works well in practice for $r \in [.1, .5]$. This supports the claim that the linear equations that arise in stiff-ODE solvers need not be solved very accurately. Moreover, the initial guess $-F(y_n^k)$ for the solution $y_n^{k+1} - y_n^k$ of the linear system proved to be quite effective in practice, particularly during the initial transient where the IVP is at most mildly-stiff.

Updating the scalar factor $h_n\beta_n$ in the Newton iteration matrix $I - h_n\beta_n J$ whenever $h_n\beta_n$ changes without re-evaluating J, the approximation to the Jacobian, reduces the number of Newton iterations and associated function evaluations required throughout the course of the numerical integration with little added cost in a stiff-ODE code incorporating an iterative linear-equation solver. Furthermore, this strategy of updating $h_n\beta_n$ whenever it changes facilitates the decision when to re-evaluate the Jacobian and, thus, helps to avoid wasted computational-work. More generally, as mentioned in § 2.2, the removal of the constraint imposed by the necessity to avoid refactoring $I - h_n\beta_n J$ in a stiff-ODE code employing a direct linear-equation solver may lead to other benefits in the choice of formulas, strategies, and heuristics for a stiff-ODE code incorporating an iterative linear-equation solver.

Most importantly, the numerical results demonstrate that stiff-ODE codes incorporating iterative linear-equation solvers do not suffer a loss of robustness on those IVPs for which the Newton iteration matrix W_n^k is positive-real throughout the course of the numerical integration. Note, though, that this restriction on W_n^k is imposed by the iterative technique we chose to solve the linear systems: the restriction is not characteristic of all stiff-ODE codes incorporating iterative linear-equation solvers. In particular, as mentioned earlier, there exist iterative linear-equation solvers that are guaranteed to converge to the solution of the Newton system (2.2.1) if all the eigenvalues of W_n^k lie in the right-half complex plane. As we argued in § 2.1, if W_n^k does not satisfy this last restriction, then the stepsize is almost surely too large and should be reduced until this last restriction is satisfied to ensure a reliable numerical integration.

Hence, it appears possible to develop a stiff-ODE code incorporating an iterative linear-equation solver that, for a broad class of IVPs, is as robust as a similar stiff-ODE code incorporating a direct linear-equation solver, but more efficient than the latter code for a significant subclass of problems having large sparse Jacobians. We plan to continue to pursue this investigation in the future.

Acknowledgment. We thank the editor, Dianne O'Leary, for several helpful suggestions particularly with respect to the presentation of our numerical results.

REFERENCES

[1] O. AXELSSON, *Solution of linear systems of equations: iterative methods*, in Sparse Matrix Techniques, V. A. Barker, ed, Lecture Notes in Mathematics 572, Springer-Verlag, New York, 1977.

[2] J. H. BRAMBLE, *Multistep methods for quaslinear parabolic equations*, in Computational Methods in Nonlinear Mechanics, J. T. Oden, ed, North-Holland, Amsterdam, 1980, pp. 177-183.

[3] P. N. BROWN AND A. C. HINDMARSH, *Matrix-free methods in the solution of stiff systems of* ODEs, Tech. Rep. UCID-19937, Lawrence Livermore Laboratory, Univ. California, Livermore, CA, 1972.

[4] J. C. BUTCHER, *Implicit Runge-Kutta processes*, Math. Comp., 18 (1964), pp. 50-64.

[5] G. D. BYRNE AND A. C. HINDMARSH, *A polyalgorithm for the numerical solution of ordinary differential equations*, ACM Trans. Math. Software, 1 (1975), pp. 71-96.

[6] G. D. BYRNE, A. C. HINDMARSH, K. R. JACKSON AND H. G. BROWN, *Comparative test results for two* ODE *solvers*—EPISODE *and* GEAR, Tech. Rep. ANL-77-19, Argonne National Laboratory, Argonne, IL, 1977.

[7] ———, *A comparison of two* ODE *codes*: GEAR *and* EPISODE, Computers and Chemical Eng., 1 (1977), pp. 133-147.

[8] T. F. CHAN AND K. R. JACKSON, *Nonlinearly preconditioned Krylov subspace methods for discrete Newton algorithms*, this Journal, 5 (1984), pp. 533-542.

[9] ———, *The use of iterative linear-equation solvers in codes for large systems of stiff* IVPs *for* ODEs, Tech. Rep. 170/84, Dept. Computer Science, Univ. Toronto, Toronto, Canada, 1984; also appeared as Research Report YALEU/DCS/RR-312, Dept. Computer Science, Yale Univ., New Haven, CT.

[10] T. F. CHAN, K. R. JACKSON AND B. ZHU, *Alternating-direction incomplete factorizations*, SIAM J. Numer. Anal., 20 (1983), pp. 239-257.

[11] R. CHANDRA, *Conjugate gradient methods for partial differential equations*, Ph.D. thesis, Tech. Rep. 129, Dept. Computer Science, Yale Univ., New Haven, CT, 1978.

[12] R. CHANDRA, S. C. EISENSTAT AND M. H. SCHULTZ, *The modified conjugate residual method for partial differential equations*, Tech. Rep. 107, Dept. Computer Science, Yale Univ., New Haven, CT, 1977.

[13] P. CONCUS AND G. H. GOLUB, *A generalized conjugate gradient method for nonsymmetric systems of linear equations*, Proc. Second International Symposium on Computing Methods in Applied Sciences and Engineering, R. Glowinski and J. L. Lyons, eds., Lecture Notes in Economics and Mathematical Systems, 134, Springer-Verlag, New York, 1976.

[14] P. CONCUS, G. H. GOLUB AND D. P. O'LEARY, *A generalized conjugate gradient method for the numerical solution of elliptic partial differential equations*, in Sparse Matrix Computations, J. R. Bunch and D. J. Rose, eds., Academic Press, New York, 1976, pp. 309-332.

[15] ———, *Numerical solution of nonlinear elliptic partial differential equations by a generalized conjugate gradient method*, Tech. Rep. STAN-CS-76-585, Dept. Computer Science, Stanford Univ., Stanford, CA, 1976.

[16] R. S. DEMBO, S. C. EISENSTAT AND T. STEIHAUG, *Inexact Newton methods*, SIAM J. Numer. Anal., 19 (1982), pp. 400-408.

[17] J. DOUGLAS JR. AND T. DUPONT, *Preconditioned conjugate gradient iteration applied to Galerkin methods for a mildly nonlinear Dirichlet problem*, in Sparse Matrix Computations, J. R. Bunch and D. J. Rose, eds., Academic Press, New York, 1976, pp. 333-348.

[18] J. DOUGLAS JR., T. DUPONT AND R. E. EWING, *Incomplete iteration for time-stepping a Galerkin method for a quasilinear parabolic problem*, SIAM J. Numer. Anal., 16 (1979), pp. 503-522.

[19] T. DUPONT, R. O. KENDALL AND H. H. RACHFORD, *An approximate factorization procedure for solving self-adjoint elliptic difference equations*, SIAM J. Numer. Anal., 6 (1968), pp. 753-782.

[20] S. C. EISENSTAT, *Efficient implementation of a class of preconditioned conjugate gradient methods*, this Journal, 2 (1981), pp. 1-4.

[21] S. C. EISENSTAT, H. C. ELMAN AND M. H. SCHULTZ, *Variational iterative methods for nonsymmetric systems of linear equations*, SIAM J. Numer. Anal., 20 (1983), pp. 345-357.

[22] S. C. EISENSTAT, H. C. ELMAN, M. H. SCHULTZ AND A. H. SHERMAN, *The (new) Yale sparse matrix package*, Tech. Rep. YALEU/DCS/RR-265, Dept. Computer Science, Yale Univ., New Haven, CT, 1977.

[23] S. C. EISENSTAT, M. C. GURSKY, M. H. SCHULTZ AND A. H. SHERMAN, *Yale sparse matrix package* I: *The symmetric codes*, Tech. Rep. 112, Dept. Computer Science, Yale Univ., New Haven, CT, 1977.

[24] ———, *Yale sparse matrix package* II: *The nonsymmetric codes*, Tech. Rep. 114, Dept. Computer Science, Yale Univ., New Haven, CT, 1977.

[25] H. C. ELMAN, *Iterative methods for large, sparse, nonsymmetric systems of linear equations*, Ph.D. thesis, Tech. Rep. 229, Dept. Computer Science, Yale Univ., New Haven, CT, 1982.

[26] ———, PCGPACK, *a collection of preconditioned Krylov subspace methods for the solution of systems of linear algebraic equations*, private communication.

[27] W. H. ENRIGHT, *Using a testing package for the automatic assessment of numerical methods for* ODEs, in Performance Evaluation of Numerical Software, L. Fosdick, ed., North-Holland, Amsterdam, 1979, pp. 199-213.

[28] W. H. ENRIGHT AND T. E. HULL, *Comparing numerical methods for the solution of stiff systems of* ODE*s arising in chemistry*, in Methods for Differential Systems, L. Lapidus and W. E. Schiesser, eds., Academic Press, New York, 1976, pp. 45-65.

[29] ———, *Test results in initial value methods for non-stiff ordinary differential equations*, SIAM J. Numer. Anal., 13 (1976), pp. 944–961.

[30] W. H. ENRIGHT, T. E. HULL AND B. LINDBERG, *Comparing numerical methods for stiff systems of* ODEs, BIT, 15 (1975), pp. 10–48.

[31] W. H. ENRIGHT AND M. S. KAMEL, *Automatic partitioning of stiff systems and exploiting the resulting structure*, ACM Trans. Math. Software, 5 (1979), pp. 374–385.

[32] B. S. GARBOW, J. M. BOYLE, J. J. DONGARRA AND C. B. MOLER, *Matrix Eigensystem Routines—* EISPACK *Guide Extension*, Lecture Notes in Computer Science 51, Springer-Verlag, New York, 1977.

[33] N. K. GARG AND R. A. TAPIA, QDN: *a variable storage algorithm for unconstrained optimization*, Tech. Rep., Dept. Mathematical Sciences, Rice Univ., Houston, TX, 1980.

[34] C. W. GEAR, *Algorithm* 407, DIFSUB *for the solution of ordinary differential equations*, Comm. ACM, 14 (1971), pp. 185–190.

[35] ———, *Numerical Initial Value Problems in Ordinary Differential Equations*, Prentice-Hall, Englewood Cliffs, NJ, 1971.

[36] ———, *Numerical solution of ordinary differential equations: Is there anything left to do?* SIAM Rev., 23 (1981), pp. 10–24.

[37] ———, *Stiff software: What do we have and what do we need?* Tech. Rep. UIUCDCS-R-82-1109, Dept. Computer Science, Univ. Illinois, Urbana, IL, 1982.

[38] C. W. GEAR AND Y. SAAD, *Iterative solution of linear equations in* ODE *codes*, this Journal, 4 (1983), pp. 583–601.

[39] C. W. GEAR, K. W. TU AND D. S. WATANABE, *The stability of automatic programs for numerical problems*, in Differential Systems, R. A. Willoughby, ed., Plenum Press, New York, 1974, pp. 111–121.

[40] W. GIVENS, *Fields of values of a matrix*, Proc. American Math. Soc., 3 (1952), pp. 206–209.

[41] G. H. GOLUB AND R. S. VARGA, *Chebyshev semi-iterative methods, successive over relaxation iterative methods, and second order Richardson iterative methods*, Numer. Math., 3 (1961), pp. 147–168.

[42] I. GUSTAFSSON, *A class of first order factorization methods*, BIT, 18 (1978), pp. 142–156.

[43] L. A. HAGEMAN AND D. M. YOUNG, *Applied Iterative Methods*, Academic Press, New York, 1981.

[44] M. R. HESTENES AND E. STIEFEL, *Methods of conjugate gradient for solving linear systems*, J. Research National Bureau of Standards, 49 (1952), pp. 409–436.

[45] A. C. HINDMARSH, GEAR: *ordinary differential equation system solver*, Tech. Rep. UCID-30001, Rev. 2, Lawrence Livermore Laboratory, Univ. California, Livermore, CA, 1972.

[46] ———, GEARB: *solution of ordinary differential equations having banded Jacabian*, Tech. Rep. UCID-30059, Lawrence Livermore Laboratory, Univ. California, Livermore, CA, 1973.

[47] ———, *Preliminary report of* GEARIB: *solution of implicit systems of ordinary differential equations with banded Jacobian*, Tech. Rep. UCID-30130, Lawrence Livermore Laboratory, Univ. California, Livermore, CA, 1976.

[48] ———, *Preliminary documentation of* GEARBI: *solution of* ODE *systems with block iterative treatment of the Jacobian*, Tech. Rep. UCID-30149, Lawrence Livermore Laboratory, Univ. California, Livermore, CA, 1976.

[49] ———, LSODE *and* LSODI, *two new initial value ordinary differential equation solvers*, ACM SIGNUM Newsletter, 15 (1980), pp. 10–11.

[50] A. C. HINDMARSH AND A. H. SHERMAN, LSODES, private communication, 1982.

[51] N. HOUBAK, S. P. NORSETT AND P. G. THOMSEN, *Displacement or residual test in the application of implicit methods for stiff problems*, manuscript.

[52] P. J. VAN DER HOUWEN AND B. P. SOMMEIJER, *Predictor-corrector methods with improved absolute stability regions*, Tech. Rep., Mathematisch Centrum, Amsterdam, 1982.

[53] K. R. JACKSON, *Variable stepsize, variable order integrand approximation methods for the numerical solution of ordinary differential equations*, Ph.D. thesis, Tech. Rep. 129, Dept. Computer Science, Univ. Toronto, Toronto, Canada, 1978.

[54] K. R. JACKSON AND R. SACKS-DAVIS, *An alternative implementation of variable stepsize multistep formulas for stiff* ODEs, ACM Trans. Math. Software, 6 (1980), pp. 295–318.

[55] K. C. JEA, *Generalized conjugate gradient acceleration of iterative methods*, Ph.D. thesis, Tech. Rep. CNA-176, Center for Numerical Analysis, Univ. Texas, Austin, TX, 1982.

[56] O. G. JOHNSON, C. A. MICCHELLI AND G. PAUL, *Polynomial preconditionings for conjugate gradient calculations*, SIAM J. Numer. Anal., 20 (1983), pp. 362–376.

[57] D. S. KERSHAW, *The incomplete Cholesky-conjugate gradient method for the iterative solution of systems of linear equations*, J. Comput. Physics, 26 (1978), pp. 43–65.

[58] ———, *On the problem of unstable pivots in the incomplete LU-conjugate gradient method*, J. Comput. Physics, 31 (1980), pp. 114–123.

[59] D. R. KINCAID, J. R. RESPESS, D. M. YOUNG AND R. G. GRIMES, *Algorithm* 586. ITPACK 2C: *a* FORTRAN *package for solving large sparse linear systems by adaptive accelerated iterative methods*, ACM Trans. Math. Software, 8 (1982), pp. 302-322.

[60] F. T. KROGH AND K. STEWART, *Asymptotic* ($h \to \infty$) *absolute stability for* BDF*'s applied to stiff differential equations*, Computing Memorandum 478, Jet Propulsion Laboratory, California Institute of Technology, Pasedena, CA, (revised) 1983.

[61] J. D. LAMBERT, *Computational Methods in Ordinary Differential Equations*, John Wiley, New York, 1976.

[62] T. A. MANTEUFFEL, *The Tchebychev iteration for nonsymmetric linear systems*, Numer. Math., 28 (1977), pp. 307-327.

[63] ———, *Adaptive procedure for estimating parameters of the nonsymmetric Tchebychev iteration*, Numer. Math., 31 (1978), pp. 183-208.

[64] ———, *An incomplete factorization technique for positive definite linear systems*, Math. Comp., 34 (1980), pp. 473-497.

[65] J. A. MEIJERINK AND H. A. VAN DER VORST, *An iterative solution method for linear systems of which the coefficient matrix is a symmetric M-matrix*, Math. Comp., 31 (1977), pp. 148-162.

[66] W. L. MIRANKER AND I-L. CHERN, *Dichotomy and conjugate gradients in the stiff initial value problem*, Tech. Rep. RC 8032, IBM Thomas J. Watson Research Center, Yorktown Heights, NY, 1980.

[67] A. NORDSIECK, *On the numerical integration of ordinary differential equations*, Math. Comp., 16 (1962), pp. 22-49.

[68] D. P. O'LEARY, *A discrete Newton algorithm for minimizing a function of many variables*, Math. Progr., 23 (1982), pp. 20-33.

[69] J. M. ORTEGA AND W. C. RHEINBOLDT, *Iterative Solution of Nonlinear Equations in Several Variables*, Academic Press, New York, 1960.

[70] J. K. REID, *On the method of conjugate gradients for the solution of large sparse systems of linear equations*, in Large Sparse Sets of Linear Equations, J. K. Reid, ed., Academic Press, New York, 1971, pp. 231-254.

[71] Y. SAAD, *Krylov subspace methods for solving large unsymmetric linear systems*, Math. Comp., 37 (1981), pp. 105-126.

[72] ———, *The Lanczos biorthogonalization algorithm and other oblique projection methods for solving large unsymmetric systems*, SIAM J. Numer. Anal., 19 (1982), pp. 485-506.

[73] ———, *Practical use of polynomial preconditioning for the conjugate gradient method*, Res. Rep. YALEU/DCS/RR-282, Dept. Computer Science, Yale Univ., New Haven, CT, 1983.

[74] M. A. SAUNDERS, *Sparse least squares by conjugate gradients: a comparison of preconditioning methods*, Tech. Rep. SOL 79-5 Systems Optimization Laboratory, Stanford Univ., Stanford, CA, 1979.

[75] H. R. SCHWARZ, *The method of conjugate gradients in finite element applications*, J. Appl. Math. Phys., 30 (1979), pp. 342-353.

[76] L. F. SHAMPINE, *Evaluation of implicit formulas for the solution of* ODEs, Tech. Rep. SAND79-1370, Sandia Laboratories, Albuquerque, NM, 1979.

[77] ———, *Implementation of implicit formulas for the solution of* ODEs, this Journal, 1 (1980), pp. 103-118.

[78] L. F. SHAMPINE, H. A. WATTS AND S. M. DAVENPORT, *Solving non-stiff ordinary differential equations—the state of the art*, SIAM Rev., 18 (1975), pp. 376-411.

[79] A. H. SHERMAN, *On Newton-iterative methods for the solution of systems of nonlinear equations*, SIAM J. Numer. Anal., 15 (1978), pp. 755-771.

[80] A. H. SHERMAN AND A. C. HINDMARSH, GEARS: *a package for the solution of sparse, stiff ordinary differential equations*, in Electrical Power Problems: The Mathematical Challenge, A. Erisman, K. Neves and M. H. Dwarakanath, eds., Society for Industrial and Applied Mathematics, Philadelphia, 1981, pp. 190-200.

[81] B. T. SMITH, J. M. BOYLE, J. J. DONGARRA, B. S. GARBOW, Y. IKEBE, V. C. KLEMA AND C. B. MOLER, *Matrix Eigensystem Routines*—EISPACK *Guide*, Lecture Notes in Computer Science 6, Springer-Verlag, New York, 1976.

[82] H. J. STETTER, *Stability of discretizations on infinite intervals*, in Conference on Applications of Numerical Analysis, Dundee, Scotland, J. L. Morris, ed., Lecture Notes in Mathematics 228, Springer-Verlag, New York, 1971, pp. 207-222.

[83] R. S. VARGA, *Matrix Iterative Analysis*, Prentice-Hall, New York, 1962.

[84] P. K. W. VINSOME, *Orthomin, an iterative method for solving sparse sets of simultaneous linear equations*, Paper SPE 5729, Proc. Fourth Symposium on Reservoir Simulation, Society of Petroleum Engineers of AIME, 1976, pp. 149-159.

[85] D. S. WATKINS AND R. W. HANSON-SMITH, *The numerical solution of separably stiff systems by precise partitioning*, ACM Trans. Math. Software, 9 (1983), pp. 293-301.

[86] O. WIDLUND, *A Lanczos method for a class of nonsymmetric sytems of linear equations*, SIAM J. Numer. Anal., 15 (1978), pp. 801–812.

[87] J. WILLIAMS, *The problem of implicit formulas in numerical methods for stiff differential equations*, Tech. Rep. 40, Dept. Mathematics, Univ. Manchester, Manchester, England, 1979.

[88] D. M. YOUNG, *Iterative Solution of Large Linear Systems*, Academic Press, New York, 1971.

[89] D. M. YOUNG AND K. C. JEA, *Generalized conjugate gradient acceleration of nonsymmetrizable iterative methods*, Linear Algebra and Appl., 34 (1980), pp. 159–194.

SIAM J. SCI. STAT. COMPUT.
Vol. 7, No. 2, April 1986

AN ADAPTIVE SHOOTING METHOD FOR SINGULARLY PERTURBED BOUNDARY VALUE PROBLEMS*

MAXIMILIAN R. MAIER†

Abstract. An algorithm based on multiple shooting is given for the numerical solution of singularly perturbed boundary value problems for the system $y' = f(x, y)$ with possible boundary layers at the endpoints. The algorithm does not require any explicit a priori identification of any small parameters such as those that typically multiply derivatives in problems of singular perturbation type. The algorithm is shown to enjoy a certain robustness as illustrated by various examples including several nontrivial problems from the physical theory of semiconducting devices. One sees that the algorithm is both efficient and accurate for all examples considered.

Key words. shooting methods, two point boundary value problems, singular perturbations, boundary layers, semiconductor devices

1. Introduction. A singular perturbation problem is a problem that depends on a parameter (or parameters) in such a way that the solutions of the problem behave nonuniformly as the parameter tends towards some limiting value of interest. Generally the solutions have a multiscale character, i.e. there are thin boundary layer regions where the solution varies rapidly, while outside these boundary layers the solution behaves regularly. There is a large literature on the theoretical study of singularly perturbed boundary value problems, of which only O'Malley [22], Howes [16], and Smith [30] are mentioned here. The solutions of such problems depend quite sensitively on perturbations in the data including numerical perturbations due to roundoff errors, and for this reason the numerical solution of such problems has presented severe difficulties. From the large numerical literature, mention is made here only of the works of Miranker [20] and Flaherty, O'Malley [15] based on singular perturbation theory, the works of Abrahamson, Keller, Kreiss [2], Kreiss [18], and Pearson [24], [25] based on finite difference methods, and the work of Flaherty, Mathon [14] based on collocation methods.

Multiple shooting is an efficient and accurate method for solving boundary value problems (cf. Stoer and Bulirsch [31]). In the present paper an algorithm based on multiple shooting is given for the numerical solution of singularly perturbed boundary value problems for the system $y' = f(x, y)$ on an interval $[a, b]$. The algorithm automatically decomposes the interval into an *outer interval* where the solution behaves regularly, and one or two *boundary layer intervals*, each adjoining an endpoint a or b, where the solution may undergo rapid variation. Interior layers are not considered by the algorithm in its present form, although turning points are permitted within a boundary layer, as illustrated by several of the examples in § 4. Multiple shooting replaces the given boundary value problem with a certain *collection* of initial value problems, one of which lives within the outer interval, while the others live within a boundary layer interval. The multiscale character of solutions of such problems is a reflection of the fact that the Jacobian matrix of the right side of the differential equation has some eigenvalues with absolute real parts that are *large* compared with other eigenvalues. The automatic decomposition of the given interval $[a, b]$ into the outer and boundary layer subintervals is based on a certain numerically computed measure of the sizes of such eigenvalues of large absolute real parts.

* Received by the editors February 14, 1984, and in revised form November 13, 1984.

† Mathematisches Institut der Technischen Universität München, Postfach 20 2420, D-8000 Munchen, West Germany.

The eigenvalues with large absolute real parts produce very small stepsizes during the numerical solution of the *initial value problems* in multiple shooting, and this is ordinarily the case here also for the initial value problem living within the outer interval where the exact solution is regular and where one should wish to be permitted to use large stepsizes. To overcome this difficulty, the initial value problem living in the outer interval is solved with a special (implicit) Runge-Kutta method possessing a suitable stability property which permits large stepsizes. The remaining initial value problems that live within a boundary layer interval are solved with a different, high order explicit Runge-Kutta method. Multiple shooting connects the various initial value problems in a suitable manner, yielding a certain nonlinear algebraic system which is solved in the present algorithm using a modified Newton method in conjunction with Householder transformations. (Householder transformations are used in a different way in O'Malley, Flaherty [23] in the study of certain singularly perturbed initial value problems.) The entire algorithm is iterative, and nodes (subintervals) are removed and/or inserted automatically as needed, before each new (global) iterative cycle. Hence the subdivision of $[a, b]$ into outer and boundary layer intervals is generally modified somewhat during the course of the calculation until a convergence criterion has been achieved.

The multiple shooting procedure is introduced in § 2, where the subdivision of $[a, b]$ into the outer and boundary layer intervals is also discussed along with the general method of selection of nodes throughout $[a, b]$. Also discussed in § 2 is the solution method for the nonlinear algebraic system appearing in the multiple shooting. The nonlinear system is particularly troublesome for the singularly perturbed problems considered here. The successful application of multiple shooting for singularly perturbed boundary value problems is critically dependent on the use in the outer interval of an initial value integrator that is suitably stable so that one is not restricted there to unnecessarily small stepsizes. This point is discussed in § 3 where an appropriate *outer* initial value integrator is described.

The performance of the entire algorithm is illustrated in § 4 with several examples including certain nontrivial problems arising in the physical theory of semiconducting devices. Several of these examples would perhaps seem *not* to be ideally suited for the algorithm. Specifically, the bipolar transistor of § 4.3 has "large" eigenvalues that are actually rather small, whereas the problem of § 4.4 on the electrical properties of electron-irradiated silicon involves a turning point inside a boundary layer, as does the problem of § 4.5. It is seen that the algorithm has a certain robustness in the sense that it proves to be accurate and efficient for all examples considered.

2. Multiple shooting for singularly perturbed boundary value problems. Multiple shooting is a widely used method for the accurate and efficient numerical solution of two-point boundary value problems of the form

$$\begin{aligned} &y'(x) = f(x, y(x)), \qquad x \in [a, b], \\ &r(y(a), y(b)) = 0 \end{aligned} \tag{2.1}$$

with $a, b \in \mathbb{R}$, $y = (y_1, y_2, \cdots, y_n)^t$, $y_i : [a, b] \to \mathbb{R}$, $f = (f_1, \cdots, f_n)^t$, $f_i : [a, b] \times \mathbb{R}^n \to \mathbb{R}$, $r = (r_1, \cdots, r_n)^t$, $r_i : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$. The method is described e.g. in Bulirsch [6] and Bulirsch, Stoer and Deuflhard [7]. The solution of (2.1) is computed by solving a sequence of initial value problems.

The interval $[a, b]$ is first subdivided into $m-1$ subintervals as described near the end of this section, where in the present case this subdivision involves a greater number

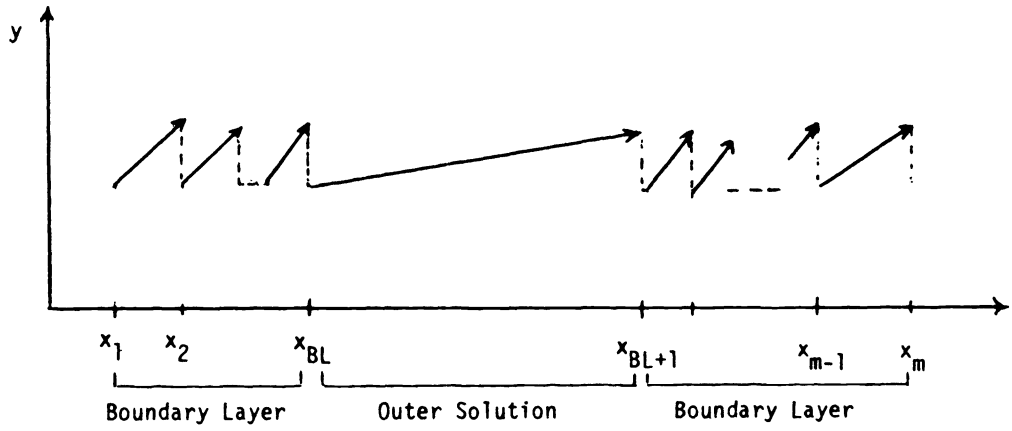


FIG. 2.2

of nodes in the boundary layers where the solution varies rapidly, as illustrated in Fig. 2.2.

At the nodes x_j, n-dimensional starting values s_j are prescribed, for $j = 1, \cdots, m$. If $y(x, x_j, s_j)$ denotes the solution of the following initial value problem

$$(2.3)_j \qquad \begin{aligned} &y' = f(x, y) \quad \text{for } x \in [x_j, x_{j+1}), \\ &y(x_j) = s_j \quad \text{for } j = 1, \cdots, m-1, \end{aligned}$$

then s_j must be determined so that the piecewise composed function

$$(2.4) \qquad \begin{aligned} &y(x) = y(x, x_j, s_j) \quad \text{for } x \in [x_j, x_{j+1}), \quad j = 1, \cdots, m-1, \\ &y(b) = s_m \end{aligned}$$

is continuous and satisfies the boundary conditions. This yields the generally nonlinear system

$$(2.5) \qquad F(s) \equiv \begin{bmatrix} F_1(s_1, s_2) \\ F_2(s_2, s_3) \\ \vdots \\ F_{m-2}(s_{m-2}, s_{m-1}) \\ F_{m-1}(s_1, s_{m-1}) \end{bmatrix} := \begin{bmatrix} y(x_2, x_1, s_1) - s_2 \\ y(x_3, x_2, s_2) - s_3 \\ \vdots \\ y(x_{m-1}, x_{m-2}, s_{m-2}) - s_{m-1} \\ r(s_1, y(x_m, x_{m-1}, s_{m-1})) \end{bmatrix} = 0$$

where $s = (s_1, \cdots, s_{m-1})^t$.

The initial value problem $(2.3)_j$ is solved numerically with an integration method that depends on whether the subinterval $[x_j, x_{j+1})$ lies in a boundary layer or not. The specially chosen, stable integrator of § 3 is used if this subinterval is in the outer interval, whereas a high-order method is used if this subinterval is in a boundary layer.

The nonlinear system (2.5) is solved iteratively with a modified Newton method as (cf. Stoer and Bulirsch [31])

$$(2.6) \qquad s^{(k+1)} = s^{(k)} - \lambda_k DF(s^{(k)})^{-1} F(s^{(k)}), \qquad k = 0, 1, 2, \cdots$$

where as usual this linear system (2.6) is solved in the form

$$(2.7) \qquad DF(s^{(k)}) \Delta s = -\lambda_k F(s^{(k)}).$$

In the present case the correction $\Delta s := s^{(k+1)} - s^{(k)}$ is written as $\Delta s = (\Delta s_1, \Delta s_2, \cdots,$

$\Delta s_{BL}, \Delta s_{BL+1}, \cdots, \Delta s_{m-2}, \Delta s_{m-1})^t$, and similarly the vector $F = F(s^{(k)})$ is written as $F = (F_1, F_2, \cdots, F_{BL}, F_{BL+1}, \cdots, F_{m-2}, F_{m-1})^t$, where the index "$BL$" is used to label components associated with the particular node x_{BL} which separates the left boundary layer interval from the outer interval in the current iteration. (If, as is sometimes the case, the multiple shooting is done instead from right to left, then one chooses x_{BL} in relation to the right boundary layer.) The matrix of the linear system (2.7) has the block form

$$DF(s) = \begin{bmatrix} G_1 & -I & & & & & & & \\ & G_2 & -I & & & & & & \\ & & \ddots & \ddots & & & & & \\ & & & G_{BL} & -I & & & & \\ & & & & G_{BL+1} & -I & & & \\ & & & & & \ddots & \ddots & & \\ & & & & & & G_{m-2} & -I & \\ A & & & & & & & & BG_{m-1} \end{bmatrix} \tag{2.8}$$

where $G_j := \partial F_j(s_j, s_{j+1})/\partial s_j$ (for $j = 1, 2, \cdots, m-1$) and $A := \partial F_{m-1}(s_1, s_{m-1})/\partial s_1$, $B := \partial F_{m-1}(s_1, s_{m-1})/\partial s_{m-1}$. The matrices G_j, A, and B are computed by numerical differentiation. The evaluation of $DF(s)$ is not necessary in every iteration step. The amount of work is reduced substantially by updating the matrices G_j, A, B in certain special steps by rank matrices as in Broyden [5]. The relaxation factor λ_k in (2.6) is determined as in Deuflhard [9], [10], where the rank-1 approximation technique is also described.

The linear system (2.7) is customarily solved by reducing it to an $n \times n$-system for the first component Δs_1:

$$\begin{aligned} & E\Delta s_1 = u \quad \text{with} \\ & E := A + BG_{m-1} \cdots G_2 G_1 \quad \text{and} \\ & u := -(F_{m-1} + BG_{m-1}F_{m-2} + \cdots + BG_{m-1} \cdots G_3 G_2 F_1), \end{aligned} \tag{2.9}$$

followed by a recursive computation of the other components of the correction Δs as

$$\Delta s_{j+1} = G_j \Delta s_j + F_j \quad \text{for } j = 1, \cdots, m-2. \tag{2.10}$$

Here one ordinarily computes two measures of the matrix E, denoted as "norm (E)" and "cond (E)", which measure the sensitivity of the problem to the initial value s_1. For a singularly perturbed problem as considered here, the solution depends quite sensitively on this initial value due to the large Lipschitz constant. The equation for Δs_1 in (2.9) corresponds to the equation for simple shooting, and the particular node distribution does not affect the condition number. For very sensitive problems, (2.9) is ill-conditioned, and an error in Δs_1 is strongly amplified by (2.10). Hence, for such problems as considered here, it is better to compute the corrections by solving the original system (2.7) which provides information about the node distribution and whose condition number can be influenced by changing the nodes. The following example illustrates the situation.

Example. Consider the boundary value problem

$$\varepsilon y'' - y = 0, \quad y(-1) = 1, \quad y(1) = 1 \tag{2.11}$$

with the relatively "large" parameter choices $\varepsilon = 10^{-3}$ and $\varepsilon = 10^{-4}$. For the system

(2.7)-(2.8) one obtains the following estimates for the norm and condition:

(2.12)

	$\varepsilon = 10^{-3}$	$\varepsilon = 10^{-4}$
norm $(DF) =$	$8.8 \cdot 10^{3}$	$2.4 \cdot 10^{10}$
cond $(DF) =$	$5.8 \cdot 10^{7}$	$2.7 \cdot 10^{14}$

(computed for (2.7)-(2.8)).

By way of comparison, if the exact solution of (2.11) is used as starting values with (2.9), one gets the estimates:

(2.13)

	$\varepsilon = 10^{-3}$	$\varepsilon = 10^{-4}$
norm $(E) =$	$2.6 \cdot 10^{26}$	$5.2 \cdot 10^{85}$

(computed for (2.9)).

The matrix E is singular and so there are no estimates for cond (E). Since the starting values are exact in this case, one expects that the resulting correction vector Δs computed from (2.9)-(2.10) with the pseudo-inverse should be small with $\|\Delta s\| < \mathrm{TOL}$; instead one finds:

$$\|\Delta s\|_2 = \begin{cases} 6.7 \cdot 10^{3} & \text{if } \varepsilon = 10^{-3}, \\ 5.4 \cdot 10^{115} & \text{if } \varepsilon = 10^{-4}, \end{cases} \quad \text{(Computed from (2.9)-(2.10)).} \tag{2.14}$$

Hence it is not possible to solve (2.7) using the reduction (2.9)-(2.10) in terms of Δs_1.

The matrix G_j in (2.8) measures the sensitivity of the solution of $(2.3)_j$ to the initial value s_j, for $j = 1, \cdots, m-1$. In particular, G_{BL} reflects the dependency of the outer solution on the initial value s_{BL}. The outer solution is not influenced by the large eigenvalues (cf. § 3), and therefore not every change of a component of the initial value s_{BL} causes a change in $y(x_{BL+1}, x_{BL}, s_{BL})$. For the example (2.11), the outer solution is $\bar{y} = 0$, and $G_{BL} = 0$. Ordinarily G_{BL} and hence also the matrix (2.8) is *not of full rank*. The solution of (2.7)-(2.8) can be computed directly by the pseudo-inverse, but it is very expensive to do this. Hence the present algorithm uses the following approach for the solution of (2.7)-(2.8). First the (block) column of the matrix (2.8) containing G_{BL} is taken out and placed as the last column. Taking into account the sparse structure of $DF(s)$, the resulting equivalent system is then transformed by Housholder transformations so that the matrix of the transformed system is upper triangular with the exception of an $n \times n$ matrix R occurring at the bottom of the last column, corresponding to the correction Δs_{BL} which occurs as the last component of the transformed column vector of corrections. In this way one finds an $n \times n$ system of the form $R\Delta s_{BL} =$ Known Vector, which is solved for Δs_{BL} using the pseudo-inverse obtained with the singular-value decomposition of Wilkinson and Reinsch [35]. The other corrections are then determined by back-substitution.

The rank of R is not known a priori and must be determined with the singular values. For this purpose an estimate (ratio of largest to smallest absolute diagonal element of the transformed matrix) of the condition number of (2.8) is computed during the elimination process. Together with the largest singular value of R, the rank is so determined that the estimate of the condition number does not exceed a machine-dependent limit or does not increase the previous estimate by more than a factor 100 respectively. In all examples tested, the rank decision was not critical, because the singular values were well separated.

The node distribution indicated in Fig. (2.2), where many nodes are placed densely in the boundary layers and none are placed in the outer interval, requires information on the boundary layer thicknesses and on the solution components which grow rapidly and limit the distances between adjacent nodes. This information is provided by an

examination of those eigenvalues EV of the functional matrix $D_y f$ of (2.1) which have large absolute real parts, where such eigenvalues can be effectively characterized by a condition of the type

$$|\mathrm{Re}\ EV| > 100 \cdot |b-a|^{-1}. \tag{2.15}$$

The precise value of the multiplicative factor on the right side, taken here to be 100, is not critical if the eigenvalues are well separated, as occurs in the examples of § 4. The listed value 100 in (2.15) corresponds to the choice $h = \frac{1}{20} \cdot |b-a|$ in (3.7) below.

It is convenient in the following to use the abbreviations $EV_{\max}$, $EV_{\min}$, and EV_+ respectively to denote the eigenvalues with largest absolute, smallest absolute, and largest positive (depending on the direction of integration) real part, among all eigenvalues satisfying (2.15). The algorithm then uses the following estimates for the boundary layer thickness BL and the largest distance D between adjacent nodes in the boundary layer:

$$BL = \frac{20}{\max\left(|\mathrm{Re}\ EV_{\min}|,\ 0.25 \cdot |\mathrm{Re}\ EV_{\max}|\right)} \tag{2.16}$$

and

$$D = \frac{10}{\mathrm{Re}\ EV_+}. \tag{2.17}$$

Note that (2.17) guarantees, in the linear case, that the growing solution components cannot exceed e^{10}. The ratio of BL to D determines the number of nodes to be placed inside the boundary layers at the endpoints, where, for convergence purposes, each boundary layer is *initially* required to have at least five nodes. The initial node distribution in the boundary layer intervals is taken to be uniform with equidistant nodes, and then this initial node distribution is generally modified during the Newton iterations (2.6) due to the changing of the eigenvalues of $D_y f$ which depend on the variables x and y. Hence the algorithm removes or inserts nodes as described in the following paragraphs. The removal of nodes is necessary if the boundary layer thickness is initially estimated too large or especially if there is no boundary layer at one or both endpoints. Additional nodes must be inserted if either of the boundary layers is estimated too thin or if an increasing value of $\mathrm{Re}\ EV_+$ in (2.17) requires a smaller distance D between the nodes.

After each iterative cycle, the boundary layer thicknesses are tested as follows. The algorithm checks whether the computed solution satisfies the "reduced system" (small parameter = 0) at the endpoints of the outer subinterval, where the identification of the reduced system is described in § 3 and does not require any a priori knowledge of the small parameter (or parameters). If this test is not valid at an endpoint of the previous outer subinterval, then the adjacent boundary layer interval is enlarged by inserting one or more additional nodes (the number depending on the error) beyond the previous boundary layer, with the spacing D between nodes computed by (2.17). In the same way the boundary layer thickness is reduced if the test is fulfilled at an endpoint of the previous outer interval. Several nodes are removed from the appropriate boundary layer interval if the test is fulfilled both for an endpoint of the outer interval along with several adjacent nodes beyond the outer interval. In the case that the computed solution satisfies the reduced system at all nodes in a previous boundary layer, then all nodes are removed from that layer and the outer interval is redefined to extend up to that boundary point.

The absolute real part of the eigenvalue EV_+ can increase quite rapidly inside a boundary layer, which requires a smaller value of D in accordance with (2.17) along with a consequent refinement of the previous node distribution. However, every such refinement requires a new start of the Newton iteration (2.6), and so it is desirable to find very early a node distribution which enables convergence without further reduction in the value of D. This can be done by linearization. One first computes $EV_+ = EV_+(x, y)$, and then, after one iteration in (2.6), one computes $EV_+(x, y+\lambda \cdot \Delta y)$. If the real part of this latter eigenvalue is greater than the previous value before the iteration, and if the relaxation factor satisfies $\lambda < 1$, then a first order approximation $\overline{EV}_+$ can be given as

$$\overline{EV}_+ = \left(1-\frac{1}{\lambda}\right) \cdot EV_+(x, y) + \frac{1}{\lambda} \cdot EV_+(x, y+\lambda\, \Delta y). \tag{2.18}$$

In this case, the largest distance D is computed using this result (2.18) in (2.17). At the endpoints ($x = a$, or $x = b$), new nodes are placed logarithmically (doubling interval length with minimal distance $D_{\min} = (EV_+)^{-1}$). Away from the endpoints, additional nodes are placed equidistantly until (2.17) is satisfied.

The boundary values s_1, s_m are used for the computation of the eigenvalues in the boundary layers. In many cases it is possible to compute the outer solution of the problem initially, and in such a case this outer solution can be used to provide good starting values for the s_j. One can also use techniques such as those of Biller [4] and Watkins [34] to choose or improve the starting values.

3. A stable initial value integrator for the outer solution. The distribution of nodes in the multiple shooting, as illustrated in Fig. (2.2), uses for the outer solution a single subinterval $[x_{BL}, x_{BL+1})$ along with the corresponding initial value problem $(2.3)_{BL}$. The outer solution for singularly perturbed problems is insensitive to the large eigenvalues EV of the functional matrix $D_y f$ discussed in § 2, and so the large eigenvalues can be disregarded in the multiple shooting during the integration of the initial value problem over the outer interval $[x_{BL}, x_{BL+1})$. For this purpose a suitably stable integration routine is required. The stability function $R(z)$ must fulfill the following condition

$$|R(z)| < 1 \tag{3.1}$$

for all z with large absolute real part. In particular this condition must be valid also for large positive z. Here a second subdiagonal Padé approximation of the exponential function is taken, and in order to obtain a fourth order method, one has the following stability function:

$$R(z) = \frac{1+z/4}{1-3z/4+z^2/4-z^3/24}. \tag{3.2}$$

The corresponding stability region for (3.2) is indicated in Fig. 3.3.

The stability function (3.2) corresponds to an implicit Runge-Kutta method with coefficients as given in Chipman [8]. One must solve a nonlinear system for the computation of the increment functions in the resulting initial value integrator. The method of successive approximation is customarily used for the nonlinear system in such cases, but this method is not used here because of the large Lipschitz constant of the singularly perturbed differential equation. Indeed, this large Lipschitz constant permits only very small integration steps for a successful computation by successive approximation. Hence a modified Newton method is used instead here to solve the nonlinear system. The resulting increased overhead is more than balanced by the large

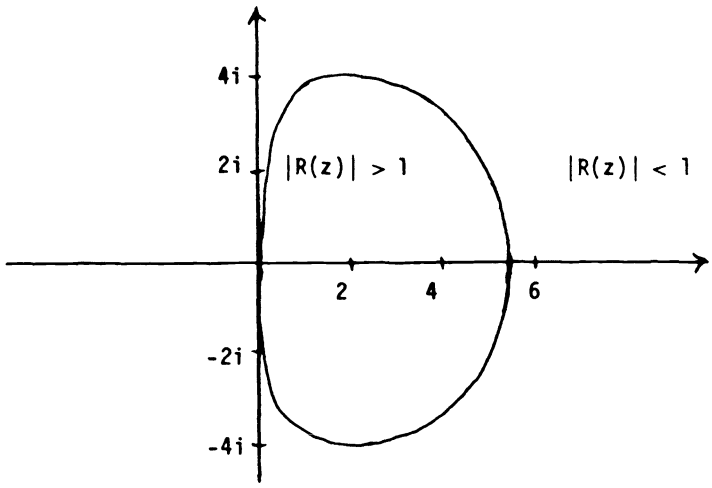


FIG. 3.3. *Stability region for* (3.2).

stepsizes permitted in the integration. The special structure of the differential equations in a singularly perturbed system (some differential equations contain small parameters, others not) requires scaling of the nonlinear equations. For this purpose the sums of the rows of the Jacobian matrix $D_y f$ of the outer system $(2.3)_{BL}$ divided by the number of differential equations are taken for the scaling factors. This scaling permits the solution of the nonlinear system without knowledge of the value(s) of the small parameter(s) or even in which differential equations the parameter appears.

In order to obtain a reliable integration method for the outer initial value problem $(2.3)_{BL}$, a stepsize control is required. For this purpose a second approximation of different order is computed, because this is much cheaper than estimating the error by using the earlier order-4 method and halving the stepsize. The chosen method for this second approximation must satisfy the stability condition (3.1) and should also be easy to compute. Therefore an implicit Runge-Kutta method of order two is used. The stability function for this latter method is

$$R(z)=\frac{1}{1-z+z^2/2}, \tag{3.4}$$

and its stability region contains that of (3.2). The coefficients for this implicit method are also found in [8].

Based on these two Runge-Kutta methods, one finds the following estimate for an optimal stepsize h (cf. Stoer and Bulirsch [31]):

$$h_{\text{new}}=2\cdot h_{\text{old}}\cdot\left(\frac{\text{TOL}}{\text{EST}}\right)^{1/3}, \qquad 0.5\cdot h_{\text{old}}\leqq h_{\text{new}}\leqq 2.0\cdot h_{\text{old}},$$

TOL: relative error

(3.5) EST: error estimate

$$\text{EST}=\max_i |y_{4i}-y_{2i}|, \quad y_2,\ y_4 \text{ approximations of orders two and four respectively.}$$

The factor 2 in the estimate for h_{new} in (3.5) has been determined by solving the test

examples in Enright, Bedet, Farkas and Hull [11] and Enright, Hull and Lindberg [12], and takes into account the fact that the value of EST overestimates the real error.

It is necessary that the stepsizes are large enough so that the eigenvalues with large absolute real parts are damped out during integration. If one takes the following value of z in (3.2),

$$z = \lambda \cdot h \quad \text{with } \lambda\text{: eigenvalue of large real part,} \quad h\text{: stepsize,} \tag{3.6}$$

then for a linear system of differential equations, condition (3.1) is satisfied for all stepsizes h with:

$$h > \frac{5}{|\operatorname{Re} \lambda|} \quad \text{(see Fig. 3.3).} \tag{3.7}$$

This inequality always holds for large Re λ, and since λ is typically inversely proportional to some positive power of a small parameter in the singularly perturbed problems considered here, it follows that the method works in the desired way. If the large eigenvalues have been computed (see § 2), one can determine a smallest stepsize h from (3.7), and then the integration can be terminated if the stepsize is underestimated. This can happen if the initial value s_{BL} in $(2.3)_{BL}$ does not satisfy the reduced system for the outer solution.

4. Numerical illustrations. The following computations were performed in FORTRAN IV with single precision (48 bit mantissa) on the CDC CYBER 175 of the Leibniz Rechenzentrum der Bayerischen Akademie der Wissenschaften. The solutions were computed with tolerance EPS $= 10^{-4}$. The Runge-Kutta-Fehlberg method RKF7 (see Fehlberg [13]) with the modifications of Seydel [27] was used for the solution of the initial value problems in the boundary layers. The abbreviations CT, IT, NFC denote respectively *computer time* in seconds, the number of *iterations* of Newton's method, and the number of *function calls* of the right side of (2.1).

4.1. Test problem with two parameters. This example from O'Malley [21] provides a useful test because the solution behaves differently in the boundary layers depending on the ratio of the two small parameters. The problem is the following, where the three cases $\varepsilon = \mu,\ \mu^2,\ \mu^{2.5}$ are considered:

$$\varepsilon y'' + \mu y' - y = 0, \quad y(0) = 1, \quad y(1) = 0.5 \qquad (\varepsilon = \mu, \mu^2, \mu^{2,5}). \tag{4.1.1}$$

In every case the outer solution is given as

$$\bar{y} = 0. \tag{4.1.2}$$

For the eigenvalues of the Jacobian functional matrix of (the equivalent first order system for) (4.1.1) there holds

$$EV_{1,2} = 0.5\left(-\frac{\mu}{\varepsilon} \mp \frac{1}{\varepsilon} \cdot [\mu^2 + 4 \cdot \varepsilon]^{1/2}\right), \tag{4.1.3}$$

from which one has the following approximations for the exact solution in the boundary layers:

$$y(x) \sim \begin{cases} \exp(EV_1 x) & \text{near } x = 0, \\ 0.5 \exp(EV_2(x-1)) & \text{near } x = 1. \end{cases} \tag{4.1.4}$$

From (4.1.3)-(4.1.4) one gets for the boundary layer thicknesses $BL_0(x=0)$ and $BL_1(x=1)$,

(4.1.5)

ε	BL_0	BL_1	
μ	$O(\mu^{0.5})$	$O(\mu^{0.5})$	$BL_1 \approx BL_0$
μ^2	$O(\mu)$	$O(\mu)$	$BL_1 \approx 2.6BL_0$
$\mu^{2.5}$	$O(\mu^{1.5})$	$O(\mu)$	

The numerical calculations for (4.1.1) were started with the two nodes $x_1=0$, $x_2=1$ with corresponding s_j taken from the exact outer solution, $s_1=s_2=0$.

In the case $\varepsilon=\mu$ where both eigenvalues have asymptotically the same value, (2.16)-(2.17) yields a new distribution of five equidistant nodes in each boundary layer interval, and in this case this node distribution remains unchanged during the remaining iterations. The table in (4.1.6) indicates the amount of computation:

(4.1.6)

μ	10^{-4}-10^{-11}	10^{-12}	
IT	4	5	$(\varepsilon=\mu)$.
CT	0.19-0.66	1.0	
NFC	4,328-23,578	37,954	

In the case $\varepsilon=\mu^2$ the eigenvalue EV_2 of (4.1.3) determines the estimate of the boundary layer thickness in (2.16) at both endpoints. Together with (2.17) one obtains a node distribution with seven equidistant nodes in each layer. After three iterations the algorithm reduces the thickness of the boundary layer at $x=0$ and removes three nodes there while leaving the earlier node distribution unchanged at $x=1$. Hence the algorithm automatically models correctly the situation which obtains for the exact solution where, as indicated by (4.1.4)-(4.1.5), the boundary layer thickness at $x=0$ is only about half that at $x=1$. The computational results are given in (4.1.7) for the indicated range of values of μ.

(4.1.7)

μ	10^{-2}	10^{-3}	10^{-4}	10^{-5}	10^{-6}	
IT	4	10	4	5	5	$(\varepsilon=\mu^2)$.
CT	0.33	1.7	0.63	1.0	1.3	
NFC	9,193	59,230	21,955	37,480	47,013	

In the case $\varepsilon=\mu^{2.5}$, the magnitudes of the eigenvalues (4.1.3) differ considerably for small μ. The rule (2.16) automatically uses EV_1 in obtaining the initial estimate for the boundary layer thickness BL at each endpoint, and then (2.16) and (2.17) yield an initial node distribution of ten equidistant nodes in each layer. After three iterations, the algorithm reduces the thickness of the layer at $x=0$ and removes six nodes there, while at the same time the algorithm enlarges the layer at $x=1$ and inserts additional nodes there in accordance with (2.17). The algorithm inserts one additional node near $x=1$ in the case $\mu=10^{-2}$, while two additional nodes are inserted for both cases $\mu=10^{-3}$ and $\mu=10^{-4}$, and four additional nodes are inserted for $\mu=10^{-5}$. In all cases the computed boundary layer thicknesses are correct. The computational results are indicated in (4.1.8).

(4.1.8)

μ	10^{-2}	10^{-3}	10^{-4}	10^{-5}	
IT	6	6	6	7	$(\varepsilon=\mu^{2.5})$.
CT	0.88	2.3	6.6	24	
NFC	29,058	86,667	266,649	1,000,923	

The increasing expense for smaller μ is explained by the stiff behavior of the solution in the right boundary layer. The boundary layer thickness there is determined by EV_2 (cf. (4.1.4)) while the negative eigenvalue EV_1 of (4.1.3) influences the stepsize of the Runge-Kutta method RKF7. As μ becomes smaller and smaller, the eigenvalues differ more and more in their magnitudes, and the resulting stepsizes in RKF7 become very small compared to the boundary layer width at $x=1$ (cf. (4.1.5)). The use of a stiff integrator for the initial value problems in the boundary layers leads to the results of (4.1.9), where the stiff integrator used here is GRK4T from Kaps and Rentrop [17].

(4.1.9) ($\varepsilon = \mu^{2.5}$)

μ	10^{-2}	10^{-3}	10^{-4}	10^{-5}
IT	6	6	6	7
CT	3.8	7.4	10.8	12.6
NFC	19,475	37,610	54,957	64,166

NFC is reduced dramatically, but the amount of computer time is reduced only for $\mu = 10^{-5}$.

In all three cases the numerical results are everywhere within the required accuracy. One or two decimals only are lost at the two nodes which serve as the endpoints of the outer interval where the algorithm connects the outer solution to the boundary layers.

4.2. Semiconductor diode model. This simplified model of a semiconductor device consists of the equations (see Vasil'eva and Stel'makh [33])

$$\begin{aligned} \mu^2 E' &= p - n + N, \\ p' &= pE - J, \\ n' &= -nE + J \end{aligned} \tag{4.2.1}$$

where E, p, n, $N=1$, $J=1$ denote field strength, hole and electron concentration, and doping and current densities respectively. The quantity μ^2 corresponds to the Debye length which is a measure of the physical thickness of the boundary layer between the p-doped and n-doped regions. Here the n-region of a symmetric diode is considered, which results in the boundary conditions:

$$p(0) = n(0), \quad p(1) = p_1, \quad n(1) = n_1 + N. \tag{4.2.2}$$

These conditions are more general than those given in [33]. An existence and local uniqueness proof for (4.2.1)-(4.2.2) for small μ (and for variable doping and current) is given in Smith [29] (see also [30]) along with an analysis of the behavior of the solution functions for small μ. In Maier and Smith [19] numerical solutions based on the results of [29] are computed for different boundary conditions. The outer solution functions ($\mu = 0$) for (4.2.1)-(4.2.2), denoted as $\bar{E}$, $\bar{p}$ and $\bar{n}$, are given as

$$\begin{aligned} &\bar{E} = \frac{2J}{u}, \quad \bar{p} = \frac{u-N}{2}, \quad \bar{n} = \frac{u+N}{2} \quad \text{with} \\ &u = [N^2 + 4p_1 \cdot (N + n_1) + 4NJ \cdot (1-x)]^{1/2}. \end{aligned} \tag{4.2.3}$$

There are boundary layer effects for all solution components E, p, n at the left endpoint $x=0$ which corresponds to the diode junction. However, the situation is different at

the right endpoint where the boundary layer structure depends on the particular boundary values p_1, n_1 used in (4.2.2). The situation can be summarized as follows (see [29]):

(4.2.4)

a) $n_1 = p_1 = 0$ no boundary layers at $x = 1$;

b) $n_1 > 0 = p_1$ E and n, but not p, have boundary layers at $x = 1$;

c) $n_1 \neq p_1 > 0$ all solution components E, n, p have boundary layers at $x = 1$.

Here numerical solutions are computed for the cases

(4.2.5a) $$n_1 = p_1 = 0,$$

(4.2.5b) $$n_1 = 0, \qquad p_1 = 1.$$

For the calculations in both cases the starting data consist of the two nodes $x_1 = 1$, $x_2 = 0$ along with the corresponding values from the outer solution (4.2.3) at these points. The amount of computation required is shown in (4.2.6) for the indicated range of values for μ^2 between 10^{-6} and 10^{-12}:

(4.2.6)

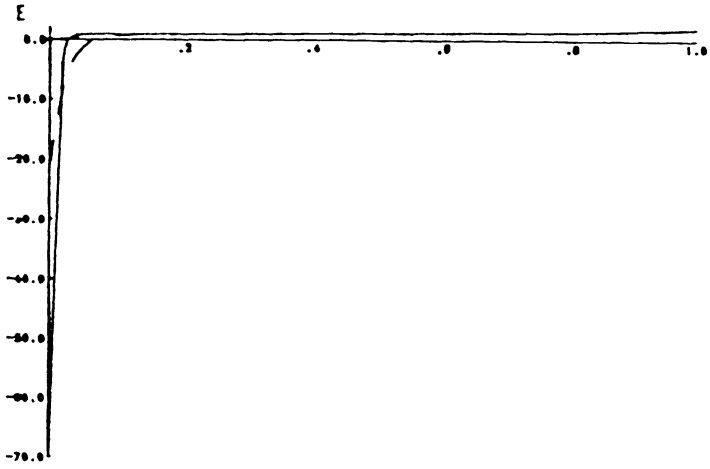
μ^2	$n_1 = p_1 = 0$			$n_1 = 0, p_1 = 1$		
	IT	CT	NFC	IT	CT	NFC
10^{-6}	5	0.65	11,649	8	1.2	24,503
10^{-7}	5	0.85	15,808	8	1.3	26,426
10^{-8}	5	0.9	18,069	8	1.6	34,798
10^{-9}	5	1.4	29,689	8	2.4	54,872
10^{-10}	5	1.4	29,256	8	3.3	81,175
10^{-11}	5	1.9	39,885	8	4.0	96,396
10^{-12}	5	2.6	54,760	9	5.1	123,934

In every case the algorithm initially distributes five equidistant nodes in each boundary layer subinterval. This node distribution remains unchanged until convergence for (4.2.5b), whereas all of the nodes near $x = 1$ are removed for (4.2.5a) after two iterations. Hence the algorithm agrees with (4.2.4) in both cases. One obtains the following estimates for the boundary layer thicknesses at convergence, where the values in parentheses are the values used in [19].

(4.2.7)

	$n_1 = p_1 = 0$	$n_1 = 0, p_1 = 1$
$x = 0$	*BL*: $13.5 \cdot \mu$ $(15 \cdot \mu)$	*BL*: $13.5 \cdot \mu$ $(15 \cdot \mu)$
$x = 1$	—	*BL*: $11.5 \cdot \mu$ $(10 \cdot \mu)$.

The solution values computed here agree within the required accuracy with the values given in Ascher [3] for $E(0)$, $p(0)$, $n(0)$ in the case (4.2.5a) and with the values for E, p, n at the endpoints for different boundary conditions obtained in [19] using a different algorithm more closely tied to the asymptotic analysis of [29]. The graphs of the computed solution functions are indicated in Figs. 4.2.8 and 4.2.9.



Field Strength E for (4.2.5a)
Hole and Electron Densities p, n for (4.2.5a)

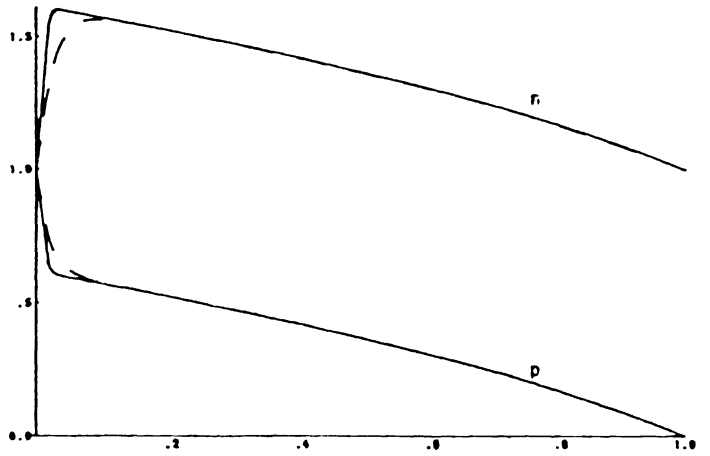
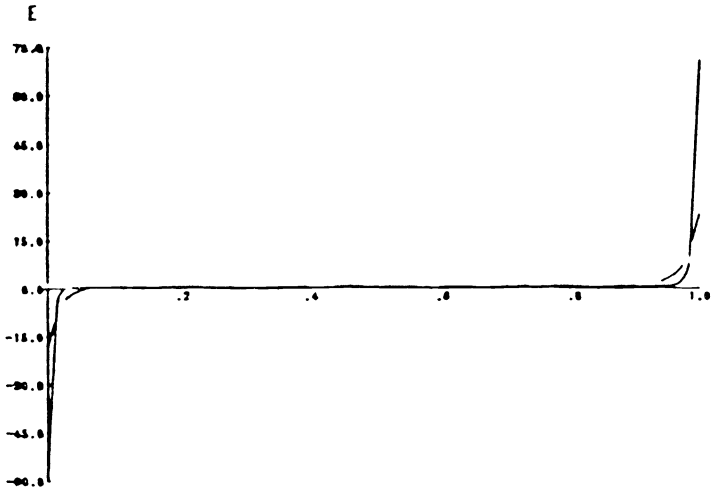


FIG. 4.2.8. $\mu^2 = 10^{-3}$ (- - -), $\mu^2 = 10^{-4}$ (——).

4.3. Bipolar transistor. The model of a bipolar transistor is discussed in Van Driel [32] and in Rentrop [26]:

$$
\begin{aligned}
V''(x) &= \frac{q}{\varepsilon} \cdot (p(x) + Pt_1(x) + Pt_2(x) - n(x) - N_D), \\
p'(x) &= \frac{1}{D_p} \cdot \left(\mu_p p(x) \cdot V'(x) - \frac{1}{q} \cdot J_p(x) \right), \\
n'(x) &= -\frac{1}{D_n} \cdot \left(\mu_n n(x) \cdot V'(x) - \frac{1}{q} \cdot J_n(x) \right), \qquad \text{(4.3.1a)} \\
J_p'(x) &= q \cdot (J - R_1(x) - R_2(x)), \\
J_n'(x) &= -J_p'(x)
\end{aligned}
$$



Field Strngth for (4.2.5b)
Hole and Electron Densities p, n for (4.2.5b)

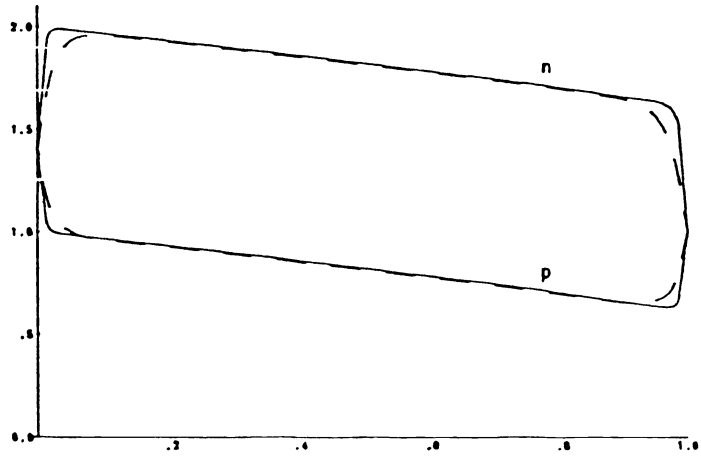


FIG. 4.2.9. $\mu^2 = 10^{-3}$ (- - -), $\mu^2 = 10^{-4}$ (——).

with

$$Pt_i(x) = \frac{Nt_i \cdot (\gamma_i p(x) + \delta_i)}{\gamma_i p(x) + \beta_i n(x) + \alpha_i + \delta_i}, \qquad i = 1, 2,$$

(4.3.1b)

$$R_i(x) = \frac{p(x) \cdot n(x) - n_{\text{intr}}^2}{\tau_{n_i}(p(x) + p_{c_i}) + \tau_{p_i}(n(x) + n_{c_i})}, \qquad i = 1, 2$$

and with constants

(4.3.1c)

$$\begin{array}{llll} q = 1.6 \cdot 10^{-19}, & \varepsilon = 10^{-12}, & J = 5 \cdot 10^{15}, & N_D = 10^7, \\ \mu_p = 10, & \mu_n = 1, & D_p = 0.26, & D_n = 0.026, \\ \alpha_1 = 0.1, & \alpha_2 = 0.058, & \beta_1 = 10^{-9}, & \beta_2 = 10^{-5}, \\ \gamma_1 = 5 \cdot 10^{-7}, & \gamma_2 = 10^{-5}, & \delta_1 = 4.1 \cdot 10^{-3}, & \delta_2 = 1.4 \cdot 10^3, \\ Nt_1 = 10^{16}, & Nt_2 = 10^{13}, & \tau_{p_1} = 2 \cdot 10^{-10}, & \tau_{p_2} = 10^{-8}, \\ \tau_{n_1} = 10^{-7}, & \tau_{n_2} = 10^{-8}, & p_{c_1} = 2 \cdot 10^5, & p_{c_2} = 5.8 \cdot 10^3, \\ n_{c_1} = 4.1 \cdot 10^6, & n_{c_2} = 1.4 \cdot 10^8, & n_{\text{intr}}^2 = n_L \cdot p_L = 8.2 \cdot 10^{11}. & \end{array}$$

The problem is considered on the interval $0 \leqq x \leqq L$ with $L = 2 \cdot 10^{-4}$ and with boundary conditions:

$$
\begin{aligned}
& V(0) = 0, && V(L) = 10, \\
& p(0) = 4.8 \cdot 10^{12}, && p(L) = p_L = 10^7, \\
& n(0) = 0.17, && n(L) = n_L = 8.2 \cdot 10^4.
\end{aligned} \tag{4.3.2}
$$

The problem is scaled as in [26], as follows:

$$
\begin{aligned}
& \bar{V}(x) = V(x), && \\
& \bar{p}(x) = \frac{q}{\varepsilon} \cdot p(x), && \bar{J}_p(x) = \frac{1}{\beta_2} \cdot J_p(x), \\
& \bar{n}(x) = \frac{q}{\varepsilon} \cdot n(x), && \bar{J}_n(x) = \frac{1}{\beta_2} \cdot J_n(x).
\end{aligned} \tag{4.3.3}
$$

In the sense of singular perturbation theory, no small parameter is given explicitly for (4.3.1). It might be expected, by analogy with the related system (4.2.1), that a small parameter might occur implicitly in the first equation of (4.3.1a). However, the results in [26] suggest instead that a small parameter enters here into the equations for p' and n', and this suggestion is confirmed by the present numerical solution. Another feature of this example is that the large eigenvalues are actually rather small as compared with those in many other singularly perturbed problems. Moreover, the large eigenvalues here change their magnitudes rapidly during the interval of integration, as indicated in Fig. 4.3.4 which gives the graph of the absolute value of the largest absolute eigenvalue for $0 \leqq x \leqq L$ (scaled to $L = 1$).

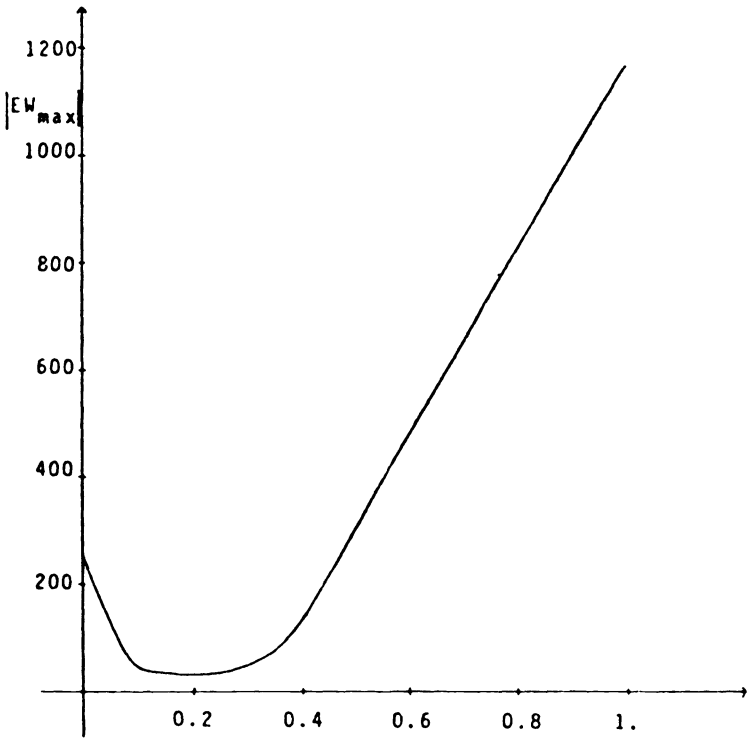


FIG. 4.3.4. *Graph of largest absolute eigenvalue.*

The subdivision of the interval into outer and boundary layer intervals cannot be done in the same way as in the other examples since the large eigenvalues influence the solution for $x \in [0, 0.5 \cdot L]$ and cannot be neglected in that subinterval. Moreover, these eigenvalues can be neglected only by using large stepsizes in accordance with (3.7), and this is not allowed here by the stepsize control. Hence the interval $[0, 0.6 \cdot L]$ is taken as the left boundary layer. The right boundary layer is taken to be determined by (2.16), which gives $BL = 0.02 \cdot L$ for the thickness of the right boundary layer. Hence the outer interval is taken initially to be $[0.6 \cdot L, 0.98 \cdot L]$, and the initial node distribution uses 25 nodes. The initial choices for the shooting initial values s_j are computed as in [26] using a homotopy chain with the successive values $(\mu_p, \mu_n) = \{0.01, 0.001\}$, $\{0.1, 0.01\}$, $\{1, 0.1\}$, $\{3, 0.3\}$, $\{5, 0.5\}$, $\{10, 1\}$. The solution is easy to compute with the initial choice $\{\mu_p, \mu_n\} = \{0.01, 0.001\}$, and then the resulting computed solution values are used as starting values for the next parameter choice $\{\mu_p, \mu_n\} = \{0.1, 0.01\}$, etc. With this homotopy trajectory one gets the computed solution for the original problem with $\{\mu_p, \mu_n\} = \{10, 1\}$ after 12 iterations of the Newton method, using 68.5 seconds of computer time, and 641,115 calls of the right-hand side of (4.3.1). The graphs of the computed solution functions are indicated in Figs. 4.3.5–4.3.7.

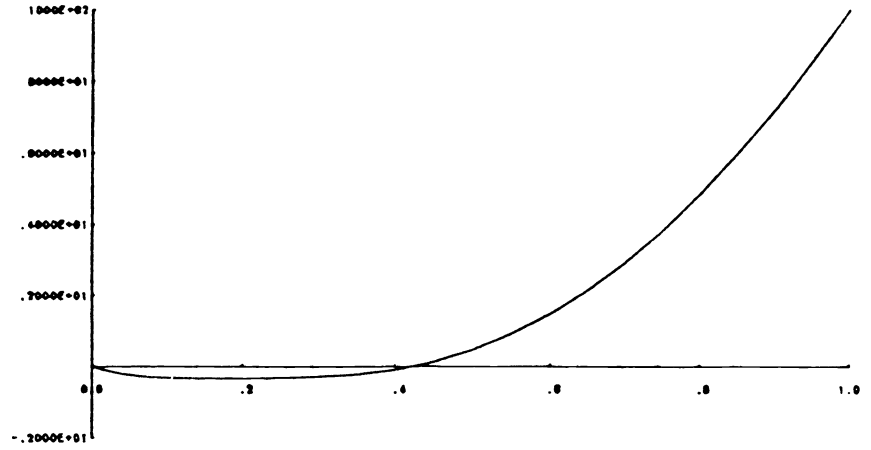


FIG. 4.3.5. *Potential* $V(x)$.

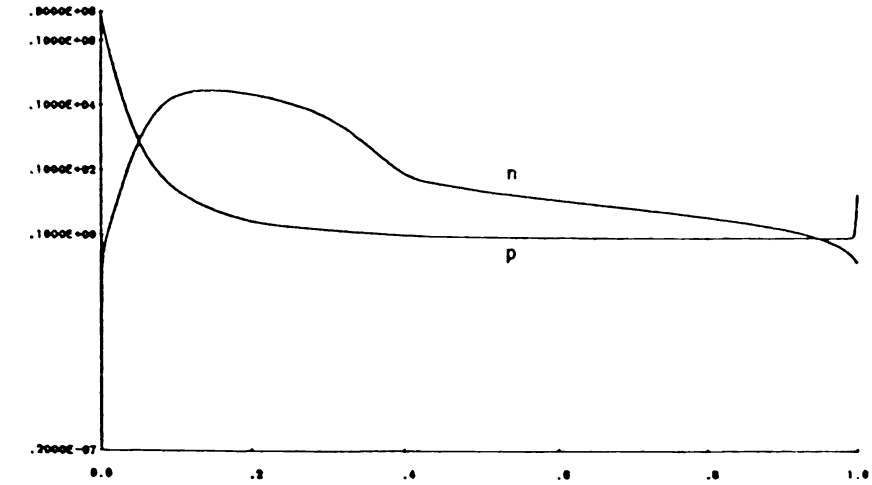


FIG. 4.3.6. *Hole and electron densities* $p(x)$, $n(x)$.

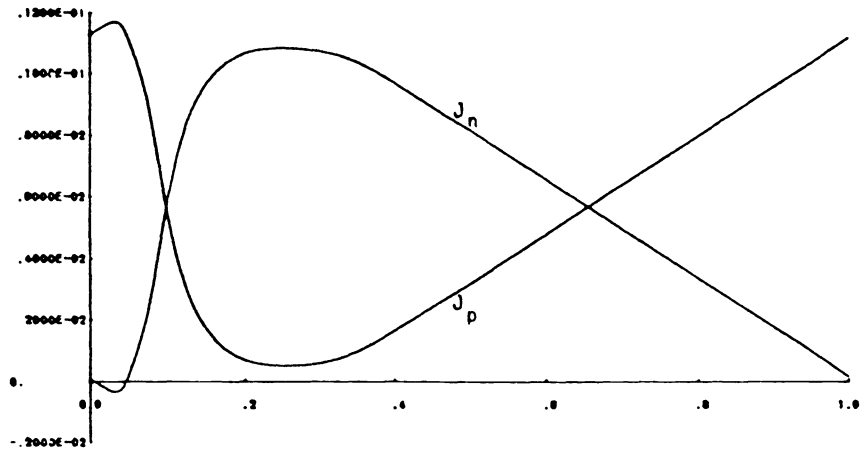


FIG. 4.3.7. *Hole and electron current densities* $J_p(x)$, $J_n(x)$.

4.4. Electrical properties of electron-irradiated silicon. This boundary value problem is described in Sigfridsson and Lindström [28] and is solved by a direct discretization (difference method) in Abrahamsson [1]. One has the problem:

$$\begin{aligned} I \cdot y' &= F(x, y, z), \qquad y(0) = 1, \\ I \cdot z' &= G(x, y, z), \qquad z(1) = 0 \end{aligned} \tag{4.4.1a}$$

where y, z, and I are the normalized electron, hole, and current densities respectively. The right sides of the differential equations are given as

$$\begin{aligned} F(x, y, z) &= (y + \beta z)\left(A_1 y \tilde{f} - \sum_{i=1}^{N_A} \hat{f}_i - \sum_{j=1}^{N_D} \bar{f}_j \right), \\ G(x, y, z) &= (y + \beta z)\left(A_1 z \tilde{f} + \frac{1}{\beta} \sum_{i=1}^{N_A} \hat{f}_i + \frac{1}{\beta} \sum_{j=1}^{N_D} \bar{f}_j \right) \end{aligned} \tag{4.4.1b}$$

where

$$\begin{aligned} \tilde{f} = 1 - y + z &- \sum_{i=1}^{N_A} \gamma_{a_i}(x) \frac{y + \alpha_{a_i} \kappa_{ap_i}}{y + \kappa_{an_i} + \alpha_{a_i}(\kappa_{ap_i} + z)} \\ &+ \sum_{j=1}^{N_D} \gamma_{d_j}(x) \frac{\kappa_{dn_j} + \alpha_{d_j} z}{y + \kappa_{dn_j} + \alpha_{d_j}(\kappa_{dp_j} + z)} \end{aligned} \tag{4.4.1c}$$

and

$$\begin{aligned} \hat{f}_i &= \alpha_{a_i} A_{an_i} \gamma_{a_i}(x) \frac{yz - \kappa_{an_i} \kappa_{ap_i}}{y + \kappa_{an_i} + \alpha_{a_i}(\kappa_{ap_i} + z)}, \\ \bar{f}_j &= \alpha_{d_j} A_{dn_j} \gamma_{d_j}(x) \frac{yz - \kappa_{dn_j} \kappa_{dp_j}}{y + \kappa_{dn_j} + \alpha_{d_j}(\kappa_{dp_j} + z)} \end{aligned} \tag{4.4.1d}$$

and where the constants are given as

$$\begin{aligned} &\beta = \tfrac{1}{3}, && \kappa_{ap_1} = 1.854 \cdot 10^{-4}, && A_1 = 0.05162, \\ &N_A = 2, && \kappa_{an_1} = 21.47, && \alpha_{a_i} = \alpha_{d_j} = 1 \\ &N_D = 1, && \kappa_{ap_2} = 0.1021, && \qquad \text{for all } i, j, \end{aligned}$$

$$\gamma_{a_1}(x) = 15, \quad \kappa_{an_2} = 3.899 \cdot 10^{-2}, \quad A_{an_i} = A_{dn_j} = 2.222 \cdot 10^{-3} \text{ for all } i, j,$$
$$\gamma_{a_2}(x) = 10, \quad \kappa_{dp_1} = 2.902 \cdot 10^{3},$$
$$\gamma_{d_1}(x) = 400, \quad \kappa_{dn_1} = 1.371 \cdot 10^{-6}, \quad I \in [10^{-12}, 1].$$

This boundary value problem is interesting in two ways: first, the eigenvalues are $O(1/I)$ which yields very narrow boundary layers which are physically interesting for small I, and second, the solution behaves quite differently in the two boundary layers, as indicated in Fig. 4.4.2. At $x = 0$ the solution is governed by an absolute increasing negative eigenvalue, while at $x = 1$ one eigenvalue changes its sign and causes a turning point inside the boundary layer.

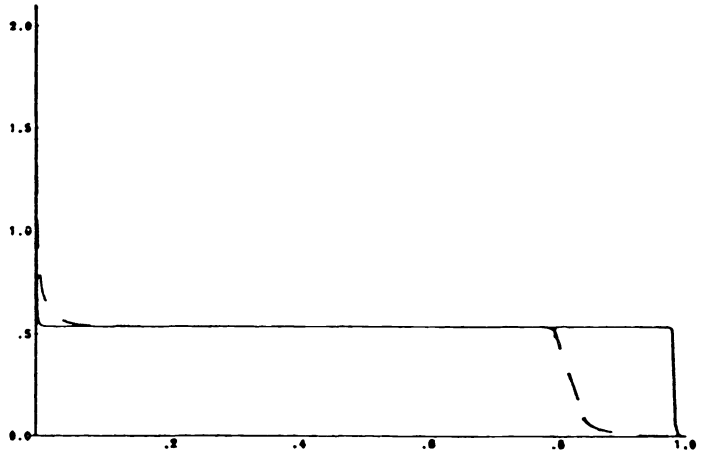


FIG. 4.4.2. *Normalized hole density z for* $I = 10^{-4}$ (- - -), $I = 10^{-5}$(——).

The outer solution $\bar{y}$, $\bar{z}$ is computed by solving the reduced system

$$F(x, \bar{y}, \bar{z}) = 0, \qquad G(x, \bar{y}, \bar{z}) = 0. \tag{4.4.3}$$

It is shown in [1] that (4.4.3) has a unique solution satisfying the conditions $\bar{y} > 0$, $\bar{z} > 0$, and the resulting solution functions are given as

$$\bar{y} = 7.4161 \cdot 10^{-3}, \qquad \bar{z} = 0.53667. \tag{4.4.4}$$

For the calculation here the starting values are taken to consist of these outer values at the two nodes $x_1 = 1$, $x_2 = 0$. The initial estimates for the eigenvalues are

$$EV_1 = 1.75 \cdot 10^{-2}/I, \qquad EV_2 = -3.86 \cdot 10^{-3}/I. \tag{4.4.5}$$

Then (2.16) and (2.17) produce an initial node distribution of ten equidistant nodes at each endpoint.

In the left boundary layer at $x = 0$ the solution is determined by an absolute increasing eigenvalue. One obtains for the computed solution at $x = 0$ the results

$$EV_1 = 7.66 \cdot 10^{-2}/I, \qquad EV_2 = -0.418/I. \tag{4.4.6}$$

Consequently the initial grid is refined by the stepsize control (2.18). After the first iteration, nine additional nodes are distributed logarithmically in the previous first subinterval adjoining $x = 0$ within the left boundary layer, while one additional node is placed in the middle of the previous second subinterval within the boundary layer

near $x=0$. After three iterations, the left boundary layer thickness is reduced by removing the rightmost node of this layer.

In the right boundary layer at $x=1$ the solution is influenced by the changing of sign of the eigenvalue EV_1 which causes a turning point. One obtains for the computed solution at $x=1$ the results

$$EV_1 = -1.27 \cdot 10^{-3}/I, \qquad EV_2 = -6.69 \cdot 10^{-4}/I. \tag{4.4.7}$$

No additional nodes are inserted by the algorithm, but the boundary layer thickness is reduced by taking out three nodes.

The amount of computation is shown in (4.4.8) for the indicated values of I between 10^{-5} and 10^{-12}.

(4.4.8)

I	IT	CT	NFC
10^{-5}	13	2.1	33,667
10^{-6}	13	2.1	34,774
10^{-7}	13	2.3	37,020
10^{-8}	13	2.2	34,175
10^{-9}	13	2.3	35,516
10^{-10}	13	2.4	37,731
10^{-11}	13	2.1	33,512
10^{-12}	12	2.1	33,643

The computed solution values agree in all five decimal places with the numerical values computed in Abrahamsson [1] for the case $I=10^{-12}$.

4.5. Test problem with failure of existence and/or uniqueness. This example provides a useful test because the problem has either no solution of boundary layer type, one solution of boundary layer type, or two such solutions, depending on the precise numerical value of the specified boundary value β,

$$\varepsilon^2 u'' = (u^2-1)(u^2-4), \qquad u'(0)=0, \quad u(0)=\beta. \tag{4.5.1}$$

Specifically, let β_1 and β_2 be the real roots of the respective cubics $p_1(\beta)$ and $p_2(\beta)$ given as

$$\begin{aligned} p_1(\beta) &:= 3\beta^3 - 6\beta^2 - 16\beta + 38, \\ p_2(\beta) &:= 3\beta^3 + 12\beta^2 + 11\beta - 4, \end{aligned} \tag{4.5.2}$$

with

$$\beta_1 \doteq -2.40569 \quad \text{and} \quad \beta_2 \doteq +0.275279. \tag{4.5.3}$$

Then it is shown in [30, Chap. 10] that, as $\varepsilon \to 0+$, (4.5.1) has a solution u_1 of boundary layer type satisfying

$$\lim_{\substack{\varepsilon \to 0+ \\ \text{Fixed } 0<x<1}} u_1(x,\varepsilon) = -1 \tag{4.5.4}$$

for any fixed $\beta > \beta_1$, but no such solution exists for $\beta < \beta_1$, and similarly (4.5.1) has a solution u_2 of boundary layer type satisfying

$$\lim_{\substack{\varepsilon \to 0+ \\ \text{Fixed } 0<x<1}} !u_2(x,\varepsilon) = +2 \tag{4.5.5}$$

for any fixed $\beta > \beta_2$, but no such solution exists for $\beta < \beta_2$, as indicated in Fig. 4.5.6.

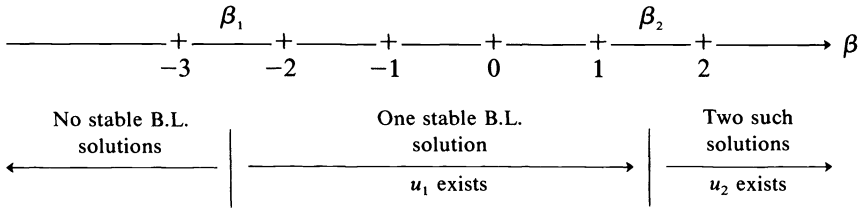


FIG. 4.5.6

For example the case $\beta = 0.5$ leads to two stable solutions of boundary layer type, as indicated in Fig. 4.5.7.

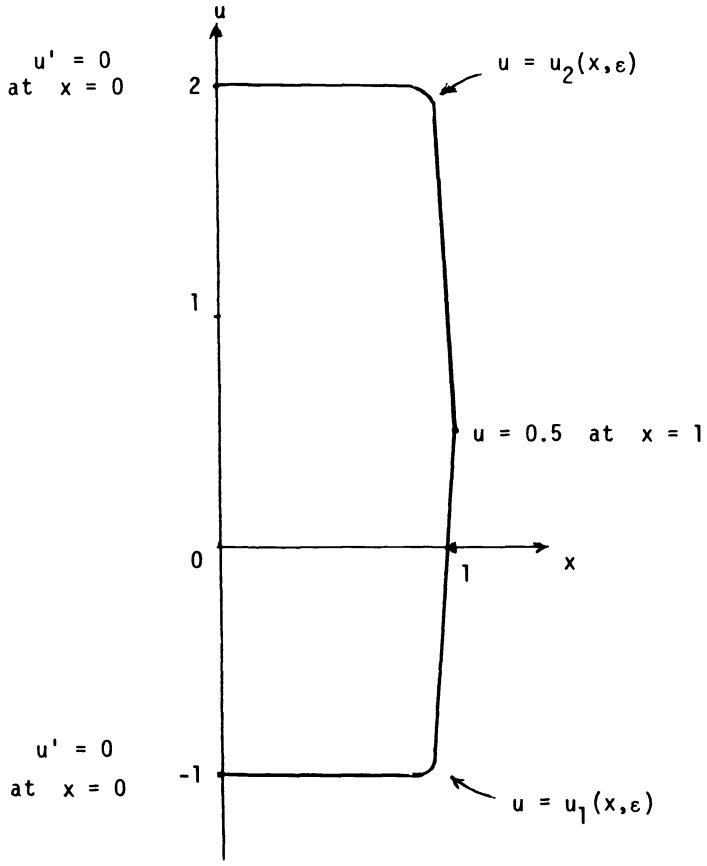


FIG. 4.5.7. *Lack of uniqueness.*

The following different choices are used in the calculations for the boundary value β,

(4.5.8) $$\beta = -3.0, -2.41, -2.4, -1.0, +0.27, +0.28, +0.5,$$

so that all three regimes of (4.5.6) are included. The calculation is performed *twice* for each choice of β in (4.5.8) and for each of several choices for ε, with one calculation corresponding to the (potential, stable) outer solution $\bar{u}_1 \equiv -1$ of (4.5.4), and with the other calculation corresponding to the outer solution $\bar{u}_2 \equiv +2$ of (4.5.5). In every case the calculation is started with the two nodes $x_1 = 0$ and $x_2 = 1$, with corresponding s_j

taken from one of the exact outer solutions $\bar{u}_1$ or $\bar{u}_2$, as:

(4.5.9) $\qquad s_1 = s_2 = -1 \quad$ for the solution u_1,

or

(4.5.10) $\qquad s_1 = s_2 = +2 \quad$ for the solution u_2.

The numerical results provided by the algorithm are summarized in the table in (4.5.11) for the values of β given in (4.5.8). In each case the calculation is performed for the different values $\varepsilon^2 = 10^{-3}, 10^{-4}, \cdots, 10^{-11}, 10^{-12}$. The algorithm produces accurate solution values for u_1 in the cases $\beta = -2.4$, -1.0, $+0.27$, $+0.28$, and $+0.5$, but the algorithm fails to converge for u_1 in the cases $\beta = -3.0$ and -2.41. For u_2 the algorithm produces accurate solution values in the cases $\beta = +0.28$ and $+0.5$, but the algorithm fails to converge for the other values of β listed in the table in (4.5.11).

(4.5.11)

β	-3.0	-2.41	-2.4	-1.0	-0.27	$+0.28$	$+0.5$
u_1	$-$	$-$	$+$	$+$	$+$	$+$	$+$
u_2	$-$	$-$	$-$	$-$	$-$	$+$	$+$

$+$ = Accurate solution values given by numerical computation.

$-$ = Numerical computation indicates solution fails to exist.

For example, in the case (4.5.9) with $\beta = -2.4$ and with $\varepsilon = 10^{-5}$ ($\varepsilon^2 = 10^{-10}$ in (4.5.1)), the algorithm produces accurate solution values in less than 2 seconds of computer time using six nodes in a boundary layer adjoining the right endpoint $x = 1$, with about 70,000 function calls of the right side of the differential equation. On the other hand, in the case (4.5.10) with $\beta = -2.4$ and with $\varepsilon = 10^{-5}$, the algorithm fails to converge, which is in agreement with the nonexistence of an exact solution satisfying (4.5.5).

When an exact solution u_1 and/or u_2 exists for (4.5.1), the algorithm produces accurate approximate solution values efficiently, and the accuracy is good throughout the interval including within the boundary layer. On the other hand, in each case when an exact solution u_1 and/or u_2 *fails* to exist, then the algorithm automatically gives an indication that a corresponding solution fails to exist. In every case listed in (4.5.11), the computed results are in agreement with the theoretical results of [30].

Summary. An algorithm based on multiple shooting is given for the numerical solution of singularly perturbed boundary value problems with possible boundary layers at the endpoints. The algorithm includes an automatic control and distribution of nodes. A stable initial value solver is used in the outer interval so that one is not restricted to unnecessarily small stepsizes there, while a high-order initial value solver (which in some cases is replaced by a stiff integrator) is used within the boundary layers. The algorithm is tested on several nontrivial problems, including certain problems which might seem not to be well suited for the method. One sees that the algorithm enjoys a certain robustness: for all test problems considered, the algorithm performs successfully as regards both accuracy and efficiency. In particular, the algorithm provides accurate solution values *throughout the entire interval*, including throughout the boundary layers, which is crucial in certain applications in the physical theory of semiconducting devices.

Acknowledgments. This paper is based on my Ph.D. dissertation written under the direction of Professor R. Bulirsch at the Institut für Mathematik der Technischen

Universität München, and I wish to express my appreciation to Professor Bulirsch for his encouragement, stimulation and advice during the prosecution of this research. I also wish to thank Professor Donald Smith for helpful technical discussions during the period 1980-81 while Professor Smith was visiting the Technical University of Munich. The English version of this paper was written during the Spring of 1983 while I was visiting the University of California at San Diego. I wish to thank Professor Smith for making that visit possible and for checking the English of this paper.

REFERENCES

[1] L. ABRAHAMSSON, *Numerical solution of a boundary value problem in semiconductor physics*, Uppsala University Report No. 81, 1979.

[2] L. ABRAHAMSSON, H. B. KELLER AND H. O. KREISS, *Difference approximations for singular perturbations of systems of ordinary differential equations*, Numer. Math., 22 (1974), pp. 367-391.

[3] U. ASCHER, *Solving boundary value problems with a spline-collocation code*, J. Comput. Phys., 34 (1980), pp. 401-413.

[4] A. BILLER, *Ein nichtlineares Verfahren zur Lösung von steifen Anfangswertproblemen*, Technische Universität München, Institut für Mathematik, Diplomarbet, 1979.

[5] C. G. BROYDEN, *A class of methods for solving nonlinear simultaneous equations*, Math. Comp., 19 (1965), pp. 577-593.

[6] R. BULIRSCH, *Die Mehrzielmethode zur numerischen Lösung von nichtlinearen Randwertproblemen und Aufgaben der optimalen Steuerung*, Report der Carl-Cranz-Gesellschaft, 1971.

[7] R. BULIRSCH, J. STOER AND P. DEUFLHARD, *Numerical treatment of nonlinear two-point boundary value problems I*, to appear in Numer. Math., Handbook Series Approximation.

[8] F. H. CHIPMAN, *A-stable Runge-Kutta processes*, BIT, 11 (1971), pp. 384-388.

[9] P. DEUFLHARD, *A modified Newton method for the solution of ill-conditioned systems of nonlinear equations with application in multiple shooting*, Numer. Math., 22 (1974), pp. 289-315.

[10] ——, *A relaxation strategy for the modified Newton method*, Lecture Notes in Optimization and Optimal Control 477, Bulirsch, Oettli and Stoer, eds., Springer, Berlin, 1975, pp. 59-73.

[11] W. H. ENRIGHT, R. BEDET, I. FARKAS AND T. E. HULL, *Test results on initial value methods for non-stiff ordinary differential equations*, Tech. Rep. No. 68, Univ. Toronto, Toronto, Ontario, 1974.

[12] W. H. ENRIGHT, T. E. HULL AND B. LINDBERG, *Comparing numerical methods for stiff systems of ordinary differential equations*, Tech. Rep. No. 69, Univ. Toronto, Toronto, Ontario, 1974.

[13] E. FEHLBERG, *Klassische Runge-Kutta Formeln fünfter und siebenter Ordnung mit Schrittweitenkontrolle*, Computing, 4 (1969), pp. 93-106.

[14] J. E. FLAHERTY AND W. MATHON, *Collocation with polynomial and tension splines for singularly-perturbed boundary value problems*, this Journal, 1 (1980), pp. 260-289.

[15] J. E. FLAHERTY AND R. E. O'MALLEY, *The numerical solution of boundary value problems for stiff differential equations*, Math. Comp., 31 (1977), pp. 66-93.

[16] F. A. HOWES, *Some old and new results on singularly perturbed boundary value problems*, Singular Perturbations and Asymptotics, R. E. Meyer and S. V. Parter, eds., Academic Press, New York, 1980.

[17] P. KAPS AND P. RENTROP, *Generalized Runge-Kutta methods of order four with stepsize control for stiff ordinary differential equations*, Numer. Math., 33 (1979), pp. 55-68.

[18] H. O. KREISS, *Difference methods for stiff ordinary differential equations*, SIAM J. Numer. Anal., 15 (1978), pp. 21-58.

[19] M. R. MAIER AND D. R. SMITH, *Numerical solution of a symmetric one-dimensional diode model*, J. Comput. Phys., 42 (1981), pp. 309-326.

[20] W. L. MIRANKER, *Numerical methods of boundary layer type for stiff systems of differential equations*, Computing, 11 (1973), pp. 221-234.

[21] R. E. O'MALLEY, JR., *Boundary layer methods for ordinary differential equations with small coefficients multiplying the highest derivatives*, Lecture Notes in Mathematics 430, Springer, Berlin, pp. 363-389.

[22] ——, *Introduction to Singular Perturbations*, Academic Press, New York, 1974.

[23] R. E. O'MALLEY, JR. and J. E. FLAHERTY, *Analytical and numerical methods for nonlinear singular singularly-perturbed initial value problems*, SIAM J. Appl. Math., 38 (1980), pp. 225, 248.

[24] C. E. PEARSON, *On a differential equation of boundary layer type*, J. Math. Phys., 47 (1968), pp. 134-154.

[25] ——, *On nonlinear differential equations of boundary layer type*, J. Math. Phys., 47 (1968), pp. 351-358.

[26] P. RENTROP, *Numerical solution of a transistor boundary value problem*, Technische Universität München, Institut für Mathematik, TUM-7628, 1976.

[27] R. SEYDEL, *Remarks on the scaling and the reliability of stepsize controls in Runge-Kutta-Fehlberg methods*, Technische Universität München, TUM-M8011, 1980.
[28] B. SIGFRIDSSON AND J. L. LINDSTRÖM, *Electrical properties of electron-irradiated n-type silicon*, J. Appl. Phys., 47 (1976), pp. 4611-4620.
[29] D. R. SMITH, *On a singularly perturbed boundary value problem arising in the physical theory of semiconductors*, Technische Universität München, Institut für Mathematik, TUM-M8021, 1980.
[30] ——, *Singular Perturbation Theory*, Cambridge Univ. Press, Cambridge, 1985.
[31] J. STOER AND R. BULIRSCH, *Introduction to Numerical Analysis*, Springer, Berlin, 1980.
[32] O. P. VAN DRIEL, *private communication.*
[33] A. B. VASIL'EVA AND V. G. STEL'MAKH, *Singularly disturbed systems of the theory of semiconductor devices*, USSR Comp. Math. and Phys., 17 (1977), pp. 48-58.
[34] D. S. WATKINS, *Determining initial values for stiff systems of ordinary differential equations*, SIAM J. Numer. Anal., 18 (1981), pp. 18-20.
[35] J. H. WILKINSON AND CH. REINSCH, *Linear Algebra, Handbook for Automatic Computation* 2, Springer, Berlin, 1971.

SIAM J. SCI. STAT. COMPUT.
Vol. 7, No. 2, April 1986

006

SOLVING EIGENVALUE AND SINGULAR VALUE PROBLEMS ON AN UNDERSIZED SYSTOLIC ARRAY*

ROBERT SCHREIBER†

Abstract. Systolic architectures due to Brent, Luk and Van Loan are today the most promising method for computing the symmetric eigenvalue and singular value decompositions in real time. These systolic arrays, however, are only able to decompose matrices of a given fixed size. Here we present two modified algorithms and a modified array that do not have this disadvantage. The results of a numerical experiment show that a combination of one of our new algorithms and the modified array can decompose matrices of arbitrary size with little or no loss of efficiency.

Key words. eigenvalue problems, singular value decomposition, systolic computation, parallel computation

1. Introduction. Systolic arrays are of significant and growing importance in numerical computing [12], especially in matrix computation and its applications in digital signal processing [13]. There is now considerable interest in systolic computation of the singular value decomposition [2], [4], [6], [10] and the symmetric eigenvalue problem [1], [8].

To date, the most powerful systolic array for the eigenvalues of a symmetric $n \times n$ matrix is a square $n/2 \times n/2$ array due to Brent and Luk. This array implements a certain cyclic Jacobi method. It takes $O(n)$ time to perform a sweep of the method, and $O(\log n)$ sweeps for the method to converge [1].

Brent and Luk have also invented a closely related $(n/2)$-processor linear array for computing the singular value decomposition (SVD) of an $m \times n$ matrix A. An SVD of A is a factorization $A = U\Sigma V^T$, where V is orthogonal, Σ is nonnegative and diagonal, and U is $m \times n$ with orthonormal columns. This array implements a cyclic Hestenes algorithm that, in real arithmetic, is an exact analogue of their Jacobi method applied to the eigenproblem for A^TA. The array requires $O(mn)$ time for a sweep, and $O(\log n)$ sweeps for convergence [2].

A new array, much like the eigenvalue array, is reported by Brent, Luk and Van Loan to be capable of finding the SVD in time $O(m + n \log n)$ [3].

The purpose of this paper is to consider an important, indeed an essential problem concerning the practical use of these arrays. How, with an array of a given fixed size, can we decompose matrices of arbitrarily large size?

2. Systolic arrays for the Jacobi and Hestenes methods. We shall concentrate on Hestenes' method for the SVD. Starting with the given matrix A, we build an orthogonal matrix V such that AV has orthogonal columns. Thus

$$AV = U\Sigma,$$

where U has orthonormal columns and Σ is nonnegative and diagonal. An SVD is given by $A = U\Sigma V^T$.

To construct V, we take $A^{(0)} = A$, and iterate

$$A^{(i+1)} = A^{(i)}Q^{(i)}, \qquad i = 0, 1, \cdots,$$

* Received by the editors June 14, 1983, and in final revised form November 12, 1984. A preliminary version of this paper appeared in *Real-Time Signal Processing* VI, SPIE vol. 431 (1983), pp. 72–77. This research was partially supported by the U.S. Office of Naval Research under contract N00014-82-K-0703.

† Guiltech Research Company, 255 San Geronimo Way, Sunnyvale, California 94086.

with $Q^{(i)}$ orthogonal, until some matrix $A^{(i)}$ has orthogonal columns. $Q^{(i)}$ is chosen to be a product of $n(n-1)/2$ plane rotations

$$Q^{(i)} = \prod_{j=1}^{n(n-1)/2} Q_j^{(i)}.$$

Every possible pair (r, s), $1 \leqq r < s \leqq n$, is associated with one of the rotations $Q_j^{(i)}$ (the association is independent of i) in this way: the rotation $Q_j^{(i)}$ is chosen to make columns r and s of

$$A^{(i)}\left(\prod_{k=1}^{j} Q_k^{(i)}\right)$$

orthogonal. The process of going from $A^{(i)}$ to $A^{(i+1)}$ is called a "sweep." Every permutation of the set of pairs corresponds to a different cyclic Hestenes method.

The correspondence with the Jacobi method is this. The sequence $A^{(i)T}A^{(i)}$ converges to the diagonal matrix Σ^2 of eigenvalues of A^TA. Moreover,

$$A^{(i+1)T}A^{(i+1)} = Q^{(i)T}(A^{(i)T}A^{(i)})Q^{(i)},$$

where $Q^{(i)}$ is the product of $n(n-1)/2$ of Jacobi rotations that zero, in some cyclic order, the off-diagonal elements of $A^{(i)T}A^{(i)}$.

The permutation chosen by Brent and Luk allows the rotations to be applied in parallel in groups of $n/2$. Their permutation consists of $n-1$ groups of $n/2$ pairs such that, in each group, every column occurs once. Thus, the $n/2$ rotations corresponding to a pair-group commute. They can be applied in any order or, in fact, in parallel.

The SVD array is shown in Fig. 1. There are $n/2$ processors. Each processor holds two matrix columns. Initially processor i holds column $2i-1$ in its "left memory" and column $2i$ in its "right memory."

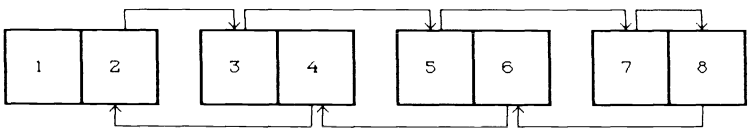


FIG. 1. *The* SVD *array*; $n = 8$.

In each cycle, each processor computes and applies to its two columns a plane rotation that makes them orthogonal. Next, using the connections shown in Fig. 1, columns move to neighboring processors. This produces a new set of $n/2$ column-pairs.

After $n-1$ cycles, $n(n-1)/2$ pairs of columns have been orthogonalized. It can be shown (by a parity argument) that no pair occurs twice during this time. Thus, every pair is orthogonalized exactly once. We call the process of orthogonalizing all pairs, in this parallel order, an A-sweep.

A diagram (given in [2] originally) showing the movement of columns through the array, very important in the considerations to follow, is given in Fig. 2.

3. Solving larger problems. We now consider the problem of finding an SVD when A has n columns, the array has p processors, and $n > 2p$.

The usual approach to this problem is to imagine that a "virtual" array, large enough to solve the problem (having $\lceil n/2 \rceil$ or more processors), is to be simulated by the given small physical array. Moreover, the simulation must be efficient. The array should not spend a large amount of time loading and unloading data.

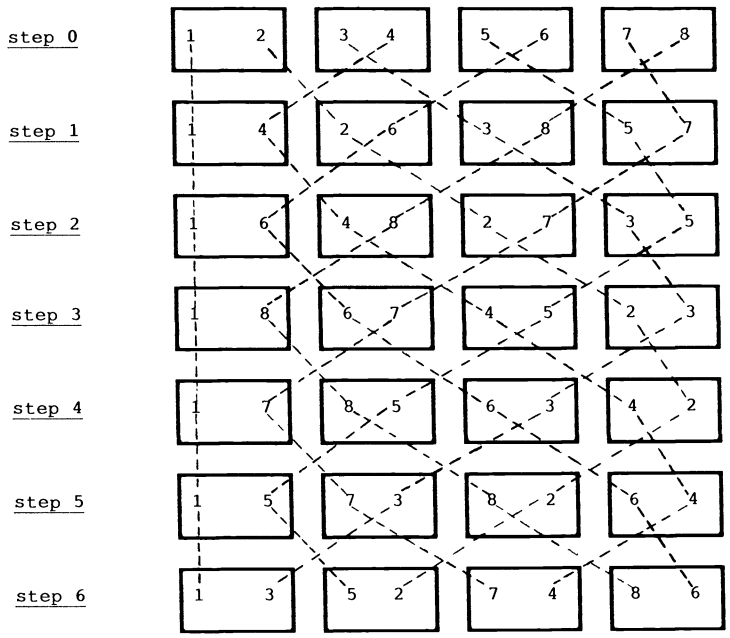


FIG. 2. *Flow of data in the* SVD *array*; $n = 8$.

For some arrays, this simulation is trivial. One finds a subarray of the virtual array, of the same size as the physical array, for which all the input streams are known. Clearly the action of such a subarray can be carried out and its outputs stored. These outputs then become the inputs to other subarrays. This process continues until, subarray by subarray, the computation of the entire virtual array has been performed. If this technique is possible, we say that the array is "decomposable." The various matrix multiply arrays [7], the array of Gentleman and Kung for QR factorization [5] and the array of Schreiber and Kuekes for solving triangular systems [9] are good examples of decomposable arrays.

Some arrays are indecomposable: the Kung-Leiserson band-matrix LU factorization array, for example [7].

Consider the 4×4 matrix multiplication array shown in Fig. 3. It computes $C + AB$ where C and B are $4 \times m$ and A is 4×4. Suppose we have a 2×2 array of the same type. With it, $C + AB$ can be computed using a block algorithm. Partition A, B and C so that

$$A = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix}, \qquad B = \begin{bmatrix} B_1 \\ B_2 \end{bmatrix}, \qquad C = \begin{bmatrix} C_1 \\ C_2 \end{bmatrix},$$

where the blocks are 2×2 or $2 \times m$. We use the array to carry out these operations:

1. $C_1 := A_{11}B_1$;
2. $C_2 := A_{21}B_1$;
3. $C_1 := C_1 + A_{12}B_2$;
4. $C_2 := C_2 + A_{22}B_2$.

An equivalent viewpoint is that we use the array to emulate (that is, perform the work of) four 2×2 sections of the 4×4 array; the sections are shown in dotted outline in Fig. 3. Note that the input data for each section is known at the time the section is emulated by the 2×2 array, either because that data is input data or because it is

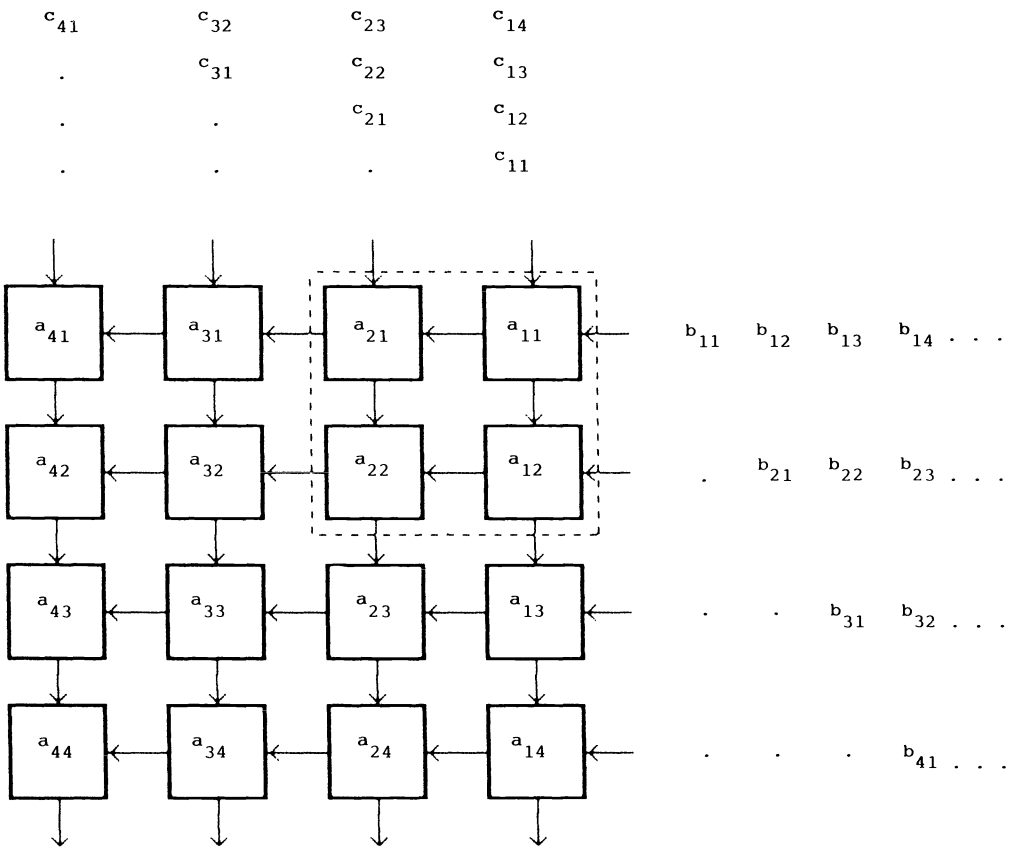


FIG. 3. *A matrix multiplication array*: $C \leftarrow C + AB$.

output from another part of the array whose action has already been emulated. This sort of decomposition of an array can work only when there are no cycles in the flow of data in the array. This is true of the decomposable arrays mentioned above.

The SVD array of § 2 is indecomposable. Consider Fig. 2. Suppose a two-processor array is available. It cannot efficiently simulate the four-processor array because there does not exist a two-processor segment of Fig. 2 for which only known data enters. If this diagram is cut by a vertical line, data flows across the line in both directions, every cycle. The data cannot be known if only the computations on one side of the line have been performed.

Here we shall present a solution to this problem. The idea is to have a given p-processor array simulate a pq-processor "superarray" which is not of the Brent–Luk type. Moreover, the superarray is decomposable. In its space-time dataflow graph, the processors occur in groups of p. For long periods of either p or $2p-1$ cycles, no data flows between groups. Thus, the physical array can efficiently carry out the computation of the superarray, group-by-group.

We give two such superarrays. The first implements a Hestenes method in which a "sweep" corresponds to a permutation of a multiset of off-diagonal pairs. There is some redundancy: some pairs are generated and orthogonalized several times. The second implements a cyclic Hestenes method with a permutation different from the one used in an A-sweep. For this method, a minor change must be made to the array.

We have compared these new sweeps to the A-sweep. These experiments indicate that the first superarray is about 20–60% less efficient than the Brent–Luk array, while the second superarray is virtually equal to the Brent–Luk array in efficiency.

3.1. Method A. This method is easiest to explain in terms of an example. Suppose we have a 4 processor array. Suppose there are 16 columns in A. We proceed as follows:

1. Load columns 1–8 and perform an A-sweep;
2. Load columns 9–16 and perform an A-sweep;
3. Load columns 1–4, 13–16; perform an A-sweep;
4. Load columns 5–8, 9–12; perform an A-sweep;
5. Load columns 1–4, 9–12; perform an A-sweep;
6. Load columns 13–16, 5–8; perform an A-sweep.

Steps 1–6 together constitute an A-supersweep. During an A-supersweep, every column pair is orthogonalized. Some are orthogonalized more than once.

To describe the general case, suppose there is a p-processor array, and $n = 2pq$ (pad A with zero columns, if necessary, so that $2p$ divides n). Imagine that the matrix A consists of $2q$ supercolumns: supercolumn A_i consists of columns

$$p(i-1)+1, \cdots, pi.$$

Now consider a q-superprocessor virtual superarray. Each superprocessor holds two supercolumns (one in each of its left and right memories). In one supercycle the superprocessors each perform a single A-sweep over the $2p$ columns in their memory.

(Obviously we can simulate a supercycle of a superprocessor using one p-processor Brent–Luk array and $2p-1$ cycles of time. Moreover, we can be loading the data for the next supercycle and unloading the data from the preceding supercycle at the same time as we process the data for the current supercycle.)

Initially, supercolumns A_1 and A_2 are in superprocessor 1, A_3 and A_4 in superprocessor 2, etc.

Between supercycles, the supercolumns move to neighboring superprocessors. The scheme for moving supercolumns is precisely the same as the scheme for moving ordinary columns in a q-processor Brent–Luk array.

After $2q-1$ supercycles, we have performed an A-sweep on every pair of supercolumns exactly once. Together these $2q-1$ supercycles constitute an A-supersweep. During an A-supersweep, every pair of columns of A is orthogonalized. If two columns are in different supercolumns, then they are orthogonalized once, during the supercycle in which their containing supercolumns occupy the same superprocessor. If they are in the same supercolumn, then they are orthogonalized $2q-1$ times.

In units of cycles, the time for an A-supersweep, T_{AS}, is

$$\begin{aligned} T_{AS} &= (2q-1) \text{ supercycles} * (2p-1) \text{ cycles/supercycle} \\ &= (2q-1)(2p-1). \end{aligned}$$

(Of course, the simulation by a p-processor array takes q times this time.) The time for an A-sweep over n columns, T_A, is

$$T_A = n-1 = 2pq-1.$$

Thus, the A-supersweep takes longer; the ratio of times satisfies

$$\frac{9}{7} \leqq \frac{T_{AS}}{T_A} < 2.$$

(The lower bound arises in the simplest nontrivial case $p = q = 2$.)

There is little theoretical basis for comparing the effectiveness of A-supersweeps and A-sweeps in reducing the nonorthogonality of the columns of A. We have therefore performed an experiment. A set of square matrices A whose elements were random and uniformly distributed in $[-1, 1]$ was generated. Both A-supersweeps and A-sweeps were used until the sum-of-squares of the off-diagonal elements of A^TA was reduced to 10^{-12} times its initial value. We show the results in Table 1. The number of test matrices, the average number of sweeps, the largest number for any test matrix, and the relative time

$$\rho \equiv T_{AS} * \text{average-sweeps } (AS) / T_A * \text{average-sweeps } (A)$$

are shown.

Evidently one A-supersweep is more effective in reducing nonorthogonality than one A-sweep. This is not surprising, since more orthogonalizations are performed. Their cost-effectiveness, however, is roughly 20–60% less.

TABLE 1
Comparison of A-sweeps and A-supersweeps.

				Average		Maximum		
p	q	n	Trials	A super	A	A super	A	ρ
2	2	8	320	3.98	4.33	5	5	1.18
2	4	16	160	5.10	5.38	6	7	1.33
2	8	32	80	6.18	6.29	7	7	1.43
4	2	16	160	4.80	5.40	5	6	1.24
4	4	32	80	5.99	6.31	7	7	1.50
4	8	64	20	7.05	7.55	8	8	1.57
8	2	32	80	5.25	6.28	6	7	1.21
8	4	64	10	6.60	7.60	7	8	1.45
16	2	64	20	6.00	7.30	6	8	1.21

In order to gauge the reliability of the statistics generated by this experiment, we also measured the standard deviations of the sampled data. In all cases, the standard deviations were less than 0.5. For the samples of size 80 or more, the standard errors of the means are no more than 0.06, so these statistics are quite reliable. For the samples of sizes 20 and 10, these data may be in error by as much as 10%.

3.2. Method B. Method A suffers some loss of speed, because in an A-supersweep some column-pairs are orthogonalized many times. By making a small modification to the Brent–Luk array and using the new array as our basic tool, we can simulate a new supersweep, called an AB-supersweep, during which every column-pair is orthogonalized exactly once.

Figure 3 shows the modified array. The connection from processor 1 to processor p is new. Note that a ring connected set of processors can easily simulate this structure. This array is still able to perform A-sweeps over sets of $2p$ columns. But it can also perform a second type of sweep, which we call an "AB-sweep," and which we now describe.

In an AB-sweep, a pair (A, B) of supercolumns, each consisting of p columns, is loaded into the array. During the sweep, all pairs (a, b) $a \in A$, $b \in B$ are orthogonalized exactly once. But no pairs from $A \times A$ or $B \times B$ are orthogonalized.

To implement an AB-sweep, place the columns of A in the p left memories and the columns of B in the p right memories of the processors. (The set of left (respectively

right) processor memories is the superprocessor's left (respectively right) memory, rather than the memories of the leftmost (respectively rightmost) $p/2$ processors.) Processors do precisely what they did before: orthogonalize their two columns. Between cycles, A remains stationary while B rotates one position, using the connections shown as solid lines in Fig. 4.

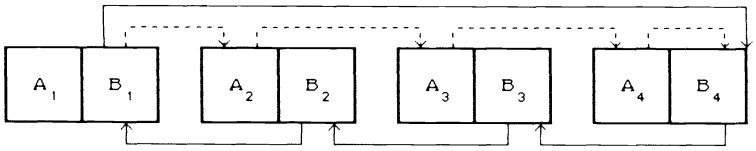


FIG. 4. *The modified* SVD *array*; $n = 8$.

An AB-supersweep is as follows. Again we work with $2q$ supercolumns of p columns each. The initial configuration is as for an A-supersweep. During the first supercycle, which takes $2p-1$ cycles, every superprocessor performs an A-sweep on the $2p$ columns in its memory. On subsequent supercycles, all superprocessors perform AB-sweeps, where the sets A and B are the two supercolumns in its memory. Between supercycles, supercolumns move as before.

It is easy to see that in an AB-supersweep, every column pair is orthogonalized once. Thus this scheme implements a true cyclic Hestenes method. The permutation differs, nevertheless, from the permutation used in an A-sweep.

Again, we have compared the new scheme to the A-sweep by an experiment. The experiment setup was precisely the same as for the previous experiment. The results are shown in Table 2.

TABLE 2
Comparison of A-sweeps and AB-supersweeps.

				Averages		Maxima	
p	q	n	Trials	AB super	A	AB super	A
2	2	8	320	4.32	4.33	5	5
2	4	16	160	5.35	5.38	6	7
2	8	32	80	6.36	6.29	7	7
4	2	16	160	5.36	5.40	6	6
4	4	32	80	6.18	6.31	7	7
4	8	64	20	7.50	7.55	8	8
8	2	32	80	6.13	6.28	7	7
8	4	64	10	7.10	7.60	8	8
16	2	64	20	7.00	7.30	7	8

Evidently, AB-supersweeps are as effective as A-sweeps. The standard deviations of the number of AB-supersweeps needed were also all less than 0.5.

3.3. Earlier work. Another scheme for solving problems with an undersized array was proposed in [3], a paper that deals with a square SVD array. The proposal is to use a block method, in which the SVDs of diagonal blocks are computed in the given array.

Applied to the linear SVD array, this idea is much like our A-supersweep scheme, except that a superprocessor iterates to convergence instead of performing only one A-sweep.

TABLE 3
Comparison of block-Jacobi A-sweeps and A-supersweeps.

p	q	n	Trials	Ratio
2	2	8	40	2.4
2	4	16	40	2.7
2	8	32	10	3.1
4	2	16	40	2.9
4	4	32	10	4.2
4	8	64	5	4.3
8	2	32	10	3.7
8	4	64	5	5.2
16	2	64	5	3.7

The extra time needed for this convergence leads to inefficiency in this scheme. The data in Table 3 show this. These give the result of a numerical experiment; the setup was the same as for previous experiments. For several values of p and q we show the ratio of the number of operations used by the block method to the number used by the AB-supersweep method discussed earlier. For this experiment, we stopped a superprocessor, which was working on the $n \times 2p$ matrix B, from further iteration when the sum of the squares of the off-diagonal elements of B^TB was less than 10^{-12} times the sum of the squares of the diagonal elements of B^TB.

Thus, the present schemes require fewer computations than the block method of [3]. On the other hand, they may require more input/output from the array. The choice between them will turn on factors such as the relative speed of computation and input/output that depend on how the array is implemented.

Clearly there is a family of methods of this type parameterized by the number of sweeps over each block. Our experiments illuminate two extreme cases.

4. The eigenvalue array. In this section we show how the two ideas for problem decomposition in § 3 can be used to solve large eigenvalue problems on the square array of [1]. This array solves the symmetric eigenvalue problem in time $O(n \log n)$ and is the fastest array known for that problem. The same ideas also apply, in exactly the same way, to the SVD array of [3].

The eigenvalue array is a $p \times p$ array that holds a $2p \times 2p$ symmetric matrix. Each processor p_{ij} holds a 2×2 submatrix b_{ij}: initially,

$$b_{ij} = \begin{bmatrix} a_{2i-1,2j-1} & a_{2i-1,2j} \\ a_{2i,2j-1} & a_{2i,2} \end{bmatrix}.$$

At each cycle, each diagonal processor p_{ii} generates a plane rotation r_i such that $r_i b_{ii} r_i^T$ is diagonal. The rotations are then sent from the diagonal processor to all processors in the same row and the same column. The off-diagonal processor p_{ij} (if $i<j$) on receiving rotations r_i and r_j, computes a new block $b_{ij} = r_i b_{ij} r_i^T$. After the rotations have been applied by the off-diagonal processors, columns and rows are interchanged. Adjacent processors exchange data with their neighbors to the right and left to permute the matrix columns as in the SVD array (Fig. 1). Then processors exchange data with the processors above and below to permute the matrix rows in the same way. After $2p-1$ cycles, all off-diagonal elements have been annihilated once: this is one sweep.

It is not necessary to broadcast rotations to an entire row or column of the array. Instead, rotations move through one processor per cycle. Thus, if rotations are generated at time $t=0$, they are applied at time t by the processors $p_{i,i\pm t}$ in diagonal t. Likewise,

the process of exchanging data occurs in a wave. It begins with exchanges between the diagonal processors and those in diagonals ± 1 at time 2. At time $t+1$, $0 < t < p-1$, processors of diagonals $\pm t$ and $\pm(t+1)$ exchange data. The second set of rotations is generated at the diagonal at time 3, the next at time 6, etc. Thus, the last rotations are generated at time $6p-6$, and the whole sweep is finished at time $7p-7$.

The decomposition of this array works as follows. Partition the given $n \times n$ matrix A as

$$\begin{matrix} A_{11} & A_{12} & \cdots & A_{1,2q} \\ A_{21} & A_{22} & \cdots & A_{2,2q} \\ \vdots & & & \vdots \\ A_{2q,1} & A_{2q,2} & \cdots & A_{2q,2q} \end{matrix}.$$

We imagine a virtual $q \times q$ superarray with superprocessors P_{ij} that hold a 2×2 block matrix B_{ij} having $p \times p$ blocks: initially,

$$B_{ij} = \begin{bmatrix} A_{2i-1,2j-1} & A_{2i-1,2j} \\ A_{2i,2j-1} & A_{2i,2j} \end{bmatrix}.$$

At each supercycle, each diagonal superprocessor P_{ii} generates an orthogonal product of plane rotations R_i by performing one sweep of the Jacobi method with the parallel order as described in the last paragraph. The rotations are then sent from the diagonal

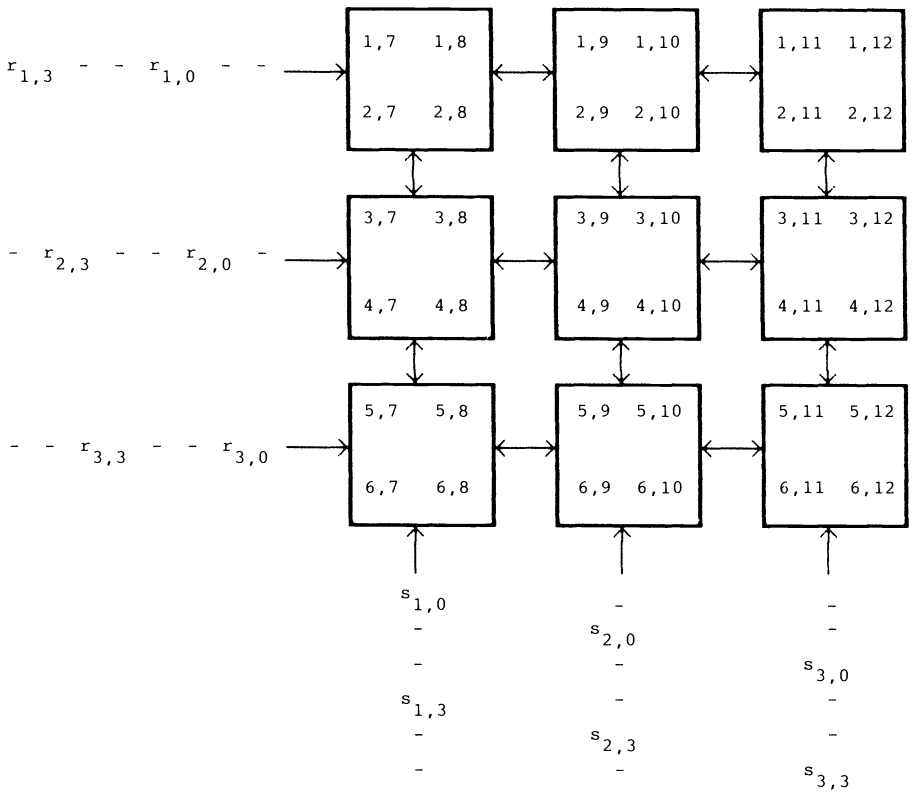


FIG. 5. *Operation of an off-diagonal superprocessor.*

superprocessor to all superprocessors in the same row and the same column. The off-diagonal superprocessor P_{ij} (if $i<j$), on receiving rotation-products R_i and R_j, computes a new block $B_{ij} = R_i B_{ij} R_j^T$. After the rotations have been applied by the off-diagonal processors, block columns and rows are interchanged. Adjacent superprocessors exchange data with their neighbors to the right and left to permute the block columns as in the SVD (Fig. 1). Then superprocessors exchange data with the superprocessors above and below to permute the block rows in the same way.

A single square $p \times p$ eigenvalue array can emulate this superarray efficiently. Indeed, the diagonal superprocessor is a $p \times p$ eigenvalue array. We assume that the rotations generated by this array flow out at the edges and can be stored for later use. We need only show that the off-diagonal superprocessors can be emulated. For this to work, the p diagonal processors must change their roles, becoming ordinary off-diagonal processors. We assume an off-diagonal block B_{ij} is loaded into the array. The rotations that make up R_i and R_j are sent into the array at its left and bottom edges. The individual plane rotations are sent into the array at the same relative places and times as when they left the array after being generated. They flow through the array and are applied to the matrix elements as in the eigenvalue array. Interchange of rows and columns begins at the lower left corner of the array and moves in a wave toward the upper right corner.

Figure 5 illustrates this. The orthogonal matrix R_i is a product of plane rotations that we denote by $r_{k,t}$, where $r_{k,t}$ is the rotation generated by processor p_{kk} at time $3t$ while the array was emulating the diagonal superprocessor P_{ii}. We denote by $s_{k,t}$ the constituent rotation of R_j that was generated by p_{kk} at time $3t$ while the array was emulating the diagonal superprocessor P_{jj}. Thus, $1 \leqq k \leqq p$ and $0 \leqq t \leqq 2p-2$.

Acknowledgment. This work was done in Stockholm while I was a guest of the Royal Institute of Technology, Department of Numerical Analysis and Computer Science, and of Uppsala University, Department of Computer Science. I thank Charles Van Loan, Erik Tidén and Björn Lisper for their comments.

REFERENCES

[1] R. P. BRENT AND F. T. LUK, *The solution of singular-value and symmetric eigenvalue problems on multiprocessor arrays*, this Journal, 6 (1985), pp. 69–84.

[2] ———, *A systolic architecture for the singular value decomposition*, Technical Report TR-CS-82-09, Dept. Computer Science, The Australian National University, Canberra, 1982.

[3] R. P. BRENT, F. T. LUK AND C. VAN LOAN, *Computation of the singular value decomposition using mesh-connected processors*, Technical Report TR 82-528, Dept. of Computer Science, Cornell Univ., Ithaca, NY, 1983; J. VLSI and Computer Systems, to appear.

[4] ALAN M. FINN, FRANKLIN T. LUK AND CHRISTOPHER POTTLE, *Systolic array computation of the singular value decomposition*, in Real-Time Signal Processing V, Joel Trimble, ed., SPIE, 341 (1982), pp. 35–43.

[5] W. M. GENTLEMAN AND H. T. KUNG, *Matrix triangularization by systolic array*, in Real-Time Signal Processing IV, Tian F. Tao, ed., SPIE, 298 (1981), pp. 19–26.

[6] DON E. HELLER AND ILSE C. F. IPSEN, *Systolic networks for orthogonal decompositions*, this Journal, 4 (1983), pp. 261–269.

[7] H. T. KUNG AND C. E. LEISERSON, *Systolic arrays (for VLSI)*, in Sparse Matrix Proceedings, I. S. Duff and G. W. Stewart, eds., Society for Industrial and Applied Mathematics, Philadelphia, PA, 1978, pp. 256–282. A slightly different version appears in § 8.3 of Introduction to VLSI Systems by C. A. Mead and L. A. Conway, Addison-Wesley, Reading, MA, 1980.

[8] ROBERT SCHREIBER, *Systolic arrays for eigenvalue computation*, in Real-Time Signal Processing V, Joel Trimble, ed., SPIE, 341 (1982), pp. 27–34.

[9] ROBERT SCHREIBER AND PHILIP J. KUEKES, *Systolic linear algebra machines in digital signal processing*, in VLSI and Modern Signal Processing, S. Y. Kung, H. Whitehouse and T. Kailath, eds., Prentice-Hall, Englewood Cliffs, NJ, 1984, pp. 389–405.

[10] R. SCHREIBER, *A systolic architecture for singular value decomposition*, in Proc. 1er Colloque International sur les Méthodes Vectorielles et Paralèles en Calcul Scientifique, Electricité de France Bulletin de la Direction des Etudes at Recherches, Serie C, No. 1, 1983, pp. 143–148.

[11] ———, *On the systolic array of Brent, Luk and Van Loan*, in Real-Time Signal Processing VI, Keith Bromley, ed., SPIE, 431 (1983), pp. 72–77.

[12] ———, *Systolic arrays: high performance parallel machines for matrix computation*, in Elliptic Problem Solvers II, Garrett Birkhoff and Arthur Schoenstadt, eds., Academic Press, New York, 1984, pp. 187–194.

[13] JEFFREY M. SPEISER AND HARPER J. WHITEHOUSE, *Parallel processing algorithms and architectures for real-time signal processing*, in Real-Time Signal Processing IV, Tian F. Tao, ed., SPIE, 298 (1981), pp. 2–9.

SIAM J. SCI. STAT. COMPUT.
Vol. 7, No. 2, April 1986

007

A ROTATION METHOD FOR COMPUTING THE QR-DECOMPOSITION*

FRANKLIN T. LUK†

Abstract. A parallel method for computing the QR-decomposition of an $n \times n$ matrix is proposed. It requires $O(n^2)$ processors and $O(n)$ units of time. The method can be extended to handle an $m \times n$ matrix ($m \geqq n$). The requirements then become $O(n^2)$ processors and $O(m)$ time.

Key words. QR-decomposition, plane rotations, systolic arrays, real-time computation, VLSI

AMS(MOS) subject classifications. 65F30-68A10

1. Introduction.

Let $A \in R^{n \times n}$ and

$$A = QR \tag{1.1}$$

(Q orthogonal, R upper triangular) be its QR-decomposition (QRD). The sequential computation of this decomposition requires time $O(n^3)$. Often, the QRD of $A \in R^{m \times n}$ ($m \geqq n$) is desired; the required time becomes $O(mn^2)$. For real-time signal processing (cf. Bromley and Speiser [6]) fast parallel algorithms are needed and various methods [1], [2], [8], [9], [10], [12] (most of which are applicable only to square matrices) have been proposed in the literature. Ahmed, Delosme and Morf [1], Bojanczyk, Brent and Kung [2] and Gentleman and Kung [8] all use Givens rotations and a triangular array of $O(n^2)$ processors. Both [1] and [2] store Q (in product form) in the array and propagate R, whereas [8] stores R and propagates Q. The decomposition is computable in time $O(n)$. The technique in [8] can be applied to an $m \times n$ matrix, and it will use time $O(m)$. Heller-Ipsen [9] and Johnsson [10] both consider a banded matrix A, say with bandwidth w. Based on the Givens rotations, the technique of Heller and Ipsen requires time $O(n)$ and a rectangular array of wq processors, where q equals the number of subdiagonals of A. Johnsson discusses a parallel implementation of the Householder transformations. He uses w processors and $O(nw)$ units of time. Sameh [12] considers an $m \times n$ matrix and a ring of p processors. He describes procedures based on the Givens and the Householder transformations; his algorithms require time $O(mn^2/p)$.

The parallel algorithms of Brent, Luk and Van Loan [4], [5] for computing the ordinary and the generalized singular value decompositions may require a preliminary QRD step. However, the mesh-connected multiprocessor arrays in [4], [5] are very different from the QR-arrays in [1], [2], [8], [9], [10], [12] and the interfacing of different arrays can be a serious problem. In this paper we present a rotation method that computes the QRD using a mesh-connected processor array. Our idea is to determine the QRD of an $n \times n$ matrix by computing in parallel $\lfloor n/2 \rfloor$ two-by-two QRDs. This strategy of decomposing an n-by-n problem into $\lfloor n/2 \rfloor$ two-by-two subproblems has been used successfully by Brent and Luk [3] for the symmetric eigenvalue decomposition, by Brent, Luk and Van Loan [4] for the singular value decomposition, and by Stewart [13] for the Schur decomposition. The strategy in [3], [4] is to divide an $n \times n$ matrix into blocks of 2×2 submatrices and to assign one

* Received by the editors June 11, 1984, and in final form March 8, 1985. This work was supported in part by the National Science Foundation under grant MCS-8213718 and by the Office of Naval Research under contract N00014-85-K-0074.

† School of Electrical Engineering, Cornell University, Ithaca, New York 14853.

processor to each block, resulting in a mesh-connected grid of $(\lceil n/2 \rceil)^2$ processors. The Schur decomposition array in [13] consists of two computational networks, each one quite similar to the multiprocessor array in [3], [4]. A total of approximately $n^2/2$ processors are needed (see § 2 for a precise count). However, if we assign two nodes (one from each network) to a processor, we can simulate this complex array using a mesh-connected grid of processors (cf. O'Leary-Stewart [11]). Our QRD algorithm requires the Schur decomposition array, hence $O(n^2)$ processors.

In § 2 we present our new algorithm and prove that it always converges after $2n$ time steps. The algorithm is extended to handle an $m \times n$ $(m \geqq n)$ matrix in § 3. The requirements become $O(n^2)$ processors and $O(m)$ time.

2. The algorithm. We parallelize the computations by simultaneously triangularizing $\lfloor n/2 \rfloor$ two-by-two submatrices of $A \in R^{n \times n}$. Consider the basic transformation: a QRD with column pivoting is computed of the 2×2 matrix

$$B = \begin{pmatrix} a_{ii} & a_{ij} \\ a_{ji} & a_{jj} \end{pmatrix}.$$

We get

$$\hat{B} \equiv JB\Pi = \begin{pmatrix} \hat{a}_{ii} & \hat{a}_{ij} \\ 0 & \hat{a}_{jj} \end{pmatrix},$$

where

$$J = \begin{pmatrix} c & s \\ -s & c \end{pmatrix} \quad \text{and} \quad \Pi = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}.$$

The rotation parameters are calculated using the formulae:

$$h = \sqrt{a_{ij}^2 + a_{jj}^2}, \quad c = a_{ij}/h, \quad s = a_{jj}/h.$$

The full transformation on A is defined by

$$T_{ij} : A \to \hat{A} \equiv J_{ij} A \Pi_{ij}, \tag{2.1}$$

where J_{ij} denotes a plane rotation and Π_{ij} a permutation, both in the (i, j)-plane. The transformation T_{ij} will annihilate the (j, i)-element.

Our new algorithm uses an "odd-even" ordering of Stewart [13]. His ordering is amply illustrated by the $n = 8$ case:

$$(i, j) = (1, 2), (3, 4), (5, 6), (7, 8), (2, 3), (4, 5), (6, 7).$$

Many results in this section come from [13] and an interested reader should consult that fine paper. Besides a parallel implementation, the "odd-even" ordering preserves the triangular structure of a given matrix (see Lemma 5). A bonus (unimportant here) of the ordering is that the sum of squares of the strictly lower triangular elements will decrease. More precisely, define

$$\sigma(A) \equiv \sum_{p>q} |a_{pq}|^2.$$

The transformation $T_{i,i+1}$ will produce a matrix $\hat{A}$ satisfying

$$\sigma(\hat{A}) = \sigma(A) - |a_{i+1,i}|^2.$$

Our procedure thus shares with other Jacobi methods (cf. [3], [4], [13]) the property that it drives the matrix to the desired form. An important difference is that our

algorithm is finite: it converges after $2n$ time steps (see Theorem 2), where one time step is defined as the time required to do a transformation T_{ij}. The orthogonal matrix Q is readily computable through accumulating the plane rotations. We present our new algorithm.

ALGORITHM QRD.
$Q \leftarrow I$;
for $t = 1, 2, \cdots, n$ do
for $i = 1, 3, \cdots$ (i odd), $2, 4, \cdots$ (i even) do
begin
$A \leftarrow J_{i,i+1} A \Pi_{i,i+1}$;
$Q \leftarrow Q J_{i,i+1}^T$
end.

The column pivotings are essential. For example, Algorithm QRD without pivoting stagnates on a matrix with a zero subdiagonal:

$$A = \begin{pmatrix} \times & \times & \times & \times \\ 0 & \times & \times & \times \\ \times & 0 & \times & \times \\ \times & \times & 0 & \times \end{pmatrix}.$$

Interestingly, the pivotings nullify one another after $2n$ steps.

LEMMA 1. *All columns of A return to their original positions after 2n time steps.*

Proof. A column vector moves forwards (backwards) after each time step until it gets to the first (last) position, where it stays for one step. It then reverses direction and moves again. The required number of steps for all columns to return to their initial positions equals the sum of $2n-2$ (for moving) and 2 (for the rest periods at the two ends). □

Fig. 1 (cf. [13, Fig. 4.1]) exhibits these interchanges for the case $n = 6$. The numbers to the side are time steps, and the six numbers following them mark the positions of the original columns. A dash between two elements indicates an interchange that will take place at the next time step. We shall prove that the matrix is triangularized after at most $2n-2$ time steps. First, we introduce a notation and define a property indicating that a column vector is in "upper triangular" form.

Notation. Let $A^{(0)} = A$ and denote by $A^{(t)}$ the matrix A *after* time step t. Set also

$$A^{(t)} \equiv (a_1^{(t)}, \cdots, a_n^{(t)}) \equiv (a_{ij}^{(t)}).$$

0. 1-2 3-4 5-6
1. 2 1-4 3-6 5
2. 2-4 1-6 3-5
3. 4 2-6 1-5 3
4. 4-6 2-5 1-3
5. 6 4-5 2-3 1
6. 6-5 4-3 2-1
7. 5 6-3 4-1 2
8. 5-3 6-1 4-2
9. 3 5-1 6-2 4
10. 3-1 5-2 6-4
11. 1 3-2 5-4 6
12. 1-2 3-4 5-6

FIG. 1. *Positions of original columns after each time step.*

DEFINITION. We say $a_i^{(t)} \in U$ (the ith column is in "upper-triangular" form) if elements $a_{i+1,i}^{(t)}, \cdots, a_{n,i}^{(t)}$ all equal 0. □

Let us state and prove three lemmas.

LEMMA 2. *If $T_{n-1,n}$ is made at time step t, then $a_{n-1-i}^{(t+i)} \in U$, for $i = 0, 1, \cdots, n-2$.*

Proof. The transformation $T_{n-1,n}$ annihilates the $(n, n-1)$-element. So $a_{n-1}^{(t)} \in U$. Now use induction. At time step $t+j$ $(j \geqq 0)$ we perform $T_{n-1-j,n-j}$ so that column $a_{n-1-j}^{(t+j)} \in U$. At the next time step, the rotation in $T_{n-2-j,n-1-j}$ will create a new zero in the $(n-1-j, n-2-j)$-position, and pivoting will bring zeros to the other subdiagonal positions of column $n-2-j$ from column $n-1-j$. Hence $a_{n-2-j}^{(t+j+1)} \in U$. □

LEMMA 3. *If $T_{n-1,n}$ is made at time step t, then $a_{2i-1}^{(t+n-2)} \in U$ for $i = 1, 2, \cdots, \lfloor n/2 \rfloor$.*

Proof. We perform the transformation $T_{n-1,n}$ at time steps t, $t+2$, $t+4$, $\cdots$. Hence $a_{n-1}^{(t+2j)} \in U$, for $j = 0, 1, \cdots$. Apply Lemma 2 to each of these vectors. □

LEMMA 4. *If $a_1^{(t)}, \cdots, a_i^{(t)}$, $a_{i+2}^{(t)}$ are all in U and $T_{i+1,i+2}$ is made at time step $t+1$, then $a_1^{(t+1)}, \cdots, a_{i+1}^{(t+1)}$ all belong to U.*

Proof. We can check that our basic transformation (2.1) applied to any two consecutive columns in $\{a_1^{(t)}, \cdots, a_i^{(t)}\}$ will not move the resulting pair out of U. The transformation $T_{i+1,i+2}$ will put column $a_{i+1}^{(t+1)}$ in U (see the proof of Lemma 2). □

In words, after transformation $T_{n-1,n}$ the $(n-1)$-st column satisfies the "upper-triangular" property. The column then moves left and picks up appropriate zero elements along the way (Lemma 2). The same event recurs every two time steps. Eventually all odd-numbered columns are in "upper-triangular" form (Lemma 3). After that, the first two columns get in U, then the first three columns, and so on (Lemma 4). We thus need only to determine when the transformation $T_{n-1,n}$ first occurs to compute the time at which the matrix becomes triangularized.

THEOREM 1. *The matrix $A^{(2n-3)}(A^{(2n-2)})$ is upper triangular for n even (n odd).*

Proof. For n even (n odd), we do the transformation $T_{n-1,n}$ at time step 1 (step 2). After $n-2$ additional time steps, all odd-numbered columns are in U (Lemma 3). At the next time step, T_{23} is made. We need $n-2$ time steps for columns $2, 3, \cdots, n-1$ to get in U (Lemma 4). □

It is clear now why the pivot block has been restricted to contiguous elements.

LEMMA 5. *Let $A^{(t)}$ be upper triangular. Then $A^{(t+1)}$ stays upper triangular.*

Proof. Apply $T_{i,i+1}$ $(1 \leqq i < n)$ to $A^{(t)}$. Columns $a_i^{(t+1)}$, $a_{i+1}^{(t+1)}$ still belong to U. □

Since each column of A returns to its original position after $2n$ steps, we have proved our principal result.

THEOREM 2. *Algorithm* QRD *computes a* QR*-factorization of A after $2n$ steps.*

Figure 2 shows how a 6×6 matrix is triangularized after 9 steps. Steps 10 to 12 are necessary to return all columns to their original positions.

To implement the algorithm we associate a processor with each 2×2 block of four contiguous elements. The architecture is the same as the Schur decomposition array introduced in Stewart [13] and detailed in O'Leary-Stewart [11]. There are $(n^2+2n-6)/2$ processors for n even and $(n^2+2n-3)/2$ processors for n odd. Only nearest neighbor connections are required of the processors, since each needs only to receive rotations from some of its neighbors, apply them and pass them on to other neighbors. We do not assume broadcasting of the rotation parameters and so each cluster of rotations requires $(\lceil n/2 \rceil - 1)$ time steps to pass completely out of the matrix. Since the clusters follow each other at intervals of two time steps and the last ($2n$th) cluster begins at step $4n-1$ our algorithm requires a total of $(\lceil 9n/2 \rceil - 2)$ time steps. The rotations propagate through the matrix as shown in Fig. 3 [13].

As mentioned in § 1 we look for a new QRD algorithm to eliminate array interfacings in an SVD computation. All other quadratic QRD arrays [1], [2], [8] are

1.
× × × × × ×
0 × × × × ×
× × × × × ×
× × 0 × × ×
× × × × × ×
× × × × 0 ×

2.
× × × × × ×
× × × × × ×
× 0 × × × ×
× × × × × ×
× × × 0 × ×
× × × 0 × ×

3.
× × × × × ×
0 × × × × ×
× × × × × ×
× × 0 × × ×
× × 0 × × ×
× × 0 × 0 ×

4.
× × × × × ×
× × × × × ×
× 0 × × × ×
× 0 × × × ×
× 0 × 0 × ×
× 0 × 0 × ×

5.
× × × × × ×
0 × × × × ×
0 × × × × ×
0 × 0 × × ×
0 × 0 × × ×
0 × 0 × 0 ×

6.
× × × × × ×
0 × × × × ×
0 0 × × × ×
0 0 × × × ×
0 0 × 0 × ×
0 0 × 0 × ×

7.
× × × × × ×
0 × × × × ×
0 0 × × × ×
0 0 0 × × ×
0 0 0 × × ×
0 0 0 × 0 ×

8.
× × × × × ×
0 × × × × ×
0 0 × × × ×
0 0 0 × × ×
0 0 0 0 × ×
0 0 0 0 × ×

9.
× × × × × ×
0 × × × × ×
0 0 × × × ×
0 0 0 × × ×
0 0 0 0 × ×
0 0 0 0 0 ×

10.
× × × × × ×
0 × × × × ×
0 0 × × × ×
0 0 0 × × ×
0 0 0 0 × ×
0 0 0 0 0 ×

11.
× × × × × ×
0 × × × × ×
0 0 × × × ×
0 0 0 × × ×
0 0 0 0 × ×
0 0 0 0 0 ×

12.
× × × × × ×
0 × × × × ×
0 0 × × × ×
0 0 0 × × ×
0 0 0 0 × ×
0 0 0 0 0 ×

FIG. 2. *The zero–nonzero structure after each time step.*

1.
1 1 × × × × × ×
1 1 × × × × × ×
× × 1 1 × × × ×
× × 1 1 × × × ×
× × × × 1 1 × ×
× × × × 1 1 × ×
× × × × × × 1 1
× × × × × × 1 1

2.
× × 1 1 × × × ×
× × 1 1 × × × ×
1 1 × × 1 1 × ×
1 1 × × 1 1 × ×
× × 1 1 × × 1 1
× × 1 1 × × 1 1
× × × × 1 1 × ×
× × × × 1 1 × ×

3.
× × × × 1 1 × ×
× 2 2 × 1 1 × ×
× 2 2 × × × 1 1
× × × 2 2 × 1 1
1 1 × 2 2 × × ×
1 1 × × × 2 2 ×
× × 1 1 × 2 2 ×
× × 1 1 × × × ×

4.
× 2 2 × × × 1 1
2 × × 2 2 × 1 1
2 × × 2 2 × × ×
× 2 2 × × 2 2 ×
× 2 2 × × 2 2 ×
× × × 2 2 × × 2
1 1 × 2 2 × × 2
1 1 × × × 2 2 ×

5.
3 3 × 2 2 × × ×
3 3 × × × 2 2 ×
× × 3 3 × 2 2 ×
2 × 3 3 × × × 2
2 × × × 3 3 × 2
× 2 2 × 3 3 × ×
× 2 2 × × × 3 3
× × × 2 2 × 3 3

6.
× × 3 3 × 2 2 ×
× × 3 3 × × × 2
3 3 × × 3 3 × 2
3 3 × × 3 3 × ×
× × 3 3 × × 3 3
2 × 3 3 × × 3 3
2 × × × 3 3 × ×
× 2 2 × 3 3 × ×

FIG. 3. *Propagations of the rotations.*

triangular structures composed of $n(n+1)/2$ processors (including the n delay registers in [1], [2]) and they require $3n-2$ time steps. Admittedly, Algorithm QRD needs a little more time and the array structure is slightly more complex. But the possibility of computing an SVD using just one programmable array of processors justifies the additional costs.

It is of independent interest to compare the methods in [1], [2], [8] with Algorithm QRD. The methods of Ahmed et al. [1] and of Gentleman–Kung [8] annihilate elements of A from top down by chess knight moves, while the method of Bojanczyk et al. [2] performs the Givens rotations from bottom up by "long" chess knight moves. These methods all require $3n-5$ stages, where one stage is defined to be a simultaneous application of disjoint plane rotations. Algorithm QRD requires $2n$ stages and creates zeros in a rather unusual manner. The precise way depends on whether n is even or odd and will be omitted. Figure 4 illustrates the three different orderings for $n=8$. Like other parallel Jacobi-type methods [3], [4], [13] Algorithm QRD cannot utilize any banded structure of A.

We complete our analysis by examining roundoff errors. The following lemma comes from Gentleman [7].

LEMMA 6. *If a sequence of plane rotations in a* QRD *scheme can be written as a sequence of s stages, then the final computed matrix obtained when this sequence of plane rotations is applied to a given matrix A will be the exact result of exact computations on a matrix whose difference from A is bounded in norm by* $\eta s(1+\eta)^{s-1}\|A\|_F$, *where* η *denotes a small multiple of the machine precision and* $\|\cdot\|_F$ *the Frobenius matrix norm.*

×	×	×	×	×	×	×	×
1	×	×	×	×	×	×	×
2	4	×	×	×	×	×	×
3	5	7	×	×	×	×	×
4	6	8	10	×	×	×	×
5	7	9	11	13	×	×	×
6	8	10	12	14	16	×	×
7	9	11	13	15	17	19	×

(a) Ahmed et al. [1] and Gentleman–Kung [8].

×	×	×	×	×	×	×	×
7	×	×	×	×	×	×	×
6	9	×	×	×	×	×	×
5	8	11	×	×	×	×	×
4	7	10	13	×	×	×	×
3	6	9	12	15	×	×	×
2	5	8	11	14	17	×	×
1	4	7	10	13	16	19	×

(b) Bojanczyk et al. [2].

×	×	×	×	×	×	×	×
15	×	×	×	×	×	×	×
14	16	×	×	×	×	×	×
13	15	11	×	×	×	×	×
12	14	10	16	×	×	×	×
11	13	9	15	7	×	×	×
10	12	8	14	6	16	×	×
9	11	7	13	5	15	3	×

(c) Algorithm QRD.

FIG. 4. *Three different orders of annihilations.*

We thus obtain a *better* error bound for Algorithm QRD ($s = 2n$) than for the other three methods ($s = 3n - 5$). This result comes as a surprise since redundant zeros are created by our algorithm.

3. Rectangular matrices. In § 2 we consider only square matrices. If the matrix A has more rows than columns, i.e., $A \in R^{m \times n}$ with $m \geqq n$, then we may adopt one of two strategies. The first approach is to apply Algorithm QRD to the square matrix

$$\tilde{A} = (A|0) \in R^{m \times m}.$$

We get

$$\tilde{A} = Q \begin{pmatrix} R & 0 \\ 0 & 0 \end{pmatrix},$$

where $R \in R^{n \times n}$ and so

$$A = Q \begin{pmatrix} R \\ 0 \end{pmatrix}.$$

The procedure requires $O(m)$ time and $O(m^2)$ processors. Its advantage is that no additional hardware is needed, assuming that our processor array can handle $m \times m$ matrices. The problem is of course that this assumption may be wrong.

The second approach is appropriate for a very large m, particularly for $m \gg n$. For simplicity, assume that our array can handle $2n \times 2n$ matrices and that

$$k = m/n$$

is an integer. Partition the matrix in the form:

$$A = \begin{pmatrix} A_1 \\ A_2 \\ \vdots \\ A_k \end{pmatrix},$$

where each block A_i is $n \times n$. We propose a procedure that eliminates the elements of A one block at a time.

ALGORITHM Block QRD.
Set $R_1 := A_k$;
For $i = 1, 2, \cdots, k-1$ do
Use Algorithm QRD to compute a $2n \times 2n$ QR-decomposition:

$$\begin{pmatrix} A_{k-i} & 0 \\ R_i & 0 \end{pmatrix} = Q_{i+1} \begin{pmatrix} R_{i+1} & 0 \\ 0 & 0 \end{pmatrix},$$

where $Q_{i+1} \in R^{2n \times 2n}$ and $R_{i+1} \in R^{n \times n}$.

We get the QR-decomposition of A from

$$Q = \begin{pmatrix} I_{k-2} & 0 \\ 0 & Q_2 \end{pmatrix} \begin{pmatrix} I_{k-3} & 0 & 0 \\ 0 & Q_3 & 0 \\ 0 & 0 & I_1 \end{pmatrix} \begin{pmatrix} I_{k-4} & 0 & 0 \\ 0 & Q_4 & 0 \\ 0 & 0 & I_2 \end{pmatrix} \cdots \begin{pmatrix} I_1 & 0 & 0 \\ 0 & Q_{k-1} & 0 \\ 0 & 0 & I_{k-3} \end{pmatrix} \begin{pmatrix} Q_k & 0 \\ 0 & I_{k-2} \end{pmatrix},$$

where I_j denotes the $jn \times jn$ identity matrix, and

$$R = \begin{pmatrix} R_k \\ 0 \\ \vdots \\ 0 \end{pmatrix}.$$

The requirements are $O(n^2)$ processors and $O(m)$ time. Note that one may speed up the algorithm by using two or more processor arrays. With $\lfloor k/2 \rfloor$ arrays we need but $\lceil \log_2 k \rceil$ steps. Thus, excluding the costs of data input and output, the required time can be cut to $O(n \log k)$.

REFERENCES

[1] H. M. AHMED, J.-M. DELOSME AND M. MORF, *Highly concurrent computing structures for matrix arithmetic and signal processing*, Computer, 15 (Jan. 1982), pp. 65–82.

[2] A. BOJANCZYK, R. P. BRENT AND H. T. KUNG, *Numerically stable solution of dense systems of linear equations using mesh-connected processors*, this Journal, 5 (1984), pp. 95–104.

[3] R. P. BRENT AND F. T. LUK, *The solution of singular-value and symmetric eigenvalue problems on multiprocessor arrays*, this Journal, 6 (1985), pp. 69–84.

[4] R. P. BRENT, F. T. LUK AND C. VAN LOAN, *Computation of the singular value decomposition using mesh-connected processors*, J. VLSI Computer Systems, 1 (1985), to appear.

[5] ———, *Computation of the generalized singular value decomposition using mesh-connected processors*, Proc. SPIE Vol. 431, Real Time Signal Processing, VI (1983), pp. 66–71.

[6] K. BROMLEY AND J. M. SPEISER, *Signal Processing Algorithms, Architectures, and Applications*, Tutorial 31, SPIE 27th Annual Internat. Tech. Symp., San Diego, Aug. 1983.

[7] W. M. GENTLEMAN, *Error analysis of QR decompositions by Givens transformations*, Lin. Alg. Applics., 10 (1975), pp. 189–197.

[8] W. M. GENTLEMAN AND H. T. KUNG, *Matrix triangularization by systolic arrays*, Proc. SPIE Vol. 298, Real Time Signal Processing IV, pp. 19–26.

[9] D. E. HELLER AND I. C. F. IPSEN, *Systolic networks for orthogonal decompositions*, this Journal, 4 (1983), pp. 261–269.

[10] L. JOHNSSON, *A computational array for the* QR*-method*, Proc. Conf. Adv. Research in VLSI, P. Penfield, ed., Artech House, Inc., Dedham, MA, 1982, pp. 123–129.

[11] D. P. O'LEARY AND G. W. STEWART, *Data-flow algorithms for parallel matrix computations*, Tech. Report 1366, Computer Science Dept., Univ. Maryland, College Park, 1984.

[12] A. H. SAMEH, *Solving the linear least squares problem on a linear array of processors*, Proc. Purdue Workshop Algorithmically-Specialized Computer Organ., W. Lafayette, IN, Sept. 1982.

[13] G. W. STEWART, *A Jacobi-like algorithm for computing the Schur decomposition of a non-Hermitian matrix*, this Journal, 6 (1985), pp. 853–864.

SIAM J. SCI. STAT. COMPUT.
Vol. 7, No. 2, April 1986

008

ORTHOGONAL REDUCTION OF SPARSE MATRICES TO UPPER TRIANGULAR FORM USING HOUSEHOLDER TRANSFORMATIONS*

ALAN GEORGE† AND ESMOND NG†

Abstract. In this paper we consider the problem of predicting where fill-in occurs in the orthogonal decomposition of sparse matrices using Householder transformations. We show that a static data structure can be used throughout the numerical computation, and that the Householder transformations can be saved explicitly in a compact format.

Key words. sparse matrices, orthogonal reduction, sparse least squares, sparse linear systems

1. Introduction. Let A be an $n \times n$ nonsingular matrix. In this paper we consider the problem of reducing A to upper triangular form using orthogonal transformations, where A is large and sparse. That is, we construct an $n \times n$ orthogonal matrix Q so that

$$A = QR,$$

where R is $n \times n$ and upper triangular. Since it is well known that computing such a decomposition is numerically stable, the QR-decomposition is useful in various numerical computations, such as the solution of nonsingular systems of linear equations. However, very few implementations of the QR-decomposition exist for A when it is large and sparse. This is apparently due to the general belief that the orthogonal matrix Q and the intermediate matrices may be dense even though A is sparse, and also due to the lack of efficient techniques for exploiting the sparsity of the orthogonal matrix and the intermediate matrices.

One such implementation is due to George and Heath [4]. They make use of the fact that the upper triangular matrix R is (mathematically) the Cholesky factor of the symmetric positive definite matrix A^TA (apart from possible sign differences in some rows). Thus, assuming A^TA and its Cholesky factor are sparse, one can easily determine the structure of R and set up a data structure which exploits the sparsity of R using techniques developed for solving sparse symmetric positive definite systems [5]. Then R can be computed using the *static* data structure by applying Givens rotations to the rows of A one at a time. The Givens rotations are not saved in their implementation. However in some applications, it is desirable or necessary to have the orthogonal matrix Q available. One such context is the solution of several problems that have the same coefficient matrix A but different right-hand side vectors. In this paper, we show that if we compute the decomposition using Householder transformations, then the nonzeros in the transformations and in the intermediate matrices can be stored in a static data structure allocated for the Cholesky factors L and L^T of the matrix A^TA, provided that the diagonal elements of A are nonzero. We also extend the technique to handle sparse rectangular matrices.

Similar results hold for the triangular factorization of sparse matrices using Gaussian elimination with partial pivoting [6]. The authors have shown that, under the assumption that the diagonal elements of a sparse matrix A are nonzero, the structures of the triangular factors obtained in the triangular decomposition of A using

* Received by the editors February 24, 1984, and in revised form February 26, 1985. This research was sponsored by the Canadian Natural Sciences and Engineering Research Council under grant A8111, and was also sponsored by the Applied Mathematical Sciences Research Program, Office of Energy Research, U.S. Department of Energy under contract DE-AC05-84OR21400 with the Martin Marietta Energy Systems Inc., and by the U.S. Air Force Office of Scientific Research under contract AFOSR-ISSA-84-00056.

† Department of Computer Science, University of Waterloo, Waterloo, Ontario, Canada N2L 3G1.

Gaussian elimination are contained in the structures of the Cholesky factors L and L^T of A^TA, regardless of the choice of row interchanges. Experience has shown that the structures provided by the Cholesky factors L and L^T of A^TA are often tighter for the orthogonal decomposition of A than for the triangular factorization of A.

The success of the approach described in this paper relies heavily on the assumption that both A^TA and its Cholesky factor L are sparse. This is often true when A is sparse. However, there are instances in which A^TA and L are dense even if A is sparse. An example and further discussions can be found at the end of § 2.

An outline of this paper is as follows. In § 2, we present the main results which show that the structures of the transformations and the intermediate matrices obtained in the orthogonal reduction of A are contained in the structures of the Cholesky factors of A^TA. The effect of permuting the columns of A in the orthogonal reduction is considered in § 3. In § 4, the basic technique of the paper is extended to handle rectangular matrices. Finally, some concluding remarks are provided in § 5.

2. Basic results. Let A be an $n \times n$ nonsingular matrix. The following notation will be used throughout our discussion. The (i, j)-element of the matrix A is denoted by a_{ij}. The set of indices of the nonzeros in A is denoted by Nonz (A); that is,

$$\text{Nonz}\,(A) = \{(i, j) | a_{ij} \neq 0\}.$$

The matrix A is said to have a *zero-free diagonal* if all its diagonal elements are nonzero.

LEMMA 2.1 [3]. *Let A be an $n \times n$ nonsingular matrix. Then there exists a permutation matrix P such that PA has a zero-free diagonal.*

For convenience, we assume in the following discussion that the rows of A have been permuted so that A has a zero-free diagonal. The next result is useful in deriving the main results.

LEMMA 2.2. *Suppose A is $n \times n$ and has a zero-free diagonal, and let B be an $n \times p$ matrix. Then*

$$\text{Nonz}\,(B) \subseteq \text{Nonz}\,(AB).$$

We will also assume that accidental *structural* or *numerical* cancellation does not occur; that is, we assume that Nonz $(A+B) =$ Nonz $(A) \cup$ Nonz (B), for any $n \times n$ matrices A and B.

Now let $A_0 = A$ and partition A_0 into

$$A_0 = \begin{pmatrix} \alpha_1 & y_1^T \\ x_1 & B_1 \end{pmatrix},$$

where B_1 is $(n-1) \times (n-1)$, and x_1 and y_1 are vectors of appropriate dimensions. By assumption, $\alpha_1 \neq 0$ and B_1 has a zero-free diagonal. Assume $x_1 \neq 0$ and consider annihilating the nonzeros of x_1 using a Householder transformation H_1. (If $x_1 = 0$, then $H_1 = I$.) One way of constructing the Householder matrix H_1 is as follows. Define an n-vector w_1 by

$$w_1 = \begin{pmatrix} \alpha_1 + \sigma_1 \\ x_1 \end{pmatrix},$$

where $\sigma_1^2 = \alpha_1^2 + x_1^T x_1$. Let $\pi_1 = \frac{1}{2} w_1^T w_1$. Then it is easy to verify that

$$H_1 = I - \frac{1}{\pi_1} w_1 w_1^T$$

is orthogonal and

$$H_1\begin{pmatrix}\alpha_1\\ x_1\end{pmatrix}=\begin{pmatrix}-\sigma_1\\ 0\end{pmatrix}.$$

There are other ways of constructing H_1 (see [9]) and they differ essentially in the way the vector $\binom{\alpha_1}{x_1}$ is scaled. Thus we can assume that in general the Householder matrix H_1 has the form

$$H_1 = I - \frac{1}{\pi_1} w_1 w_1^T,$$

where

$$w_1 = \begin{pmatrix}\beta_1\\ u_1\end{pmatrix},$$

for some appropriate π_1, β_1 and u_1, with $\beta_1 \neq 0$ and Nonz (u_1) = Nonz (x_1). Note that by storing the nonzeros of u_1 (and β_1 and π_1), one can save H_1 in a compact format.

Consider applying H_1 to A_0. Let

$$H_1 A_0 = \begin{pmatrix}-\sigma_1 & z_1^T\\ 0 & A_1\end{pmatrix},$$

where

$$\begin{pmatrix}z_1^T\\ A_1\end{pmatrix} = H_1\begin{pmatrix}y_1^T\\ B_1\end{pmatrix} = \left(I - \frac{1}{\pi_1} w_1 w_1^T\right)\begin{pmatrix}y_1^T\\ B_1\end{pmatrix} = \begin{pmatrix}y_1^T\\ B_1\end{pmatrix} - \frac{1}{\pi_1}\begin{pmatrix}\beta_1\\ u_1\end{pmatrix}(\beta_1 \quad u_1^T)\begin{pmatrix}y_1^T\\ B_1\end{pmatrix}.$$

Thus,

$$z_1 = y_1 - \frac{1}{\pi_1}\beta_1(\beta_1 y_1 + B_1^T u_1)$$

and

$$A_1 = B_1 - \frac{1}{\pi_1} u_1(\beta_1 y_1^T + u_1^T B_1).$$

Since $\beta_1 \neq 0$, and if exact structural cancellation does not occur,

$$\text{Nonz}\,(z_1) = \text{Nonz}\,(y_1) \cup \text{Nonz}\,(B_1^T u_1)$$

and

$$\text{Nonz}\,(A_1) = \text{Nonz}\,(B_1) \cup \text{Nonz}\,(u_1 y_1^T) \cup \text{Nonz}\,(u_1 u_1^T B_1).$$

Furthermore, since Nonz (u_1) = Nonz (x_1), we obtain the following which we state as a lemma for future reference.

LEMMA 2.3.

(1) Nonz (z_1) = Nonz (y_1) $\cup$ Nonz $(B_1^T x_1)$.

(2) Nonz (A_1) = Nonz (B_1) $\cup$ Nonz $(x_1 y_1^T)$ $\cup$ Nonz $(x_1 x_1^T B_1)$.

COROLLARY 2.4. *A_1 has a zero-free diagonal.*

Note that similar results hold if Givens rotations are used to annihilate the nonzeros in x_1. The effect of applying a Givens rotation to eliminate a nonzero, say a_{k1}, is to replace the first and the kth rows of A by a *linear combination* of those two rows. Consequently, after annihilating a_{k1}, the structures of rows 1 and k of A are the *union* of the structures of the original rows. Thus, after *all* the nonzeros in x_1 have been

annihilated, the structure of row 1 of A will be the union of the structures of the first row of A and those rows that have a nonzero in column 1. That is, Nonz (z_1) will be given by

$$\text{Nonz}(z_1) = \text{Nonz}(y_1) \cup \text{Nonz}(B_1^T x_1).$$

Using similar arguments, it is easy to see that, in the worst case, the structure of the remaining $(n-1)\times(n-1)$ matrix A_1 will be given by

$$\text{Nonz}(A_1) = \text{Nonz}(B_1) \cup \text{Nonz}(x_1(y_1^T + x_1^T B_1)).$$

Thus Lemma 2.3 holds even if Givens rotations are used. Now for each nonzero in x_1, there will be one Givens rotation. In order to save these Givens rotations in the space provided by the nonzeros in x_1, we need to represent each of them by a single number using the scheme proposed by Stewart [10]. There is one disadvantage though; extra time is needed to form this single number in the numerical computation phase.

We now show that the structures of u_1, z_1 and A_1 are related to the structures of the matrices obtained after applying one step of Cholesky decomposition to the symmetric positive definite matrix $A_0^T A_0$. Note that

$$A_0^T A_0 = \begin{pmatrix} \alpha_1 & x_1^T \\ y_1 & B_1^T \end{pmatrix} \begin{pmatrix} \alpha_1 & y_1^T \\ x_1 & B_1 \end{pmatrix} = \begin{pmatrix} \alpha_1^2 + x_1^T x_1 & \alpha_1 y_1^T + x_1^T B_1 \\ \alpha_1 y_1 + B_1^T x_1 & B_1^T B_1 + y_1 y_1^T \end{pmatrix} = \begin{pmatrix} \tau_1 & v_1^T \\ v_1 & E_1 \end{pmatrix}.$$

Applying one step of Cholesky decomposition to $A_0^T A_0$, we obtain

$$A_0^T A_0 = \begin{pmatrix} \tau_1^{1/2} & 0 \\ v_1/\tau_1^{1/2} & I \end{pmatrix} \begin{pmatrix} 1 & 0 \\ 0 & F_1 \end{pmatrix} \begin{pmatrix} \tau_1^{1/2} & v_1^T/\tau_1^{1/2} \\ 0 & I \end{pmatrix},$$

where

$$F_1 = E_1 - \frac{1}{\tau_1} v_1 v_1^T.$$

The first observation is that

$$\text{Nonz}(v_1) = \text{Nonz}(y_1) \cup \text{Nonz}(B_1^T x_1),$$

assuming again exact structural cancellation does not occur and also because $\alpha_1 \neq 0$. Since B_1 has a zero-free diagonal, it follows from Lemma 2.2 that

$$\text{Nonz}(x_1) \subseteq \text{Nonz}(B_1^T x_1),$$

and hence

$$\text{Nonz}(u_1) = \text{Nonz}(x_1) \subseteq \text{Nonz}(B_1^T x_1) \subseteq \text{Nonz}(y_1) \cup \text{Nonz}(B_1^T x_1) = \text{Nonz}(v_1).$$

Moreover, from Lemma 2.3,

$$\text{Nonz}(z_1) = \text{Nonz}(y_1) \cup \text{Nonz}(B_1^T x_1) = \text{Nonz}(v_1).$$

Consider the matrix F_1.

$$F_1 = E_1 - \frac{1}{\tau_1} v_1 v_1^T = (B_1^T B_1 + y_1 y_1^T) - \frac{1}{\tau_1}(\alpha_1 y_1 + B_1^T x_1)(\alpha_1 y_1^T + x_1^T B_1).$$

If exact structural cancellation does not occur, then

$$\begin{aligned} \text{Nonz}(F_1) = {} & \text{Nonz}(B_1^T B_1) \cup \text{Nonz}(y_1 y_1^T) \\ & \cup \text{Nonz}(B_1^T x_1 y_1^T) \cup \text{Nonz}(y_1 x_1^T B_1) \cup \text{Nonz}(B_1^T x_1 x_1^T B_1). \end{aligned}$$

Recall from Lemma 2.3 that

$$\text{Nonz}(A_1) = \text{Nonz}(B_1) \cup \text{Nonz}(x_1 y_1^T) \cup \text{Nonz}(x_1 x_1^T B_1).$$

Since B_1 has a zero-free diagonal, it follows from Lemma 2.2 that

$$\text{Nonz}(A_1) \subseteq \text{Nonz}(B_1^T B_1) \cup \text{Nonz}(B_1^T x_1 y_1^T) \cup \text{Nonz}(B_1^T x_1 x_1^T B_1) \subseteq \text{Nonz}(F_1).$$

Thus we have proved the following result.

THEOREM 2.5. *Assume exact structural cancellation does not occur. Then*

(1) $\text{Nonz}(u_1) \subseteq \text{Nonz}(v_1)$.

(2) $\text{Nonz}(z_1) \subseteq \text{Nonz}(v_1)$.

(3) $\text{Nonz}(A_1) \subseteq \text{Nonz}(F_1)$.

That is, the structures of u_1, z_1 and A_1 which are obtained when x_1 is annihilated by a Householder transformation are *contained* in those of the matrices obtained after applying one step of Cholesky decomposition to $A_0^T A_0$. The fact that A_0 has a zero-free diagonal plays an important role here. Some of the results above may not hold if A_0 does not have a zero-free diagonal. For example, it is easy to construct an example in which $\text{Nonz}(B_1) \not\subseteq \text{Nonz}(B_1^T B_1)$, where B_1 does not have a zero-free diagonal. It should be emphasized that this "zero-free diagonal" property is essential only in predicting the fill-in, but not in computing (numerically) the orthogonal decomposition using Householder transformations.

Now partition A_1 into

$$A_1 = \begin{pmatrix} \alpha_2 & y_2^T \\ x_2 & B_2 \end{pmatrix},$$

and assume $x_2 \neq 0$. Consider annihilating the nonzeros of x_2 using a Householder transformation H_2. Let

$$H_2 = I - \frac{1}{\pi_2} w_2 w_2^T,$$

where $w_2 = \binom{\beta_2}{u_2}$ with $\text{Nonz}(u_2) = \text{Nonz}(x_2)$. As before, π_2, β_2 and u_2 are chosen so that

$$H_2 \begin{pmatrix} \alpha_2 \\ x_2 \end{pmatrix} = \begin{pmatrix} -\sigma_2 \\ 0 \end{pmatrix},$$

where $\sigma_2^2 = \alpha_2^2 + x_2^T x_2$. Suppose

$$H_2 A_1 = \begin{pmatrix} -\sigma_2 & z_2^T \\ 0 & A_2 \end{pmatrix}.$$

By Corollary 2.3, A_1 has a zero-free diagonal and hence Theorem 2.5 applies again. That is, the structures of u_2, z_2 and A_2 must be contained in the structures of the matrices obtained by applying one step of Cholesky decomposition to $A_1^T A_1$.

Apparently the results obtained so far do not provide us with a mechanism to implement the orthogonal reduction of sparse matrices efficiently using Householder transformations since we now have to consider the Cholesky decomposition of $A_1^T A_1$. However, the next result takes care of this problem.

LEMMA 2.6. *Assuming exact cancellation does not occur,*

$$\text{Nonz}(A_1^T A_1) = \text{Nonz}(F_1).$$

Proof. Recall that

$$A_1 = B_1 - \frac{1}{\pi_1} u_1 (\beta_1 y_1^T + u_1^T B_1).$$

It is then straightforward to verify that

$$A_1^T A_1 = B_1^T B_1 + \frac{\beta_1^2 u_1^T u_1}{\pi_1^2} y_1 y_1^T + \left(\frac{u_1^T u_1}{\pi_1^2} - \frac{2}{\pi_1}\right) B_1^T u_1 u_1^T B_1$$
$$+ \left(\frac{\beta_1 u_1^T u_1}{\pi_1^2} - \frac{\beta_1}{\pi_1}\right)(B_1^T u_1 y_1^T + y_1 u_1^T B_1).$$

Thus, assuming exact structural cancellation does not occur and assuming $\beta_1 \neq 0$,

$$\begin{aligned}\text{Nonz}(A_1^T A_1) &= \text{Nonz}(B_1^T B_1) \cup \text{Nonz}(y_1 y_1^T) \cup \text{Nonz}(B_1^T u_1 y_1^T) \cup \text{Nonz}(y_1 u_1^T B_1) \\ &\quad \cup \text{Nonz}(B_1^T u_1 u_1^T B_1) \\ &= \text{Nonz}(B_1^T B_1) \cup \text{Nonz}(y_1 y_1^T) \cup \text{Nonz}(B_1^T x_1 y_1^T) \cup \text{Nonz}(y_1 x_1^T B_1) \\ &\quad \cup \text{Nonz}(B_1^T x_1 x_1^T B_1),\end{aligned}$$

since $\text{Nonz}(u_1) = \text{Nonz}(x_1)$. Hence

$$\text{Nonz}(A_1^T A_1) = \text{Nonz}(F_1). \qquad \square$$

COROLLARY 2.7. *The Cholesky factors of $A_1^T A_1$ and F_1 have identical nonzero structures, assuming exact structural cancellation does not occur.*

Lemma 2.6 and Corollary 2.7 are important since they say that we do not have to worry about the Cholesky decomposition of $A_1^T A_1$. We only have to consider the Cholesky decomposition of F_1. That is, suppose

$$F_1 = \begin{pmatrix} \tau_2 & v_2^T \\ v_2 & E_2 \end{pmatrix} = \begin{pmatrix} \tau_2^{1/2} & 0 \\ v_2/\tau_2^{1/2} & I \end{pmatrix} \begin{pmatrix} 1 & 0 \\ 0 & F_2 \end{pmatrix} \begin{pmatrix} \tau_2^{1/2} & v_2^T/\tau_2^{1/2} \\ 0 & I \end{pmatrix}.$$

Then,

$$\text{Nonz}(u_2) \subseteq \text{Nonz}(v_2),$$
$$\text{Nonz}(z_2) \subseteq \text{Nonz}(v_2),$$

and

$$\text{Nonz}(A_2) \subseteq \text{Nonz}(F_2).$$

By applying the arguments above recursively to A_2 and F_2, one can obtain a result which is a generalization of Theorem 2.5. Before stating the result, we introduce more notation.

Let A be an $n \times n$ matrix with a zero-free diagonal and let $A_0 = A$. Consider the sequence of matrices

$$\{A_0, A_1, A_2, \cdots, A_{n-1}\}$$

generated as follows. For $k = 1, 2, \cdots, n-1$, partition A_{k-1} into

$$A_{k-1} = \begin{pmatrix} \alpha_k & y_k^T \\ x_k & B_k \end{pmatrix}.$$

Assume $x_k \neq 0$ and construct a Householder transformation H_k so that

$$H_k A_{k-1} = \begin{pmatrix} -\sigma_k & z_k^T \\ 0 & A_k \end{pmatrix},$$

where $\sigma_k^2 = \alpha_k^2 + x_k^T x_k$. Assume

$$H_k = I_{n-k+1} - \frac{1}{\pi_k} w_k w_k^T,$$

where $w_k = \binom{\beta_k}{u_k}$ with $\text{Nonz}(u_k) = \text{Nonz}(x_k)$. Here I_j denotes the identity matrix of order j. It is easy to see that

$$A = Q_1 Q_2 \cdots Q_{n-2} Q_{n-1} \begin{pmatrix} a_1 & & z_1^T & & \\ & \alpha_2 & & z_2^T & \\ & & \alpha_3 & & z_3^T \\ & & & \ddots & \end{pmatrix} = QR,$$

where

$$Q_k = \begin{pmatrix} I_{k-1} & O \\ O & H_k \end{pmatrix} \quad \text{for } k = 1, 2, \cdots, n-1,$$

and

$$Q = Q_1 Q_2 \cdots Q_{n-1}.$$

Also consider the sequence of matrices

$$\{F_0, F_1, F_2, \cdots, F_{n-1}\},$$

which is defined as follows. Let $F_0 = A^T A$. For $k = 1, 2, \cdots, n-1$, partition F_{k-1} into

$$F_{k-1} = \begin{pmatrix} \tau_k & v_k^T \\ v_k & E_k \end{pmatrix}.$$

Applying one step of Cholesky decomposition to F_{k-1} yields

$$F_{k-1} = \begin{pmatrix} \tau_k^{1/2} & 0 \\ v_k/\tau_k^{1/2} & I_{n-k} \end{pmatrix} \begin{pmatrix} I_k & O \\ O & F_k \end{pmatrix} \begin{pmatrix} \tau_k^{1/2} & v_k^T/\tau_k^{1/2} \\ 0 & I_{n-k} \end{pmatrix}.$$

If we define L_k by

$$L_k = \begin{pmatrix} I_{k-1} & 0 & 0 \\ 0 & \tau_k^{1/2} & 0 \\ 0 & v_k/\tau_k^{1/2} & I_{n-k} \end{pmatrix}, \qquad k = 1, 2, \cdots, n-1,$$

and L_n by

$$L_n = \begin{pmatrix} I_{n-1} & O \\ O & F_{n-1} \end{pmatrix} = \begin{pmatrix} I_{n-1} & O \\ O & \tau_n^{1/2} \end{pmatrix},$$

then it is clear that

$$A^T A = F_0 = L_1 L_2 \cdots L_{n-1} L_n L_n^T L_{n-1}^T \cdots L_2^T L_1^T = LL^T,$$

where $L = L_1 L_2 \cdots L_{n-1} L_n$. Moreover, because of the way v_k and F_k are constructed, we have

$$\text{Nonz}(F_k) \subseteq \bigcup_{k=1}^{n} \text{Nonz}(L_k + L_k^T) = \text{Nonz}(L + L^T).$$

The following result is a generalization of Theorem 2.5. Its proof is similar to that of Theorem 2.5 and hence is omitted.

THEOREM 2.8. *Assume exact structural cancellation does not occur. Then for* $k = 1, 2, \cdots, n-1$,

(1) A_k *has a zero-free diagonal.*

(2) Nonz $(u_k) \subseteq$ Nonz (v_k).

(3) Nonz $(z_k) \subseteq$ Nonz (v_k).

(4) Nonz $(A_k) \subseteq$ Nonz$(F_k) \subseteq$ Nonz $(L + L^T)$.

Theorem 2.8 has an important implication. If A is sparse, then it says that the structures of the vectors u_k (which are the major components in the construction of Q) and the upper triangular matrix R are *all contained* in the structures of the Cholesky factors of A^TA. The crucial point is that if A^TA and its Cholesky factor are sparse, then it is possible to determine the structure of the Cholesky factor L of A^TA from that of A^TA efficiently. The reader is referred to [5] for details. Knowing the structure of L, one can set up an efficient data structure that exploits the sparsity of L and L^T. Now Theorem 2.8 simply implies that one can compute the orthogonal decomposition using Householder transformations in that *static* data structure. No dynamic storage allocation is necessary. Furthermore, the orthogonal matrix Q (in factored form) can be retained. This may be useful in some situations, for example, when the QR-decomposition of A has to be used several times.

In deriving the results, we have made heavy use of Lemma 2.2. Note that even though the structure of B is contained in that of AB when A has a zero-free diagonal, and AB may be sparse, the structure of AB may still overestimate the structure of B. Also, we usually have a relationship of the form

$$\text{Nonz}\,(X) \subseteq \text{Nonz}\,(MX) \cup \text{Nonz}\,(Y),$$

for some matrices M, X and Y, where M has a zero-free diagonal. Since Y has nothing to do with X, it may appear that the way in which we bound the fill-in is too generous. However, experience has shown that our approach is quite reasonable in most cases.

As we have mentioned in § 1, the success of our approach relies on the assumption that A^TA and its Cholesky factor L are sparse if A is sparse. There are examples in which this may not be true; the matrices A^TA and L may be dense even if A is sparse. The following small example illustrates the difficulty.

$$A = \begin{pmatrix} \times & \times & \times & \times \\ & \times & & \\ & & \times & \\ & & & \times \end{pmatrix}$$

$$A^TA = \begin{pmatrix} \times & \times & \times & \times \\ \times & \times & \times & \times \\ \times & \times & \times & \times \\ \times & \times & \times & \times \end{pmatrix} \quad \text{and} \quad L = \begin{pmatrix} \times & & & \\ \times & \times & & \\ \times & \times & \times & \\ \times & \times & \times & \times \end{pmatrix}.$$

In this example, $Q = I$ and the structure of R is identical to that of A. Thus, the structures of the Cholesky factors (L and L^T) of A^TA overestimate the structures of the Householder transformations and the upper triangular matrix. Fortunately, experience has shown that this situation arises usually because there are a relatively small number of dense rows in A. Even though identifying these rows is a difficult problem, there are schemes which can handle dense rows in an efficient manner [6], [7]. Another instance in which an overestimate will occur is when the original matrix is upper triangular or nearly so.

3. Effect of permuting the columns. Let P_c be an $n\times n$ permutation matrix and denote the QR-decomposition of AP_c by

$$AP_c=\hat{Q}_1\hat{Q}_2\cdots\hat{Q}_{n-2}\hat{Q}_{n-1}\hat{R},$$

where $\hat{Q}_k$ is an appropriate Householder transformation and $\hat{R}$ is an $n\times n$ upper triangular matrix. Our results in the previous section indicate that the structures of $\hat{R}$ and the vectors used in constructing $\hat{Q}_k$ are contained in the structures of the Cholesky factors $\hat{L}^T$ and $\hat{L}$ of the symmetric positive definite matrix $(AP_c)^T(AP_c)=P_c^TA^TAP_c$. If A^TA is sparse, it is well known that the structure and the sparsity of $\hat{L}$ depend not only on the structure of A^TA, but also on the choice of the permutation matrix P_c. Thus it is desirable to choose P_c so that $\hat{L}$ is as sparse as possible. Unfortunately, the problem of finding such a permutation has been shown to be an NP-complete problem [11]. On the other hand, there are several reliable heuristic algorithms for finding P_c that yield a reasonably sparse $\hat{L}$. Examples include the nested dissection algorithm and the minimum degree algorithm. See [5] for a detailed discussion of the ordering problem in sparse Cholesky decomposition.

$$A=\begin{pmatrix}\times&\times&\times&&\\&\times&&&\times\\\times&&\times&&\\&&&\times&\\&\times&&\times&\times\end{pmatrix}\quad P_c=\begin{pmatrix}&&&&1\\&&&1&\\&&1&&\\&1&&&\\1&&&&\end{pmatrix}\quad AP_c=\begin{pmatrix}&&&\times&\times&\times\\\times&&&&\times&\\&&&\times&&\times\\&&\times&&&\\\times&\times&&&\times&\end{pmatrix}$$

FIG. 3.1. *An example illustrating the fact that AP_c may not have a zero-free diagonal even though A has one.*

Note that post-multiplying A by P_c may change the zero-nonzero pattern of A. In particular the matrix AP_c may no longer have a zero-free diagonal (assuming A originally has one). This is illustrated by an example in Fig. 3.1. In order to preserve the zero-free diagonal, we can apply P_c to A symmetrically. That is, instead of looking at AP_c, we consider $P_c^TAP_c$. It is a simple exercise to verify that $P_c^TAP_c$ has a zero-free diagonal for the matrix A given in Fig. 3.1. Also note that pre-multiplying AP_c by P_c has no effect on the structure of $\hat{L}$ since

$$(P_c^TAP_c)^T(P_c^TAP_c)=P_c^TA^TAP_c=\hat{L}\hat{L}^T.$$

Another approach which solves this problem is to find a column permutation P_c first. Then we find a row permutation P_r to make sure that $P_r(AP_c)$ has a zero-free diagonal. The main observation here is that the Cholesky factor of $(P_rAP_c)^T(P_rAP_c)$ is mathematically the same as that of $(AP_c)^T(AP_c)$.

4. Generalization to rectangular matrices. In some situations, such as the solution of sparse linear least squares problems, it may be necessary to reduce a rectangular matrix to upper trapezoidal form. The approach we described in §§ 2 and 3 can be modified to handle these cases. Let A be an $m\times n$ sparse matrix with $m\geqq n$. We assume that A has full column rank. Partition A into

$$A=\begin{pmatrix}B\\C\end{pmatrix},$$

where B is $n\times n$ and C is $(m-n)+n$. For simplicity, we also assume B has a zero-free diagonal.

Denote the orthogonal decomposition of A by

$$A=Q_1Q_2\cdots Q_n\begin{pmatrix}R\\O\end{pmatrix},$$

where Q_k is an $m\times m$ Householder matrix and R is an $n\times n$ upper triangular matrix. Suppose

$$Q_k=\begin{pmatrix}I_{k-1} & O\\ O & H_k\end{pmatrix},$$

with

$$H_k=I_{m-k+1}-\frac{1}{\pi_k}\begin{pmatrix}\beta_k\\ u_k\end{pmatrix}(\beta_k \quad u_k^T).$$

Here u_k is an $(m-k)$-vector. Note that the decomposition is equivalent to performing the first n steps in the orthogonal reduction of the $m\times m$ matrix $\bar{A}$:

$$\bar{A}=\begin{pmatrix}B & O\\ C & I\end{pmatrix}.$$

Consider the matrix $\bar{A}^T\bar{A}$.

$$\bar{A}^T\bar{A}=\begin{pmatrix}B^T & C^T\\ O & I\end{pmatrix}\begin{pmatrix}B & O\\ C & I\end{pmatrix}=\begin{pmatrix}B^TB+C^TC & C^T\\ C & I\end{pmatrix}=\begin{pmatrix}D & C^T\\ C & I\end{pmatrix}.$$

Applying the first n steps of the Cholesky decomposition to $\bar{A}^T\bar{A}$ yields

$$\bar{A}^T\bar{A}=\begin{pmatrix}D & C^T\\ C & I\end{pmatrix}=\begin{pmatrix}L & O\\ W & I\end{pmatrix}\begin{pmatrix}I & O\\ O & F\end{pmatrix}\begin{pmatrix}L^T & W^T\\ O & I\end{pmatrix},$$

where

$$LL^T=D=B^TB+C^TC=A^TA=R^TR,$$

and

$$W=CL^{-T}.$$

Since $\bar{A}$ has a zero-free diagonal, the results in § 2 apply. That is, the structure of u_k must be contained in the structure of the kth column of the matrix $\binom{L}{W}$. Similarly, the structure of R must be contained in the structure of L^T. Thus, one way to implement the reduction of A is as follows:

(1) Determine the structure of $M=\bar{A}^T\bar{A}$.

(2) Perform the first n steps of symbolic Cholesky factorization of M, and determine the structures of L and $W=CL^{-T}$. Set up a data structure that exploits the sparsity of L^T and $\binom{L}{W}$.

(3) Reduce the matrix A to upper trapezoidal form using Householder transformations, storing R and u_k in the static data structure determined in Step 2.

Note that we only want to reduce $\binom{B}{C}$ to upper trapezoidal form. Thus we do not want to worry about the last $(m-n)$ columns in $\bar{A}$. In other words, if we want to permute the columns of $\bar{A}$ so as to obtain a sparse Cholesky factorization, we should only permute the first n columns of $\bar{A}$.

5. Concluding remarks. We have shown in this paper that when a sparse matrix A is reduced to upper triangular form using Householder transformations, the structures of the transformations, the intermediate matrices and the final upper triangular matrix

are contained in the structures of the Cholesky factors of A^TA. These results have an important practical implication. It is well-known that the structure of the Cholesky factor of a sparse symmetric positive definite matrix B can be determined efficiently from the structure of B. Thus, by analyzing the structure of A^TA, we can determine the structures of the Cholesky factors L and L^T of A^TA, and can set up an appropriate data structure for L and L^T. Then we can perform the orthogonal reduction of A using this static data structure. By using techniques and software developed for solving sparse symmetric positive definite systems, the implementation of sparse orthogonal decomposition using Householder transformations is straightforward. The results have been extended to handle the case in which A is rectangular.

Preliminary experiments have indicated that the approach described in this paper is quite reasonable for some classes of problems, although there are situations in which the structure of the Cholesky factor L of A^TA may overestimate the structure of the Householder transformations. Recently, the authors have developed a new technique for predicting where fill-in in sparse orthogonal decomposition occurs [7]. Preliminary experiments show that the data structure obtained is often tighter than that generated by the approach described in this paper, especially for the Householder transformations.

In [2], Coleman, Edenbrandt and Gilbert considered the problem of characterizing matrices for which the structure of the Cholesky factor L^T of A^TA predicts exactly the structure of R. They were concerned with the case where R is computed by applying Givens rotations to the rows of A. Their basic result is as follows. Let A be an $m\times n$ matrix, with $m\geqq n$. For *each* $k=1, 2, \cdots, n$, if there are *more than* k non-null rows in *every* $m\times k$ submatrix of A, then the structure of the Cholesky factor L^T of A^TA predicts the structure of R exactly. This property is referred to as the *strong Hall property*. When the matrix A does not have the strong Hall property, they pointed out that the rows and columns of A can be permuted symmetrically so that the permuted matrix has a block triangular form and the diagonal blocks have the strong Hall property. In this case, only the diagonal blocks have to be decomposed. Suppose A has the strong Hall property. Since R is mathematically unique regardless of whether Givens rotations or Householder transformations are used in computing it, the structure of the Cholesky factor L^T of A^TA must predict accurately the structure of R even when it is computed using Householder transformations. However, it is not necessarily true that the structure of L (or $\binom{L}{W}$ if A is rectangular) will also predict correctly the structures of the Householder transformations. The following example illustrates this.

$$A=\begin{pmatrix} \times & \times & 0 & \times & 0 \\ 0 & \times & 0 & \times & \times \\ \times & 0 & \times & 0 & \times \\ 0 & 0 & \times & \times & 0 \\ \times & 0 & \times & \times & \times \\ \times & 0 & 0 & \times & \times \end{pmatrix}.$$

The matrix above has the strong Hall property and

$$A^TA=\begin{pmatrix} \times & \times & \times & \times & \times \\ \times & \times & 0 & \times & \times \\ \times & 0 & \times & \times & \times \\ \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \end{pmatrix} \quad \text{and} \quad L=\begin{pmatrix} \times & & & & \\ \times & \times & & & \\ \times & \times & \times & & \\ \times & \times & \times & \times & \\ \times & \times & \times & \times & \times \end{pmatrix}.$$

Assuming exact numerical cancellation does not occur, it is then easy to verify that the structure of the upper triangular matrix R obtained in the orthogonal reduction of A using Householder transformations is given by

$$R=\begin{pmatrix} \times & \times & \times & \times & \times \\ & \times & \times & \times & \times \\ & & \times & \times & \times \\ & & & \times & \times \\ & & & & \times \end{pmatrix}.$$

Hence, the structure of R is the same as that of L^T. However, the structures of the Householder transformations are given by

$$H=\begin{pmatrix} \times & & & & \\ 0 & \times & & & \\ \times & \times & \times & & \\ 0 & 0 & \times & \times & \\ \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \end{pmatrix}.$$

The structure of $\binom{L}{W}$ is given by

$$\binom{L}{W}=\begin{pmatrix} \times & & & & \\ \times & \times & & & \\ \times & \times & \times & & \\ \times & \times & \times & \times & \\ \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \end{pmatrix};$$

it overestimates the structure of H. Note that the computation can be carried out using Givens rotations as well. If we store each rotation by a single scalar using the technique proposed by Stewart [10] (see § 2), it is easy to see that these scalars will occupy the nonzero positions in H.

Finally, the referee has kindly pointed out the work of Brønlund and Johnsen on generalized Householder transformations [1]. Suppose we have an $m\times k$ matrix A partitioned into the form

$$A=\binom{X}{Y},$$

where X is $k\times k$ and Y is $(m-k)\times k$. Assume A has full rank. A generalized Householder transformation of rank k has the form

$$H=I-ZD^{-1}Z^T,$$

where

$$Z=\binom{X+S}{Y}, \qquad D=S^T(S+X) \quad \text{and} \quad S^TS=A^TA.$$

Then

$$HA=\binom{-S}{O}.$$

Suppose we choose S so that it is upper triangular and $S^TS=A^TA$; that is, S is the Cholesky factor of A^TA. Then H reduces A to upper trapezoidal form and we only have to store S, X, Y, (and perhaps D). Note that *no* fill-in occurs in Y. The "pivot" matrix $D=S^T(S+X)$, which has to be nonsingular, can be chosen easily, as the following discussion shows. Assume for the moment that X is nonsingular. We first reduce the $k\times k$ matrix X to upper triangular form U using orthogonal transformations. Then we adjust the signs of the diagonal of S to be the same as those of U. Thus, $S+U$ is nonsingular (and upper triangular), and we have a triangular decomposition of D. Note that if k is small, the orthogonal decomposition of X can be computed efficiently by treating X as dense.

If X is singular, we may incorporate column pivoting when U is computed, but this causes no essential difficulties.

The idea described above can be used to reduce an $m\times n$ matrix to upper trapezoidal form by partitioning the columns appropriately. While this generalization may turn out to be effective in reducing fill-in, there are two complications in this approach. First, the saving in storage depends on the choice of the column partitioning, and it is not clear how the columns should be partitioned so as to minimize the storage requirement. Second, the symbolic factorization process (for determining the structures of the matrices) appears to be considerably more complicated. These are interesting topics for future research.

REFERENCES

[1] O. E. BRØNLUND AND T. L. JOHNSEN, *QR-factorization of partitioned matrices*, Comput. Meth. Appl. Mech. Eng. 3 (1974), pp. 153-172.

[2] T. F. COLEMAN, A. EDENBRANDT AND J. R. GILBERT, *Predicting fill for sparse orthogonal factorization*, technical report 83-578, Dept. Computer Science, Cornell Univ., Ithaca, NY.

[3] I. S. DUFF, *Analysis of sparse systems*, D.Phil. Thesis, Oxford Univ., Cambridge, 1972.

[4] J. A. GEORGE AND M. T. HEATH, *Solution of sparse linear least squares problems using Givens rotations*, Linear Algebra and Appl., 34 (1980), pp. 69-83.

[5] J. A. GEORGE AND J. W. H. LIU, *Computer Solution of Large Sparse Positive Definite Systems*, Prentice-Hall, Englewood Cliffs, NJ, 1981.

[6] J. A. GEORGE AND E. G. Y. NG, *An implementation of Gaussian elimination with partial pivoting for sparse systems*, this Journal, 6 (1985), pp. 390-402.

[7] ———, *Symbolic factorization for sparse Gaussian elimination with partial pivoting*, technical report CS-84-43, Dept. Computer Science, Univ. of Waterloo, Waterloo.

[8] M. T. HEATH, *Some extensions of an algorithm for sparse linear least squares problems*, this Journal, 3 (1982), pp. 223-237.

[9] G. W. STEWART, *Introduction To Matrix Computations*, Academic Press, New York, 1973.

[10] ———, *The economical storage of plane rotations*, Numer. Math. 25 (1976), pp. 137-138.

[11] M. YANNAKAKIS, *Computing the minimum fill-in is* NP*-complete*, SIAM J. Alg. Disc. Meth., 2 (1981), pp. 77-79.

SIAM J. SCI. STAT. COMPUT.
Vol. 7, No. 2, April 1986

009

CONFIDENCE INTERVALS FOR INEQUALITY-CONSTRAINED LEAST SQUARES PROBLEMS, WITH APPLICATIONS TO ILL-POSED PROBLEMS*

DIANNE P. O'LEARY† AND BERT W. RUST‡

Abstract. Computing confidence intervals for functions $\phi(x) = w^T x$, where $Kx = y + e$ and e is a normally distributed error vector, is a standard problem in multivariate statistics. In this work, we develop an algorithm for solving this problem if additional information, $x \geqq 0$, is given. Applications to estimating solutions to integral equations of the first kind are given.

Key words. confidence intervals, ill-posed problems, integral equations, quadratic programming

1. Introduction.

Consider the linear model

$$Kx = y + e \tag{1.1}$$

where K is a known $m \times n$ matrix, y is an $m \times 1$ vector of observations, x is the unknown solution vector, and e is an unknown $m \times 1$ error vector. The usual case is that $m \geqq n$. In this paper, we develop techniques for obtaining confidence interval estimates on functions $w^T x$ when extra information such as $x \geqq 0$ is known.

When the model above arises from discretization of ill-posed problems, such as integral equations of the first kind, the matrix K is highly ill-conditioned, and small changes in the right-hand side can make large changes in the solution x. Many approaches have been made to finding realistic solutions to this problem. Implicitly or explicitly, all of them make use of some extra information or side conditions, not included in (1.1), in order to eliminate unreasonable solutions. We summarize some of these methods below. Varah [20] gives a more extensive survey of some of the methods and the effects of ill-conditioning.

(a) Regularization techniques [8], [12], [17], [18] favor solutions that are smooth in the sense of having a small value of $\|Lx\|$, where L is a given matrix. Choosing L as a kth difference operator matrix forces the solution to have a small kth derivative. The solution is obtained by solving

$$\min_x \{\|Kx - y\|^2 + \eta \|Lx\|^2\}$$

where η is a given parameter. Large values of η force increasing smoothness; small values allow better fidelity to equation (1.1).

(b) Projection techniques [2], [16] restrict x to lie in some subspace:

$$x = Bu$$

where B is an $n \times p$ matrix of basis vectors, $p < n$. The objective is then to solve

$$\min_u \|KBu - y\|_2^2.$$

The columns of B are chosen to admit only those solutions which are smooth or have some other desirable property. The truncated singular value decomposition is one way

* Received by the editor November 1, 1983, and in revised form October 4, 1984.

† Division 711: Mathematical Analysis, National Bureau of Standards, Gaithersburg, Maryland 20899, and Computer Science Department, University of Maryland, College Park, Maryland 20742.

‡ Division 713: Scientific Computing, National Bureau of Standards, Gaithersburg, Maryland 20899.

to choose B in a data-dependent way in order to force the solution vector to be of small norm.

(c) Side conditions [1], [7], [9], [13] may be used explicitly in order to eliminate undesirable solutions. For instance, in many physical applications the solution is known to be nonnegative, and may be estimated by

$$\min_{x \geqq 0} \|Kx - y\|_2^2.$$

Confidence interval methods use statistical information on the distribution of e in order to estimate the solution. In contrast to the methods above, these procedures give not only an estimate of the true solution x^*, but also a region which contains x^* with a given probability. That is, if a probability α is given and the experiment is repeated many times, then the region will contain the true result $100\alpha\%$ of the time. The vector e is assumed to be normally distributed with mean 0 and nonsingular variance matrix S^2 where S is symmetric. Then the best linear unbiased estimate of the true solution is that vector $\tilde{x}$ which solves

$$\min_x \|Kx - y\|_S^2,$$

where $\|z\|_S^2 = z^T S^{-2} z$. For any linear function $\phi = w^T x$, the best linear unbiased estimate $w^T\tilde{x}$ is normally distributed about the true value $\phi^* = w^T x^*$ with variance $w^T(K^T S^{-2} K)^{-1} w$. For any given probability α, if κ is chosen so that

$$\alpha = \int_{-\kappa}^{\kappa} n(x; 0, 1)\, dx,$$

where $n(x; 0, 1)$ is the probability density function for the standard normal distribution, then the interval $[\phi_{\mathrm{LO}}, \phi^{\mathrm{UP}}]$ is a $100\alpha\%$ confidence interval for ϕ^* when ϕ_{LO} and ϕ^{UP} are given by

$$w^T\tilde{x} \pm \kappa\sqrt{w^T(K^T S^{-2} K)^{-1} w}.$$

We can write ϕ_{LO} and ϕ^{UP} in another way, similar to the forms we will be using in the next section. Using the technique of Lagrange multipliers, it can be shown that

$$\phi_{\mathrm{LO}} = \min_x \{w^T x: \|K(x - \tilde{x})\|_S^2 = \kappa^2\},$$

$$\phi^{\mathrm{UP}} = \max_x \{w^T x: \|K(x - \tilde{x})\|_S^2 = \kappa^2\}.$$

Note that a confidence interval can be given for any component x_i^* of the solution vector by choosing w equal to the ith unit vector.

In this paper we combine the techniques of confidence interval estimation and the use of inequality constraints as side conditions to develop a mathematical framework and some numerical techniques for computing ϕ_{LO} and ϕ^{UP} satisfying:

$$\mathrm{Prob}\{\phi_{\mathrm{LO}} < \phi^* < \phi^{\mathrm{UP}}\} \geqq \alpha$$

when it is known a priori that

$$x^* \geqq 0.$$

This work can be considered an extension of the work of Cope and Rust [7] on confidence interval estimation with inequality constraints. The case of linear equality constraints has been rather well-studied; see, for example, Rao [13]. In this paper,

some statistical background is discussed in § 2, algorithms are presented in § 3, and computational results are given in § 4.

2. Statistical framework. Throughout this work we make the following assumptions:

(a) $Kx = y + e$.

(b) $e \sim N(0, S^2)$, where S is nonsingular symmetric.

(c) It is known a priori that x is nonnegative.

See Bard [3, p. 180] for a discussion of the implication of (c) on the form of the distribution.

In this section we present the results that define the computational task of computing a confidence interval for $\phi = w^T x$.

THEOREM 1. *Under the assumptions above, the probability that ϕ is contained in the interval $[\phi_{\mathrm{LO}}, \phi^{\mathrm{UP}}]$ is greater than or equal to α, where*

$$\phi_{\mathrm{LO}}^{\mathrm{UP}} = \begin{matrix}\max\\ \min\end{matrix} \{w^T x: \|Kx - y\|_S \leqq \mu, x \geqq 0\},$$

$$\operatorname{rank}(K) = q,$$

$$\int_0^{\gamma^2} \chi_q^2(\rho)\, d\rho = \alpha,$$

$$r_0 = \min_x \|Kx - y\|_S^2,$$

$$\mu^2 = r_0 + \gamma^2,$$

and χ_q^2 is the probability density function for the chi-squared distribution with q degrees of freedom.

Proof. If μ^2 is chosen as stated above, then $\{x: \|Kx - y\|_S \leqq \mu\}$ is a $100\alpha\%$ confidence region (ellipsoidal) for the true x^* [15, § 6.4]. (Note that this makes no use of the information that x is nonnegative.) At the same time, the region $\{x: x \geqq 0\}$ is a 100% confidence region. Intersecting the two, we find that $\{x: \|Kx - y\|_S \leqq \mu, x \geqq 0\}$ is a $100\alpha\%$ confidence region for x^*. Thus, for $\phi = w^T x$, a $100\alpha\%$ confidence interval is given by $[\phi_{\mathrm{LO}}, \phi^{\mathrm{UP}}]$, where ϕ_{LO} and ϕ^{UP} are, respectively, the maximum and minimum of ϕ over this set.

In some cases the procedure outlined by this theorem could fail to give useful information. The set over which the maximum and minimum are taken could be null, although a nonnull set always exists for γ large enough. Further, unless w is orthogonal to the null space of K (i.e., unless w satisfies an estimability condition), the interval could be infinite for all α; however, the interval could have one or two finite endpoints even if w does not satisfy this orthogonality condition, and this technique could give information on ϕ even in these cases where, without the side conditions, a minimum variance unbiased estimator fails to exist.

Note that if several vectors w_ν, $\nu = 1, \cdots, N$ are given, the procedure above gives *simultaneous* confidence intervals for all $\phi_\nu = w_\nu^T x$, since the region over which the maximum and minimum are taken is a $100\alpha\%$ confidence region for x^*; i.e., the probability that all $w_\nu^T x$ are in the computed intervals is $100\alpha\%$.

LEMMA 1. *Let*

$$L(\phi) = \min_x \{\|Kx - y\|_S^2: x \geqq 0,\ w^T x = \phi\}.$$

Then $L(\phi)$ is unimodal, piecewise quadratic, differentiable, and convex.

Proof. $L(\phi)$ is the objective function value for a convex parametric quadratic programming problem. This is the essence of the argument proving the lemma. A more detailed discussion follows, since this argument forms the basis for algorithms in the next section.

Let I be a set of indices, let $\bar{I}$ be its complement in the set $\{1, 2, \cdots, n\}$, and hold x_i at zero for $i \in \bar{I}$. We partition the x vector as $(x_I^T, x_{\bar{I}}^T)$, and partition K and w to conform to it. Then, using Lagrange multipliers, the solution to the problem

$$L(\phi, I) = \min_x \{\|Kx - y\|_S^2 : x_{\bar{I}} = 0,\ w^T x = \phi\} \tag{2.1}$$

is determined by

$$\begin{bmatrix} K_I^T S^{-2} K_I & w_I \\ w_I^T & 0 \end{bmatrix} \begin{bmatrix} x_I \\ \lambda \end{bmatrix} = \begin{bmatrix} K_I^T S^{-2} y \\ \phi \end{bmatrix}. \tag{2.2}$$

If K_I is not full column rank, then the solution may be nonunique, but we will take the solution $x_I = x_I^P + x_I^N$ with the smallest null space component x_I^N needed to keep $x_I \geqq 0$. For a given index set I, x_I^P is a linear function of ϕ, and $L(\phi, I)$ is convex and quadratic in ϕ. Furthermore, it is clear that

$$L(\phi) = \min_I \{L(\phi, I) : x_I \geqq 0\}.$$

The vector x for which this minimum is achieved is a continuous function of ϕ, and therefore $L(\phi)$ is a continuous and differentiable function of ϕ. Let $\min_x \{\|Kx - y\|_S^2 : x \geqq 0\}$ be attained at $\tilde{x}$. (If the minimum is achieved at a set of points, let $\tilde{x}$ be any one for which $w^T\tilde{x}$ is minimal.) Let $\tilde{\phi} = w^T\tilde{x}$. Then $L(\phi)$ has a global minimum at $\tilde{\phi}$. A sketch of $L(\phi)$ is given in Fig. 1. In the following discussion we assume that a designated point $\hat{\phi}$ is greater than $\tilde{\phi}$. The argument for $\hat{\phi} < \tilde{\phi}$ is analogous.

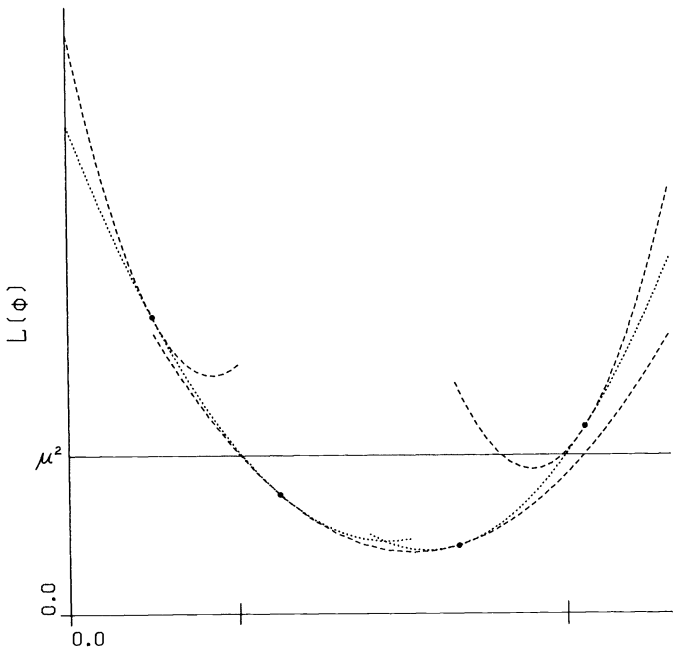


FIG. 1. *The $L(\phi)$ curve, showing various quadratic segments. The points ϕ_{LO} and ϕ^{UP} are indicated on the ϕ axis.*

Let $\hat{\phi} > \tilde{\phi}$ be the first point where an index j needs to be dropped from the index set I (because x_j has been driven to zero and would go negative if it were kept in the set), or a variable needs to be added to the index set (because using it produces a smaller function value without violating any constraint). Then the set of equations defining x as a function of ϕ has one fewer or one extra degree of freedom, but $L(\phi)$ over the next interval (i.e., until the index set changes again) is still convex and quadratic. Let I denote the previous index set, and $\hat{I}$ the index set for $\phi \geqq \hat{\phi}$. Since $dL(\hat{\phi}, \hat{I})/d\phi = dL(\hat{\phi}, I)/d\phi$ and $dL(\hat{\phi}, I)/d\phi \geqq 0$, then $L(\phi, \hat{I})$ must be monotone nondecreasing for $\phi \geqq \hat{\phi}$.

This argument can be repeated on all intervals beyond this one, thus establishing the result.

THEOREM 2. *The values $\phi_{\mathrm{LO}}^{\mathrm{UP}}$ are defined by the two extreme roots of $L(\phi) - \mu^2 = 0$, where $L(\phi) = \min_x \{\|Kx - y\|_S^2 : x \geqq 0,\ w^T x = \phi\}$.*

Proof. We sketch the proof for ϕ_{LO}; the discussion for ϕ^{UP} is analogous.

Let $\phi_{\mathrm{LO}} = \min_x \{w^T x : \|Kx - y\|_S^2 \leqq \mu^2,\ x \geqq 0\}$ and let x_{LO} be an x vector for which the minimum is attained. Similarly, let $\bar{\phi}_{\mathrm{LO}}$ and $\bar{x}_{\mathrm{LO}}$ be the corresponding minimal root and x values for $L(\phi) - \mu^2 = 0$. Then

$$L(\bar{\phi}_{\mathrm{LO}}) = \|K\bar{x}_{\mathrm{LO}} - y\|_S^2 = \mu^2, \quad \bar{x}_{\mathrm{LO}} \geqq 0, \quad \text{and } w^T \bar{x}_{\mathrm{LO}} = \bar{\phi}_{\mathrm{LO}}.$$

Therefore,

$$\phi_{\mathrm{LO}} = \min_x \{w^T x : \|Kx - y\|_S^2 \leqq \mu^2,\ x \geqq 0\} \leqq \bar{\phi}_{\mathrm{LO}},$$

since $\bar{x}_{\mathrm{LO}}$ is one of the candidate points in the minimization.

On the other hand,

$$L(\phi_{\mathrm{LO}}) = \min_x \{\|Kx - y\|_S^2 : x \geqq 0,\ w^T x = \phi_{\mathrm{LO}}\} \leqq \|Kx_{\mathrm{LO}} - y\|_S^2 \leqq \mu^2,$$

since x_{LO} is one of the candidates in the minimization. Thus $L(\phi_{\mathrm{LO}}) \leqq \mu^2$, and, since by Lemma 1, $L(\phi)$ is unimodal and convex, this implies that $\phi_{\mathrm{LO}} \geqq \bar{\phi}_{\mathrm{LO}}$. Thus $\phi_{\mathrm{LO}} = \bar{\phi}_{\mathrm{LO}}$.

3. Algorithms for constrained interval estimation. Two approaches to solving $L(\phi) = \mu^2$ are discussed in this section. Methods for finding ϕ_{LO} are discussed; methods for computing ϕ^{UP} are analogous. The algorithms are written under the assumption that K has full column rank. Modifications for the rank deficient case are possible but tedious.

3.1. Tracing the $L(\phi)$ curve. Conceptually, the simplest approach for finding ϕ_{LO} is to start from a known lower bound ϕ_0 (or some other convenient point) and follow the $L(\phi)$ curve until it crosses μ^2. This is a parametric quadratic programming problem as ϕ varies:

$$\min_x \{\|Kx - y\|_S^2 : x \geqq 0,\ w^T x = \phi\}. \tag{3.1}$$

The optimal values of the objective function form a piecewise quadratic function of ϕ; the nodes occur where variables x change from zero to positive or vice versa. The algorithm is as follows:

ALGORITHM TRACE

1. Solve (3.1) for $\phi = \phi_0$ and set $\phi_{\mathrm{NODE}} = \phi_{\mathrm{NEW}} = \phi_0$.
2. Until $L(\phi_{\mathrm{NEW}}) < \mu^2$

   Let $\phi_{\mathrm{NODE}} = \phi_{\mathrm{NEW}}$.

Find the next node, the minimum value $\phi_{\text{NEW}} > \phi_{\text{NODE}}$ such that a variable x_j changes from zero to positive or vice versa.

3. The value ϕ_{LO} is now known to satisfy $\phi_{\text{NODE}} \leqq \phi_{\text{LO}} < \phi_{\text{NEW}}$. It can be determined by solving a quadratic equation.

We now provide more details on each step.

Step 1. The problem (3.1) can be solved, for example, by the code WNNLS of Haskell and Hanson [10], but Step 2 is done more efficiently using an algorithm from which matrix factorizations are accessible. We will present such an algorithm in § 3.2.

Step 2. By equation (2.2), the value of x over a particular quadratic segment can be determined by

$$A\begin{bmatrix} x_I \\ \lambda \end{bmatrix} = \begin{bmatrix} K_I^T S^{-2} y \\ \phi \end{bmatrix},$$

$$x_{\bar{I}} = 0$$

where

$$A = \begin{bmatrix} K_I^T S^{-2} K_I & w_I \\ w_I^T & 0 \end{bmatrix}.$$

Thus

$$\begin{bmatrix} x_I(\phi_{\text{NODE}} + \Delta\phi) \\ \lambda(\phi_{\text{NODE}} + \Delta\phi) \end{bmatrix} = \begin{bmatrix} x_I(\phi_{\text{NODE}}) \\ \lambda(\phi_{\text{NODE}}) \end{bmatrix} + \Delta\phi A^{-1} \begin{bmatrix} 0 \\ 1 \end{bmatrix},$$

and it is easy to find the minimum positive $\Delta\phi$ which drives a variable in x_I to zero. Similarly, the minimum positive $\Delta\phi$ for which a variable in $x_{\bar{I}}$ becomes positive can also be determined. Note that the optimality conditions for (3.1) are

$$\begin{bmatrix} K^T S^{-2} K & w & I \\ w^T & 0 & 0 \end{bmatrix} \begin{bmatrix} x \\ \lambda \\ \begin{bmatrix} 0 \\ \delta_{\bar{I}} \end{bmatrix} \end{bmatrix} = \begin{bmatrix} K^T S^{-2} y \\ \phi \end{bmatrix},$$

$$x_i \delta_i = 0, \qquad i \in I,$$

$$x_I \geqq 0, \qquad x_{\bar{I}} = 0,$$

$$\delta \geqq 0.$$

Thus a variable in $\bar{I}$ is released from zero when the corresponding δ value is driven to zero, and the first equation block above can be used to calculate $\delta_{\bar{I}}$ as a function of ϕ.

Step 3. To find ϕ_{LO}, note that

$$L(\phi) = (Kx(\phi) - y)^T S^{-2} (Kx(\phi) - y)$$

and, from Step 2,

$$x(\phi) = x_{\text{NODE}} + \Delta\phi d$$

where d is a computable column vector. Thus,

$$L(\phi) = L(\phi_{\text{NODE}}) + 2\Delta\phi d^T K^T S^{-2} (Kx_{\text{NODE}} - y) + (\Delta\phi)^2 d^T K^T S^{-2} K d,$$

and solving $L(\phi) = \mu^2$ just involves finding the root of the quadratic equation in the interval $[\phi_{\text{NODE}}, \phi_{\text{NEW}}]$.

Algorithm Trace is appealing in its simplicity, and is quite efficient and adequate for well-conditioned problems. In practice, however, it does not work well on the ill-conditioned problems of interest to us. Degeneracies are often encountered in which the index set I makes several changes with no appreciable change in ϕ. Computationally, this can cause an infinite loop, with variables entering and leaving the set I without changing L or ϕ_{NODE}. A more robust although somewhat more expensive approach is given in the next section.

3.2. Rootfinding for $L(\phi)=\mu^2$. This approach amounts to applying a nonlinear equation solver to $L(\phi)=\mu^2$, solving a quadratic programming problem whenever a function evaluation is needed. The simplest example is:

ALGORITHM BRENT-WNNLS

Brent's program ZEROIN [5] is used as the rootfinder, and Haskell and Hanson's WNNLS [10] is used to evaluate $L(\phi)$.

This algorithm has the advantage of being easy to program, but loses efficiency in two ways: ZEROIN does not take advantage of the piecewise quadratic nature of the function, and WNNLS solves each problem starting with the guess $x=0$, taking no advantage of previous information. Efficiency can be gained by tailoring the rootfinder and the quadratic program solver to this application. We discuss these two alternate algorithms in turn.

Hanson and Haskell's WNNLS, based on Lawson and Hanson's NNLS [11], solves the problem

$$\min_x \|Kx-y\|_S^2, \qquad x \geqq 0, \quad w^T x=\phi$$

by converting it to

$$\min_x \left\| \begin{bmatrix} \alpha w^T \\ S^{-1}K \end{bmatrix} x - \begin{bmatrix} \alpha\phi \\ S^{-1}y \end{bmatrix} \right\|^2, \qquad x \geqq 0$$

where α is a large weight, related to the wordlength of the machine, so that the equation $w^T x=\phi$ is satisfied to within a very small tolerance. We modify the algorithm NNLS to

1) Start from the solution to the previous problem rather than from $x=0$.
2) Compute and save the orthogonal factorization of the least squares matrix in a form usable for side calculations in the rootfinder.

A description of the algorithm follows. The numbering of the steps parallels Lawson and Hanson's description in [11, p. 161].

ALGORITHM LS.

Saved from previous solution: the index set I of nonzero x variables and a factorization

$$\begin{bmatrix} \alpha w_I^T \\ S^{-1}K_I \end{bmatrix} = QR, \quad \text{such that } Q^TQ=I, \quad \text{and } R=\text{upper triangular}$$

1. Compute x from

$$Rx_I = Q^T \begin{bmatrix} \alpha\phi \\ S^{-1}y \end{bmatrix}, \qquad x_{\bar{I}}=0.$$

If any $x_j<0$, drop such indices j from I and such columns j from the QR factors and repeat this step.

2. Compute the residual r and negative half gradient g:

$$r = \begin{bmatrix} \alpha\phi \\ S^{-1}y \end{bmatrix} - \begin{bmatrix} \alpha w^T \\ S^{-1}K \end{bmatrix} x, \qquad g = [\alpha w, K^T S^{-1}]r.$$

3. If $g_{\bar{I}} \leqq 0$ or if $\bar{I}$ is null, then the optimal solution has been found.
4. Let $g_t = \max_{j \in \bar{I}} g_j$.
5. $I = I \cup \{t\}$. Add column t to the QR factors.
6. Compute z from

$$Rz_I = Q^T \begin{bmatrix} \alpha\phi \\ S^{-1}y \end{bmatrix}, \qquad z_{\bar{I}} = 0.$$

7. If $z_I \geqq 0$ then let $x = z$ and go to Step 2.
8-9. Otherwise, a variable needs to be dropped. Find an index $q \in I$ such that

$$\hat{\beta} = \frac{x_q}{x_q - z_q} = \min_{\substack{z_j \leqq 0 \\ j \in I}} \frac{x_j}{x_j - z_j}.$$

10. Let $x = x + \hat{\beta}(z - x)$.
11. Let $I = I \backslash \{j: x_j = 0\}$. Drop column q (and any others for which the x component is zero) from the QR factors. Go to 6.

The QR updating and downdating in LS can be performed using a modified Gram-Schmidt algorithm.

Tailoring the rootfinder to the piecewise quadratic nature of the $L(\phi)$ curve also enhances efficiency. We present an algorithm which keeps the root bracketed but bases its new prediction on a quadratic extrapolation of the function.

ALGORITHM BRACKET

Given: an initial interval $[\phi_a, \phi_b]$ containing a single root, and two convergence tolerances ε_1 and ε_2. Let $\Delta\phi = 0$ and let $\phi = \phi_b$ if looking for ϕ_{LO} and $\phi = \phi_a$ if looking for ϕ^{UP}. Until $|(\phi_a - \phi_b)/\phi| < \varepsilon_1$ or $|(L(\phi) - \mu^2)/\mu^2| < \varepsilon_2$, perform steps 1 through 5.

1. Let $\Delta\phi_{\text{OLD}} = \Delta\phi$.
2. Make a prediction of the root, $\phi + \Delta\phi$, based on the current quadratic segment: *Find* $\Delta\phi_T$ *such that* $L(\phi + \Delta\phi_T) = \mu^2$ *assuming that no basis changes occur.* This involves solving a quadratic equation for $\Delta\phi_T$. Set $\Delta\phi = \Delta\phi_T$.
3. If $\Delta\phi_T$ is too small, increase it:

$$\textit{If } |\Delta\phi_T| < .9\varepsilon_1|\phi| \textit{ then } \Delta\phi_T = .9\varepsilon_1|\phi| \operatorname{sgn}(\Delta\phi_T).$$

4. Reject the quadratic prediction if
   a) convergence is linear (i.e., $\Delta\phi_{\text{OLD}}/\Delta\phi$ is close to 1), or
   b) the new prediction $\phi + \Delta\phi_T$ does not fall within the current interval, or
   c) the change in ϕ is small and both the new and the old ϕ values are on the same side of the root (i.e., $\Delta\phi \neq \Delta\phi_T$ and $(L(\phi) - \mu^2) \times (L(\phi + \Delta\phi_T) - \mu^2) > 0$).

   In these cases, use the bisection algorithm (i.e., set $\phi = (\phi_a + \phi_b)/2$). Otherwise, ϕ is set equal to the quadratic prediction $\phi = \phi + \Delta\phi_T$.
5. Evaluate $L(\phi)$. Shorten the interval known to contain the root: if $(L(\phi_a) - \mu^2)(L(\phi) - \mu^2) > 0$ then $\phi_a = \phi$; otherwise, $\phi_b = \phi$.

Algorithm Bracket-LS has been successful on many ill-conditioned test problems. Examples are given in the next section.

4. Numerical results. We present the results of numerical experiments on two problems. The first is a standard test problem, and we use it to compare the performance of several algorithms. The second is a problem in radiation physics, presented to demonstrate the practical utility of the Bracket-LS algorithm.

Example 1. The Phillips equation [12] is as follows:

$$\int_{-6}^{6} K(t, s)x(s)\,ds = y(t), \qquad |t| \leqq 6,$$

where

$$K(t, s) = \begin{cases} 1+\cos(\pi(s-t)/3), & |s-t| \leqq 3, \quad |t| \leqq 6, \\ 0, & |s-t| \geqq 3, \quad |t| \leqq 6, \end{cases}$$

and

$$y(t) = \begin{cases} (6-|t|)[1+.5\cos(\pi t/3)]+9/(2\pi)\sin(\pi|t|/3), & |t| \leqq 6, \\ 0, & |t| > 6. \end{cases}$$

The solution is

$$x(s) = \begin{cases} 1+\cos(\pi s/3), & |s| \leqq 3, \\ 0, & |s| > 3. \end{cases}$$

We approximate this problem by

$$Kx = y + e$$

where $y_i = y(t_i)$ and the t_i are midpoints of 78 intervals of equal length in the interval $[-6, 6]$, and the quadrature is performed by the trapezoidal rule with 48 intervals of equal length on $[-3, 3]$. Then K is a matrix of dimension 78×49 and $r_0 = 0$. The diagonal elements of the matrix S were taken to be $s_{ii} = 10^{-4} y_i$ and μ was taken to be 9.792, corresponding to $\alpha = .9999$. The a priori information is that the x vector is nonnegative, and the vectors w are chosen to give nonoverlapping three point averages of the x function using weights (1/4, 1/2, 1/4) for each triplet of adjacent components of the x vector.

The initial interval was determined in two ways:

(1) The "naive interval" is

$$0 \leqq x_j \leqq \min_{1 \leqq i \leqq m} \frac{y_i + \mu s_i}{K_{ij}}, \qquad j = 1, \cdots, n.$$

For matrices with nonnegative components, this defines a box containing the intersection of the ellipsoid $\{x: \|Kx-y\|_S \leqq \mu\}$ with the positive orthant [7].

(2) The "FERDIT interval" is the interval determined by one iteration of the algorithm described in [7]. This is a method for determining suboptimal confidence intervals by minimizing or maximizing $w^T x$ for x in an ellipsoidal region containing the intersection of $\{x: \|Kx-y\|_S \leqq \mu\}$ and the naive interval above.

We show the results of several experiments. Figure 2 shows the confidence intervals obtained by standard methods, without the inequality constraints. The lengths are of order 10^4, reflecting the 10^5 condition number of the rank 42 matrix K, although the true solution is of order 1. Figure 3 shows confidence intervals obtained by requiring that the solution be symmetric in its variable. This corresponds to adding equality constraints and reduces the condition number to 10^3. The confidence intervals now are of order 10^3. Even the naive bounds give more information than this: the solution

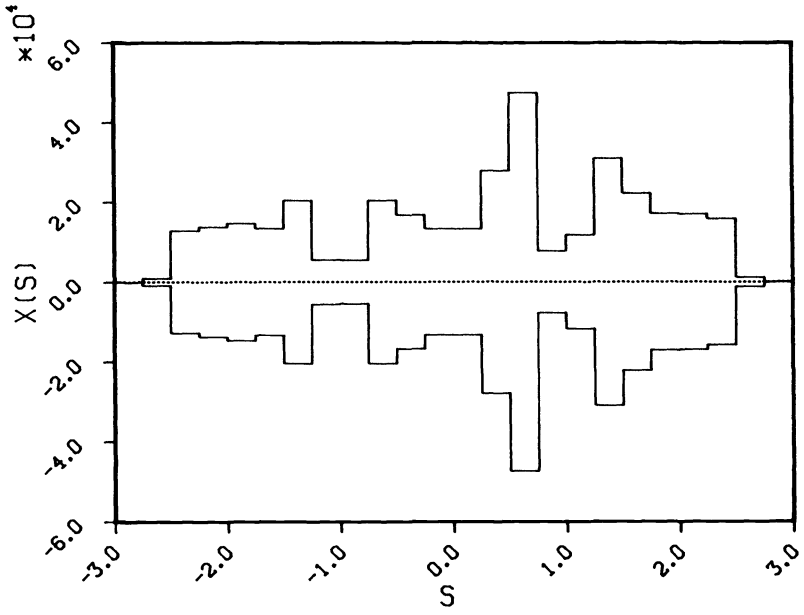


FIG. 2. *Standard confidence intervals for the* 78×49 *Phillips problem. The dotted line plots sample values of the true solution to the continuous problem.*

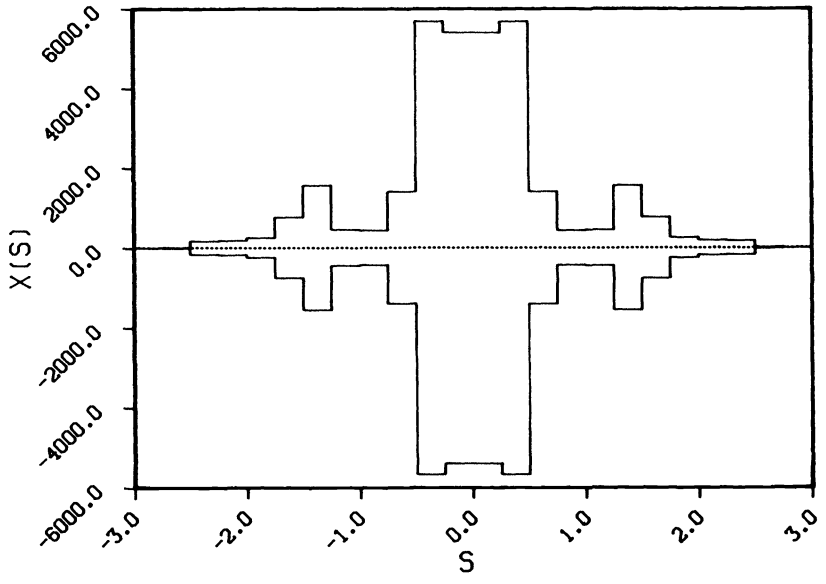


FIG. 3. *Standard confidence intervals for the* 78×49 *Phillips problem, using the constraint that x is symmetric around the point* $s = 0$. *The true solution is represented by the dotted line.*

is bounded by 0 and 35. Figure 4 gives the estimated 99.99% confidence bounds using the inequality constraints. The intervals now have a maximum length of less than 2. Figure 5 presents the bounds obtained using the inequality constraints and the symmetry condition.

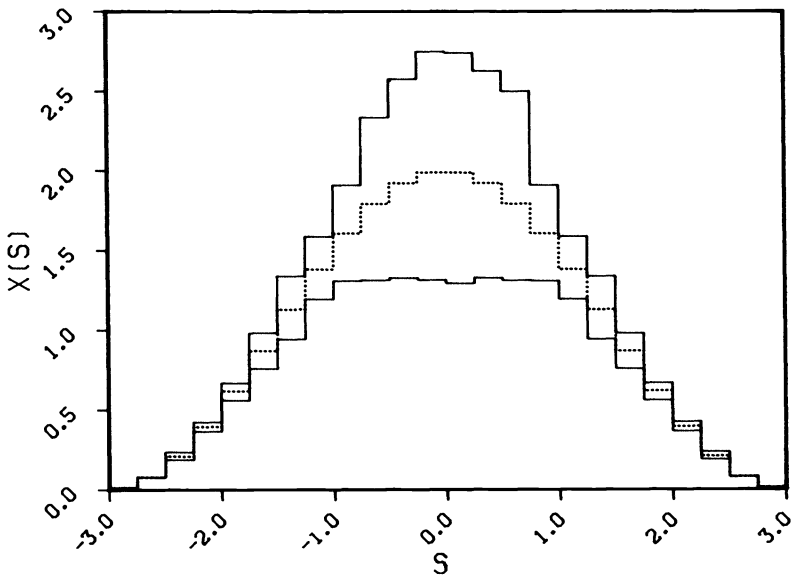


FIG. 4. *Confidence intervals for the* 78×49 *Phillips problem, using the constraint that x is nonnegative. The true solution is represented by the dotted line.*

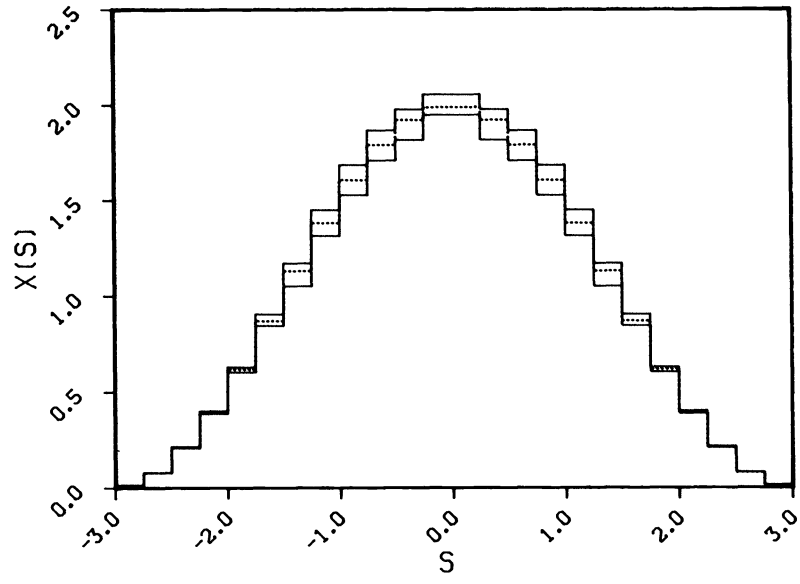


FIG. 5. *Confidence intervals for the* 78×49 *Phillips problem, using the constraints that x is nonnegative and symmetric around the point* $s = 0$. *The true solution is represented by the dotted line.*

Table 1 gives the performance statistics for various algorithms used to compute the results shown in Fig. 4. The runs were performed in FORTRAN single precision on a UNIVAC 1100/82. The table indicates the number of digits accuracy requested in the ϕ bounds, the run time (including FERDIT, if used), and the "basis changes", i.e., the number of changes of the index set I in Algorithm LS. The FERDIT iteration

TABLE 1
Algorithm performance on example 1.

Algorithm	Digits accuracy	Time (min.)	Basis changes
Brent-NNLS			
naive interval	4	25.0	53,314
Brent-LS			
naive interval	4	22.9	18,704
naive interval	3	13.9	10,993
Bracket-LS			
naive interval	4	8.9	6,220
naive interval	3	8.8	6,194
FERDIT interval	4	4.8	2,081
FERDIT interval	3	4.6	1,982

took approximately .6 minutes on this problem and gave upper bounds within a factor of 3.2 of the true three point averages, but most lower bounds were 0. The best algorithm, Bracket-LS with FERDIT, was 5 times faster than Brent-NNLS. Most of the improvement comes from the use of Bracket rather than Brent, and the use of initial interval information from FERDIT.

On other test problems, FERDIT continued to prove useful. If the initial bounds were close to optimal, FERDIT sometimes increased the cost of the algorithm, but on problems where the initial bounds were crude, it often led to large savings.

Example 2. We now consider the spectrum unfolding problem which arises in radiation physics. The integral equation is

$$\int_{E_{\mathrm{LO}}}^{E_{\mathrm{UP}}} K_i(E)x(E)\,dE = y_i + \varepsilon_i, \qquad i = 1, 2, \cdots, m,$$

where $x(E)$ is an unknown energy spectrum, y_i is the number of particles or photons counted in channel i of a multi-channel analyzer, and $K_i(E)$ is the spectrometer's energy response for channel i, (i.e. $K_i(E)\,dE$ is the probability that a particle or photon in the energy range $E \pm \frac{1}{2}\,dE$ will produce a pulse which gives a count in channel i). Figures 6a and 6b show two views of the response function for an NE-213 neutron spectrometer which has been described in detail by Verbinski et al. [21] and by Burrus and Verbinski [6]. The figures show a piecewise linear discretization K_{ij} to be used with the finite approximation equations

$$\sum_{j=1}^{n} K_{ij}x_j = y_i + \bar{\varepsilon}_i, \qquad i = 1, 2, \cdots, m,$$

where x_j is the total number of neutrons in the jth energy interval $(E_j \pm \frac{1}{2}\,dE_j)$. The values of m and n were taken to be 113 and 77 respectively, but in the interest of graphical clarity, the figure shows only every 3rd ordinate in each of the two abscissa directions. In order to show the structure for the higher energies, we have plotted the base 10 logarithm of $(1+K)$ rather than the response function K itself. The mesh spacings for both energy and pulse height vary over their entire ranges, with narrower meshes being used for lower energies and pulse heights.

Ideally a response function should be a narrow, symmetric, sharply-peaked ridge centered along some linear relation between energy and pulse height. It is clear from the figures that the NE-213 spectrometer response function has none of these properties and that the measured pulse height spectrum will be a poor representation of the actual

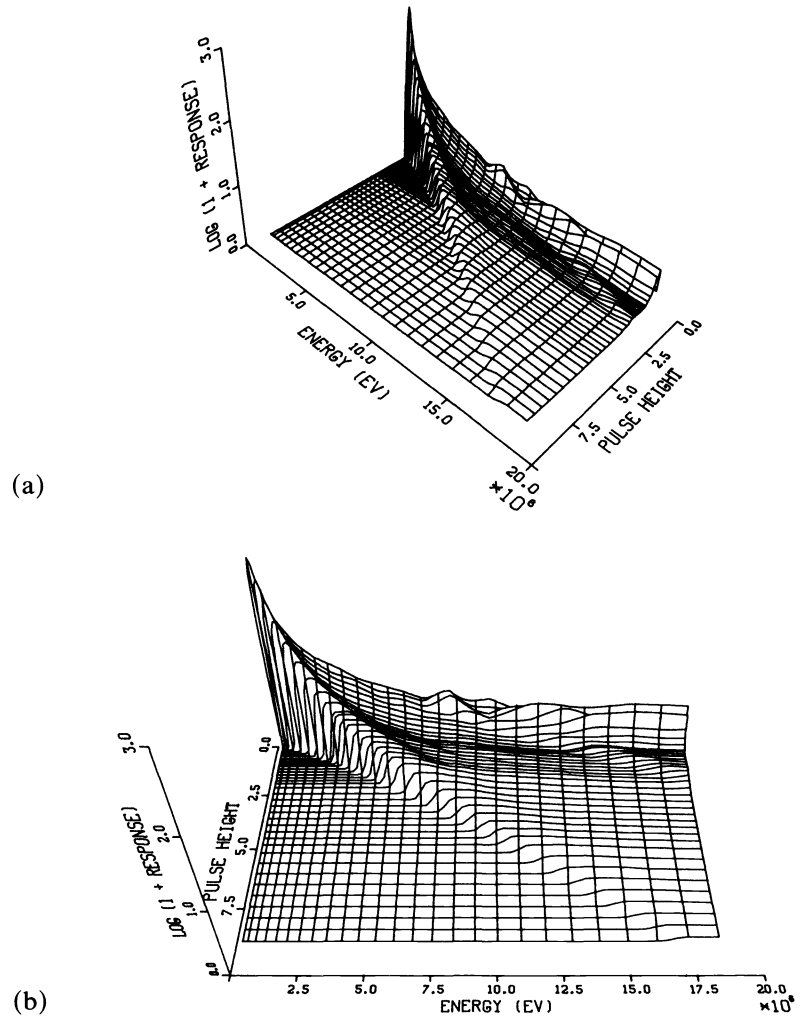


FIG. 6. *The instrument response function which provides data for the matrix K for the second example.*

energy spectrum. Figure 7 shows the plus and minus one standard deviation bounds for the measured spectrum of monoenergetic neutrons produced by the nuclear reaction T(d, n)^{4}He, (i.e. tritium nuclei bombarded with deuterons to produce helium nuclei and the neutrons whose spectrum was measured). It is assumed that there are no correlations between the numbers counted in separate channels and that the number in each channel is normally distributed. The variance matrix S is then an $m \times m$ diagonal matrix whose diagonal elements are one-half the widths shown in the figure. It is assumed that the standard deviations are known exactly so that chi-square statistics can be used rather than F-distribution statistics. In reality the standard deviations are not really known exactly but the estimates for most channels are very accurate because they are based on large numbers of counts in each channel. The standard deviation estimate for the number counted in each channel was the square root of the number counted (see Trumpler and Weaver [19, pp. 166–169] except that in channels containing only a few counts the estimates were chosen to be larger than this in order to assure that they were conservative. Figure 7 shows not the actual raw counts but rather a

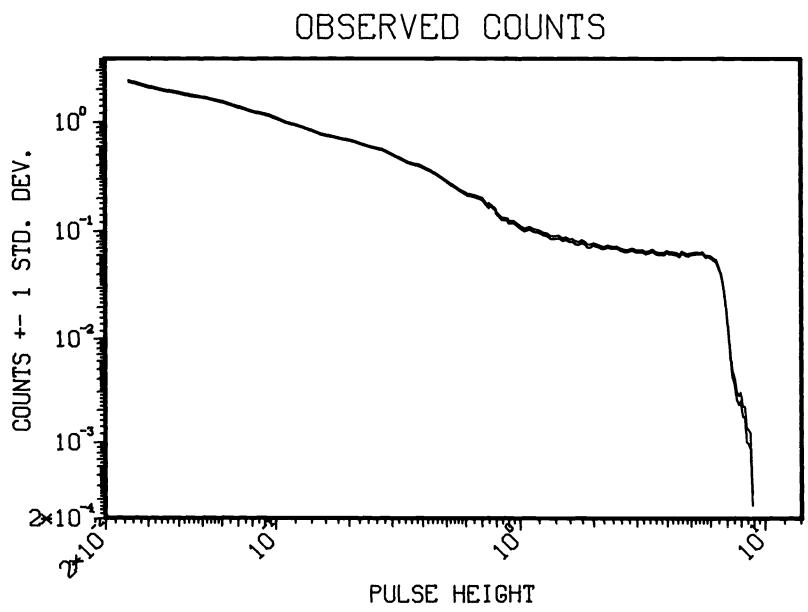


FIG. 7. *The instrument output, which provides data for the standard deviation matrix and the right-hand side function y for the second example.*

normalization which was required to correct for instrument gain effects that need not concern us here.

Since it is hopeless to try to estimate $x(E)$ at each energy E we seek instead to estimate averages of $x(E)$ over various energy ranges. A common practice is to estimate weighted averages with Gaussian weighting functions. Accordingly we seek estimates of the quantities

$$\phi_k(E) = \int_{-\infty}^{\infty} w_k(E)x(E)\,dE, \qquad k = 1, 2, \cdots, p,$$

where

$$w_k(E) = \frac{1}{\sigma_k\sqrt{2\pi}} \exp\left[-\frac{1}{2\sigma_k^2}(E - E_k)^2\right], \qquad k = 1, 2, \cdots, p,$$

with the peak energies E_k chosen to cover the same energy range as the response functions and each peak width σ_k chosen to give the energy resolution desired in the neighborhood of E_k. The weighting functions $w_k(E)$ are called window functions and are tabulated at the same energy mesh points as the instrument response functions. The result is a set of window vectors

$$w_k = (w_{k1}, w_{k2}, \cdots, w_{kn})^T, \qquad k = 1, 2, \cdots, p,$$

and the "unfolded spectrum" is the set of p points (E_k, ϕ_k) where

$$\phi_k = \sum_{j=1}^{n} w_{kj}x_j = w_k^T x, \qquad k = 1, 2, \cdots, p.$$

Note that this procedure, in effect, replaces the real instrument response functions with the window functions so that the unfolded spectrum represents the measurements that would have been obtained using an instrument whose response functions were the latter rather than the former. The set of window vectors actually used is shown in

Fig. 8. The total number of window vectors was 105, with each vector being tabulated at $n = 77$ points. The windows widths σ_k were all chosen much smaller than the total energy range (0–20 Mev) so that most of the 77 elements in each vector were negligible and therefore set to zero. Note that the higher energy windows are wider than those at lower energies. The widths chosen reflect the experimentalist's estimate of the degree of energy resolution obtainable at each energy. All of the windows are normalized so that

$$\int_{-\infty}^{\infty} w_k(E)\, dE = 1.0, \qquad k = 1, 2, \cdots, p.$$

Note that we have again plotted $\log_{10}(1 + w_{kj})$ rather than the w_{kj} themselves.

To obtain an estimate of the "unfolded spectrum" we first computed the confidence intervals by the standard method. The results are shown in Fig. 9, which is a plot of

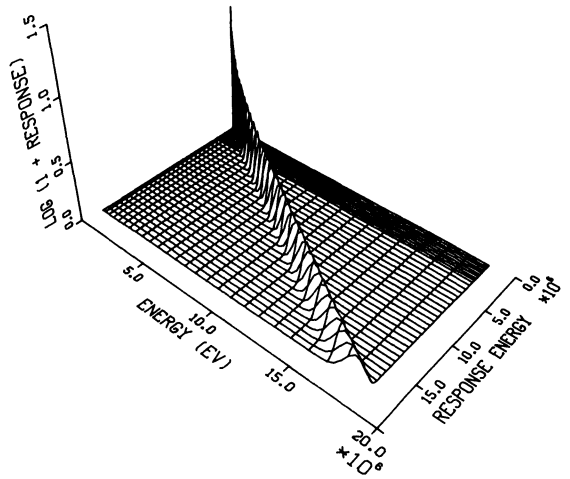


FIG. 8. *The window vectors w for the second example.*

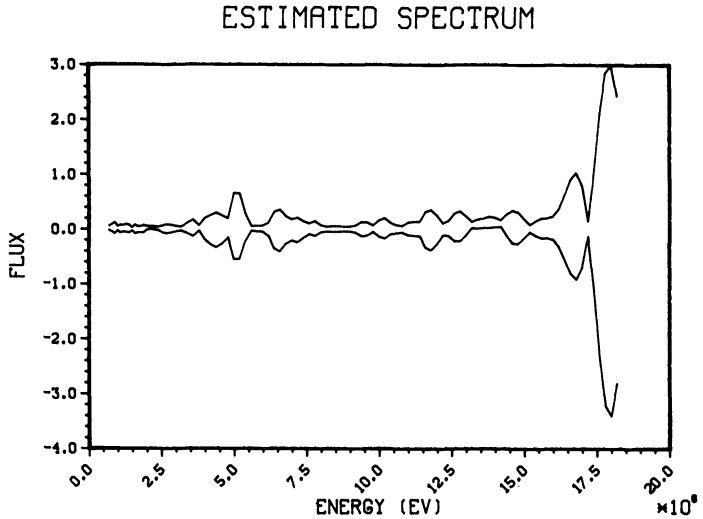


FIG. 9. *Standard confidence intervals for the second example.*

the bounds ϕ_k^{LO} and ϕ_k^{UP} as a function of E_k, $k = 1, 2, \cdots, 105$. The value of μ^2 was chosen so that each of the intervals $[\phi_k^{LO}, \phi_k^{UP}]$ was an $\alpha = 99.99\%$ confidence interval. The value of $r_0 = \min_x \|Kx - y\|_S^2$ was 38.048, and we calculated γ^2 from the asymptotic formula

$$\gamma^2 = \tfrac{1}{2}[\kappa_\alpha + \sqrt{2N-1}]^2$$

where N is the number of degrees of freedom and κ_α is the percentage point of the cumulative normal distribution corresponding to confidence level α [4, p. 293]. For the present problem, the value of N should ideally be the rank of the response matrix but this is difficult to determine reliably, so we used the most conservative estimate $N = n = 77$. We used $\kappa_\alpha = 4.0$ which gives $\alpha = 0.9999$. The value of μ^2 was then computed by $\mu^2 = r_0 + \gamma^2$. In Fig. 9 the computed bounds ϕ_k^{LO} and ϕ_k^{UP} were joined by straight line segments to form an estimated uncertainty band.

An experimentalist would expect a peak in the spectrum between 13 and 14 Mev, but there is no evidence of this in the standard confidence interval estimates. We applied Algorithm Bracket-LS to each of the window vectors, yielding the results shown in Fig. 10. The spectrum in Fig. 10 is dominated by a single peak centered at about 13.8 Mev and with a full width at half maximum of about 1.5 Mev. Part of this width (about 0.6 Mev) can be attributed to the inherent resolution limit of the instrument, but the remainder arises from the choice of the window function widths σ_k for the windows centered in the neighborhood of 14 Mev. The width of the peak can be reduced by choosing smaller window widths σ_k, but this procedure also produces wider confidence intervals $[\phi_k^{LO}, \phi_k^{UP}]$. Thus, in choosing window widths, it is necessary to balance statistical uncertainty against energy resolution. If the windows are too wide, details in the spectrum are smeared out and lost. If they are too small the widths of the confidence intervals become excessively large. Three or four digit accuracy can be achieved in 22.1 minutes using FERDIT and Bracket-LS, or in 26.7 minutes without FERDIT.

Acknowledgments. We are grateful to Claire Wolfe for preparing the three dimensional plots and to G. W. Stewart for providing Gram-Schmidt software.

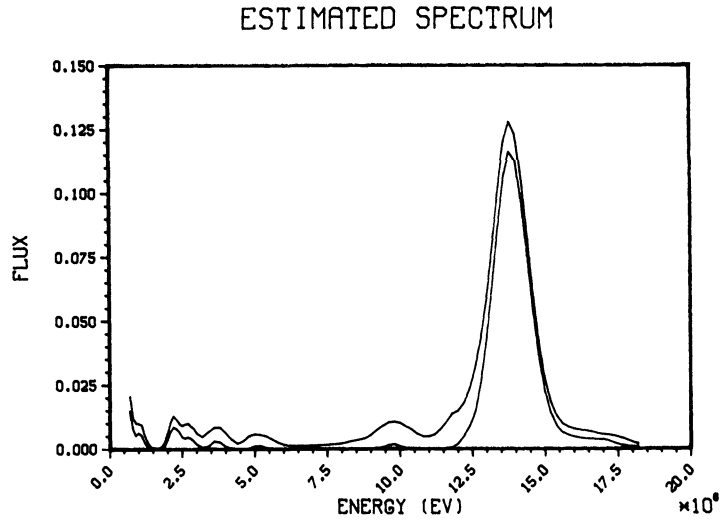


FIG. 10. *Confidence intervals for the second example, using the constraint that x is nonnegative.*

REFERENCES

[1] A. M. AURELA AND J. TORSTI, *Stabilization of unstable resolution correction problems*, Ann. Acad. Sci. Fennical Series, A 255 (1967), pp. 1-17.

[2] C. T. H. BAKER, L. FOX, D. F. MAYERS AND K. WRIGHT, *Numerical solution of Fredholm integral equations of first kind*, Comput. J., 7 (1964), pp. 141-148.

[3] Y. BARD, *Nonlinear Parameter Estimation*, Academic Press, New York, 1974.

[4] W. H. BEYER, *Handbook of Tables for Probability and Statistics*, 2nd Edition, CRC Press, Boca Raton, FL, 1979.

[5] R. P. BRENT, *An algorithm with guaranteed convergence for finding a zero of a function*, Comput. J., 14 (1971), pp. 422-425.

[6] W. R. BURRUS AND V. V. VERBINSKI, *Fast-neutron spectroscopy with thick organic scintillators*, Nuclear Instruments and Methods, 67 (1969), pp. 181-196.

[7] J. E. COPE AND B. W. RUST, *Constrained least squares interval estimation*, this Journal, 6 (1985), pp. 670-683.

[8] M. FOSTER, *An application of the Wiener-Kolmogorov smoothing theory to matrix inversion*, J. Soc. Indust. Appl. Math, 9 (1961), pp. 387-392.

[9] R. J. HANSON, *Integral equations of immunology*, Comm. Assoc. Comput. Mach., 15 (1972), pp. 883-890.

[10] K. H. HASKELL AND R. J. HANSON, *An algorithm for linear least squares problems with equality and nonnegativity constraints*, Math. Programming, 21 (1981), pp. 98-118.

[11] C. L. LAWSON AND R. J. HANSON, *Solving Least Squares Problems*, Prentice-Hall, Englewood Cliffs, NJ, 1974.

[12] D. L. PHILLIPS, *A technique for the numerical solution of certain integral equations of the first kind*, J. Assoc. Comput. Mach., 9 (1962), pp. 84-97.

[13] C. RADHAKRISHNA RAO, *Linear Statistical Inference and Its Applications*, John Wiley, New York, 1973.

[14] J. REPLOGLE, B. D. HOLCOMB AND W. R. BURRUS, *The use of mathematical programming for solving singular and poorly conditioned systems of equations*, J. Mathematical Analysis and Applics., 20 (1967), pp. 310-324.

[15] B. W. RUST AND W. R. BURRUS, *Mathematical Programming and the Numerical Solution of Linear Equations*, American Elsevier, New York, 1972.

[16] O. STRAND AND E. R. WESTWATER, *Statistical estimation of the numerical solution of a Fredholm integral equation of the first kind*, J. Assoc. Comput. Mach., 15 (1968), pp. 100-114.

[17] A. N. TIKHONOV AND V. Y. ARSENIN, *Solutions of Ill-Posed Problems*, V. H. Winston, Washington DC, 1977.

[18] S. TWOMEY, *On the numerical solution of Fredholm integral equations of the first kind by the inversion of the linear systems produced by quadrature*, J. Assoc. Comput. Mach., 10 (1963), pp. 97-101.

[19] R. J. TRUMPLER AND H. F. WEAVER, *Statistical Astronomy*, Dover, New York 1962.

[20] J. M. VARAH, *A practical examination of some numerical methods for linear discrete ill-posed problems*, SIAM Rev., 21 (1979), pp. 100-111.

[21] V. V. VERBINSKI, W. R. BURRUS, T. A. LOVE, W. ZOBEL, N. W. HILL AND R. TEXTOR, *Calibration of an organic scintillator for neutron spectrometry*, Nuclear Instruments and Methods, 65 (1968), pp. 8-25.

SIAM J. SCI. STAT. COMPUT.
Vol. 7, No. 2, April 1986

010

IS SOR COLOR-BLIND?*

LOYCE M. ADAMS† AND HARRY F. JORDAN‡

Abstract. The work of Young in 1950, see Young [1950], [1971], showed that the Red/Black ordering and the natural rowwise ordering of matrices with Property A, such as those arising from the 5-point discretization of Poisson's equation, lead to SOR iteration matrices with identical eigenvalues. With the advent of parallel computers, multicolor point SOR schemes have been proposed for more complicated stencils on 2-dimensional rectangular grids, see Adams and Ortega [1982] for example, but to our knowledge, no theory has been provided for the rate of convergence of these methods relative to that of the natural rowwise scheme.

New results show that certain matrices may be reordered so the resulting multicolor SOR matrix has the same eigenvalues as that for the original ordering. In addition, for a wide range of stencils, we show how to choose multicolor orderings so the multicolor SOR matrices have the same eigenvalues as the natural rowwise SOR matrix. The strategy for obtaining these orderings is based on "data flow" concepts and can be used to reach Young's conclusions above for the 5-point stencil.

The importance of these results is threefold. Firstly, a constructive and easy means of finding these multicolorings is a direct consequence of the theory; secondly, multicolor SOR methods can be found that have the same rate of convergence as the natural rowwise SOR method for a wide range of stencils used to discretize partial differential equations; and thirdly, these multicolor SOR methods can be efficiently implemented on a wide class of parallel computers.

Keywords. multi-color ordering, SOR, data flow, parallel processing
AMS (MOS) subject classifications. 65, 68

1. Introduction. The successive overrelaxation (SOR) iterative method can be used to solve a linear system of equations,

$$A\mathbf{u} = \mathbf{b} \tag{1}$$

and is guaranteed to converge if the matrix A is symmetric and positive definite and the relaxation factor, ω, is in the interval $0 < \omega < 2$. If we express A as,

$$A = D - L - U \tag{2}$$

where D, L, and U are the diagonal, strictly lower and upper triangular parts of A respectively, the SOR iteration matrix, $\mathcal{L}_\omega$, is given by (3).

$$\mathcal{L}_\omega = (D - \omega L)^{-1}(\omega U + (1-\omega)D). \tag{3}$$

If we reorder the equations in (1) to get the system $\hat{A}\hat{\mathbf{u}} = \hat{\mathbf{b}}$, the resulting SOR matrix $\hat{\mathcal{L}}_\omega$ is not guaranteed to have the same eigenvalues as $\mathcal{L}_\omega$ of (3) and hence the convergence rates of the two SOR schemes may be different.

The matrix A frequently arises from the discretization of an elliptic partial differential equation on a rectangular region by a local stencil and by numbering the grid nodes in the natural rowwise fashion (left to right, bottom to top). For example, for Poisson's equation on a rectangle, we can use the 5-point stencil and number the grid as shown in Fig. 1. Young [1950], [1971] showed that the Red/Black ordering of the nodes as shown in Fig. 2 and the natural rowwise ordering of Fig. 1 led to SOR iteration matrices with the same eigenvalues. Knowing that these SOR matrices have the same eigenvalues is important; since for parallel computers we can choose the

* Received by the editors March 18, 1984, and in revised form November 21, 1985.

† Institute for Computer Applications in Science and Engineering, Hampton, Virginia, and Department of Mathematics, University of California, Los Angeles, California 90024. This research paper was supported by the National Aeronautics and Space Administration under contracts NAS1-17070 and NAS1-17130 while the author was in residence at ICASE, NASA Langley Research Center, Hampton, Virginia 23665.

‡ University of Colorado, Boulder, Colorado 80309.

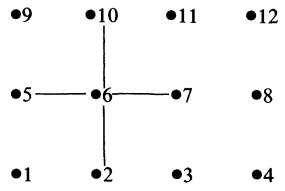


FIG. 1. *5-point stencil and natural rowwise ordering.*

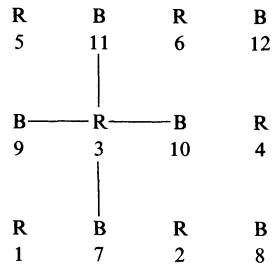


FIG 2. *5-point stencil and red/black ordering.*

Red/Black ordering instead of the natural rowwise ordering without a degradation in the asymptotic convergence rate. More parallelism is achieved with this ordering since nodes of the same color are not neighbors, which implies all Red and then all Black nodes may be updated simultaneously.

With the advent of parallel computers, multicolor point SOR schemes have been proposed for more complicated stencils on 2-dimensional rectangular grids, see Adams and Ortega [1982] for example, but to our knowledge no theory has been provided for the rate of convergence of these methods relative to that of the natural rowwise scheme. This paper will show that the geometry of the stencil can be used to derive a coloring for the region such that the multicolor and natural rowwise SOR iteration matrices have the same eigenvalues. The fact that this is true for a class of stencils containing those of interest in partial differential equations means that highly parallel iterative methods can be formulated which converge equally as well as their sequential counterparts.

In § 2, we describe how trying to implement natural rowwise SOR for a 9-point stencil on the Denelcor HEP, Patel and Jordan [1984], led to a strategy based on data flow ideas for finding orderings that are highly parallel and that produce the same iterates as the natural rowwise iteration. The ideas in § 2 are formalized in § 3 where we prove that the SOR iteration with the orderings generated by the data flow strategy has the same asymptotic rate of convergence as the SOR iteration with particular multicolor orderings. In § 4 we show how to find these multicolor orderings for a wide range of stencils. In § 5, we describe some interesting implementation issues for multicolor SOR on various parallel computers. Finally, in conclusion, we summarize our results, mention generalizations of our ideas to block SOR and multiple equations per grid point, and list unanswered questions.

2. Parallelizing the natural rowwise SOR algorithm. In the multiple instruction stream, or MIMD, environment such as that of the Denelcor HEP, it is useful to investigate speeding up the sequential rowwise SOR computation by using multiple instruction streams called processes. If an arbitrary number of processes are available in the computing environment, we want to investigate a MIMD algorithm that implements a parallel SOR iteration that is equivalent to the natural rowwise SOR algorithm.

The important ideas that this approach yields can best be seen by a specific example. We consider the 9-point stencil shown in Fig. 3. The natural rowwise ordering of a rectangular grid imposes the following update rule, or data flow dependencies, for the unknown at the center of the stencil:

NR *stencil rule.* The value at the center of the stencil for iteration $k+1$ can be calculated after the values to the left of and below center (backward neighbors) have been calculated on iteration $k+1$ and the values to the right and above center (forward neighbors) have been calculated on iteration k.

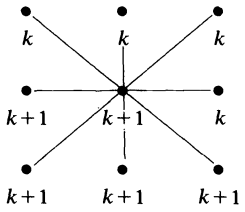


FIG. 3. *9-point stencil.*

This rule is depicted in Fig. 3 for the 9-point stencil.

To formulate the NR stencil rule in the language of data flow (Ackerman [1982]), we look at the set of $n \times m \times K$ updates implied by the rule for unknowns $u^{(k)}(i,j)$ at grid point (i,j) on iteration k, $k=1,\cdots,K$, $i=1,\cdots,n$, and $j=1,\cdots,m$. We will use (i,j) to indicate the point in the ith row and jth column of the grid. These define a computation in a value oriented, or single-assignment, computational model with initial values $u^{(0)}$ and boundary values considered as constants. In this model all values are viewed as having an independent existence with no concept of updating a storage location, so that $n \times m \times K$ storage locations would be needed to store the $n \times m$ unknowns on the K iterations. It is clear that a sequential rowwise sweep for iteration 1 followed by one for iteration 2, and so on, is not the only scheduling of equation evaluations consistent with rule NR. For example, with the 9-point stencil, as soon as $u^{(1)}(1,1)$ and $u^{(1)}(1,2)$ have been evaluated, the computations for $u^{(1)}(1,3)$ and $u^{(1)}(2,1)$ can proceed simultaneously and when $u^{(1)}(1,2)$, $u^{(1)}(2,1)$ and $u^{(1)}(2,2)$ have been computed the iteration 2 value $u^{(2)}(1,1)$ can be produced. If we assume an arbitrarily large number of processors and schedule each computation as early as possible, then many computations, possibly associated with different iterations, will occur simultaneously at points spanning the region.

In general, to cast this $n \times m \times K$ scheduling problem into the form of an iteration on an $n \times m$ region, with only $n \times m$ storage locations required, it is convenient to require symmetry of the stencil as explained below. For a structurally symmetric stencil, like our example in Fig. 3, a point (i,j) is a forward neighbor of all its backward neighbors and a backward neighbor of all its forward neighbors. Thus, by the time it is possible to compute $u^{(k+1)}(i,j)$, the value $u^{(k)}(i,j)$ has been used to compute $u^{(k+1)}$ for all backward neighbors and $u^{(k)}$ for all forward neighbors and these are the only computations which require it. Thus only the "current" iteration value is needed at a specific point (i,j) and the re-use of an $n \times m$ storage array is possible for all the schedulings satisfying the NR rule. The "current" iteration number may be different for different nodes of the region as a function of the specific schedule. We will assume a structurally symmetric stencil henceforth and use rule NR to refer to the $(k+1)$st update of the point at the center of the stencil.

For the 9-point stencil and for many others of interest in PDEs this data flow scheme of scheduling updates as early as allowed by rule NR will be shown to lead to multicolor iterative methods. If each computation takes one time unit then a subset R of grid points will be associated with computations scheduled at time t. At $t+1$ a new subset B of nodes will be computed. If these successively scheduled subsets are disjoint, exhaust the region, and the time difference between subsequent iterations is the same constant for all nodes, then we will show that the resulting time schedule has the form of a multicolor iteration.

To continue our example, we consider the 6 by 5 grid of interior nodes that results from discretizing an 8 by 7 grid with the 9-point stencil with boundary values assumed

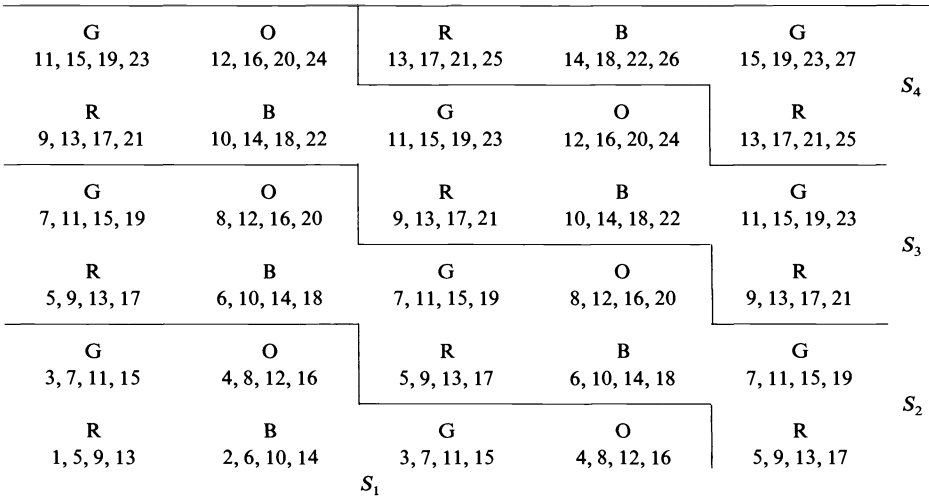


FIG. 4. *Earliest times for the 9-point stencil.*

to be known. This grid is shown in Fig. 4 where the numbers below each node indicate the earliest times that consecutive iterations can begin at the node according to the NR stencil rule if the computation at a node requires one time unit. Notice that each node updates every $\Delta t = 4$ time units. The solid lines in Fig. 4 separate the nodes into four disjoint sets, denoted S_1, S_2, S_3, and S_4 with set S_1 containing nodes that update for the first time during times 1 to Δt, S_2 containing nodes with first update times in the interval $\Delta t + 1$ to $2\Delta t$, etc. These sets will be formally defined later. The following statements can be made from Fig. 4 by observing these earliest update times. Note that at times (1, 2, 3, 4) iteration 1 can be done on the nodes in S_1, at times (5, 6, 7, 8) iteration 1 can be done on the nodes in S_2 and iteration 2 on the nodes in S_1; at times (9, 10, 11, 12) iteration 1 on set S_3, iteration 2 on S_2 and iteration 3 on S_1 can be done; at times (13, 14, 15, 16) iteration 1 on S_4, iteration 2 on S_3, iteration 3 on S_2, and iteration 4 on S_1 can be done; at times (17, 18, 19, 20) iteration 2 on S_4, iteration 3 on S_3 and iteration 4 on S_2 can be done; at times (21, 22, 23, 24) iteration 3 on S_4, iteration 4 on S_3 can be done; and finally at times (25, 26, 27) iteration 4 can be done on S_4.

At this point, it is instructive to remark on the difference between a *data flow scheme* of assigning times to the nodes and an *ordering* of the nodes as reflected by the order of the equations of A in (1). For the 30 nodes (30 equations) in Fig. 4, the scheme of assigning update times at a node to be the earliest times for which the data for the node was available according to the NR Stencil Rule resulted in 15 unique times being assigned to the nodes on the first iteration; whereas, an ordering must assign the integers 1 to 30 to the nodes. However, there are orderings that are consistent with the NR Stencil Rule that can be constructed by considering the update times for the first iteration that resulted from the data flow scheme. We define these orderings in Definition 1.

DEFINITION 1. An NR *data flow ordering* of the nodes of a grid is an ordering by the first earliest update times according to the NR rule with ties being resolved arbitrarily.

There are many orderings of the 30 equations of A in (1) that do not violate the NR stencil rule, but do not have the same update times for the first iteration as shown in Fig. 4. One such ordering is the natural rowwise, left to right, bottom to top ordering. All orderings that do not violate the NR Stencil Rule are nonmigratory permutations

of one another, as described in Young [1971], and will result in SOR matrices with equal eigenvalues. However, it is the NR data flow orderings that are useful in proving our theorems about particular multicolor SOR iterations.

A crucial observation is that at times (13, 14, 15, 16), nodes from all four sets are being updated and that all nodes which can be updated at a particular time, say 13, are not neighbors. Hence, if we color all nodes Red that are updated at time 13, all nodes Black that are updated at time 14, all nodes Green that are updated at time 15, and all nodes Orange that are updated at time 16 as shown in Fig. 4, it is clear that the schedule of updates for four iterations of SOR consistent with the NR Stencil Rule *contains* one iteration (times 13, 14, 15, 16) of a multicolor SOR iteration (R/B/G/O-SOR) as defined by Adams and Ortega [1982]. If we order the colors R/B/G/O as 1/2/3/4, a closer examination of Fig. 4 also reveals that nodes of color i in S_i are connected only to nodes of colors greater than i in S_{i-1} and only to nodes of colors less than i in S_{i+1}. Also nodes in S_i are not connected to any nodes in sets greater than $i+1$ or less than $i-1$.

A strategy for proving that the SOR iteration matrix that results from ordering the Red equations first, followed by the Black, Green, and then Orange equations will have the same eigenvalues as the natural rowwise SOR iteration matrix has evolved from this example. We now formalize this strategy.

3. Theory. The data flow example of the last section leads us to begin with the following definitions.

DEFINITION 2. A *multicolor*, or c-color, matrix is a $c \times c$ block matrix of the form

$$M = \begin{bmatrix} D_1 & X_{12} & \cdot & \cdot & X_{1c} \\ X_{21} & D_2 & \cdot & \cdot & X_{2c} \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ X_{c1} & \cdot & \cdot & \cdot & D_c \end{bmatrix}$$

where the D_i are diagonal matrices, each associated with a different color, and the X_{ij} are arbitrary.

DEFINITION 3. A *multicolor T matrix* is a block tridiagonal matrix of the form,

$$T_M = \begin{bmatrix} M_1 & L_1 & & & \\ U_1 & M_2 & L_2 & & \\ & U_2 & \cdot & & \\ & & \cdot & M_{s-1} & L_{s-1} \\ & & & U_{s-1} & M_s \end{bmatrix}$$

where M_i, $2 \leqq i \leqq s-1$, are multicolor matrices with c colors numbered $1, \cdots, c$ respectively; M_1 is a multicolor matrix with $c-f+1$ colors numbered $f, \cdots, c$ respectively; M_s is a multicolor matrix with $e < c$ colors numbered $1, \cdots, e$ respectively. In addition, the matrices L_i, $2 \leqq i \leqq s-2$ are strictly lower $c \times c$ block triangular matrices; the matrix L_1 is a $(c-f+1) \times (c)$ block matrix with blocks $B_{ij} = 0$ if $j \geqq i+f-1$; the matrix L_{s-1} is a $c \times e$ block matrix with blocks $B_{ij} = 0$ if $j \geqq i$; and the matrices U_i, $i = 1, \cdots, s-1$ have the transposed form of the matrices L_i, $i = 1, \cdots, s-1$ respectively.

It is easy to show that the matrix that results from permuting the rows and columns of T_M to bring nodes of color j from all M_i together into the same block D_j is a multicolor matrix with c colors ordered $1, \cdots, c$ respectively. Such a matrix is called the *multicolor matrix associated with* T_M.

In relation to Fig. 4 with R/B/G/O representing colors 1/2/3/4, the blocks M_i, $i=1,\cdots,4$ correspond to the four sets S_i, $i=1,\cdots,4$, with the nodes in a given block ordered by colors, the value of $c=4$, the value of $f=1$ and the value of $e=3$. The matrix T_M has the following form where D_{ij} indicates the nodes of color i from block M_j.

$$T_M = \begin{bmatrix}
\begin{bmatrix} D_{11} & \times & \times & \times \\ \times & D_{21} & \times & \times \\ \times & \times & D_{31} & \times \\ \times & \times & \times & D_{41} \end{bmatrix} \begin{bmatrix} 0 & & & \\ \times & 0 & & \\ \times & \times & 0 & \\ \times & \times & \times & 0 \end{bmatrix} & & & \\
\begin{bmatrix} 0 & \times & \times & \times \\ & 0 & \times & \times \\ & & 0 & \times \\ & & & 0 \end{bmatrix} \begin{bmatrix} D_{12} & \times & \times & \times \\ \times & D_{22} & \times & \times \\ \times & \times & D_{32} & \times \\ \times & \times & \times & D_{42} \end{bmatrix} \begin{bmatrix} 0 & & & \\ \times & 0 & & \\ \times & \times & 0 & \\ \times & \times & \times & 0 \end{bmatrix} & & \\
\begin{bmatrix} 0 & \times & \times & \times \\ & 0 & \times & \times \\ & & 0 & \times \\ & & & 0 \end{bmatrix} \begin{bmatrix} D_{13} & \times & \times & \times \\ \times & D_{23} & \times & \times \\ \times & \times & D_{33} & \times \\ \times & \times & \times & D_{43} \end{bmatrix} \begin{bmatrix} 0 & & & \\ \times & 0 & & \\ \times & \times & 0 & \\ \times & \times & \times & 0 \end{bmatrix} & \\
\begin{bmatrix} 0 & \times & \times & \times \\ & 0 & \times & \times \\ & & 0 & \times \\ & & & 0 \end{bmatrix} \begin{bmatrix} D_{14} & \times & \times \\ \times & D_{24} & \times \\ \times & \times & D_{34} \end{bmatrix}
\end{bmatrix}.$$

The associated 4-color matrix is

$$M = \begin{bmatrix}
\begin{bmatrix} D_{11} & & & \\ & D_{12} & & \\ & & D_{13} & \\ & & & D_{14} \end{bmatrix} & \begin{bmatrix} \times & \times & \times & \times \\ \times & \times & \times & \times \\ \times & \times & \times & \times \\ \times & \times & \times & \times \end{bmatrix} & \begin{bmatrix} \times & \times & \times & \times \\ \times & \times & \times & \times \\ \times & \times & \times & \times \\ \times & \times & \times & \times \end{bmatrix} & \begin{bmatrix} \times & \times & \times & \times \\ \times & \times & \times & \times \\ \times & \times & \times & \times \\ \times & \times & \times & \times \end{bmatrix} \\
\begin{bmatrix} \times & \times & \times & \times \\ \times & \times & \times & \times \\ \times & \times & \times & \times \\ \times & \times & \times & \times \end{bmatrix} & \begin{bmatrix} D_{21} & & & \\ & D_{22} & & \\ & & D_{23} & \\ & & & D_{24} \end{bmatrix} & \begin{bmatrix} \times & \times & \times & \times \\ \times & \times & \times & \times \\ \times & \times & \times & \times \\ \times & \times & \times & \times \end{bmatrix} & \begin{bmatrix} \times & \times & \times & \times \\ \times & \times & \times & \times \\ \times & \times & \times & \times \\ \times & \times & \times & \times \end{bmatrix} \\
\begin{bmatrix} \times & \times & \times & \times \\ \times & \times & \times & \times \\ \times & \times & \times & \times \\ \times & \times & \times & \times \end{bmatrix} & \begin{bmatrix} \times & \times & \times & \times \\ \times & \times & \times & \times \\ \times & \times & \times & \times \\ \times & \times & \times & \times \end{bmatrix} & \begin{bmatrix} D_{31} & & & \\ & D_{32} & & \\ & & D_{33} & \\ & & & D_{34} \end{bmatrix} & \begin{bmatrix} \times & \times & \times & \times \\ \times & \times & \times & \times \\ \times & \times & \times & \times \\ \times & \times & \times & \times \end{bmatrix} \\
\begin{bmatrix} \times & \times & \times & \times \\ \times & \times & \times & \times \\ \times & \times & \times & \times \\ \times & \times & \times & \times \end{bmatrix} & \begin{bmatrix} \times & \times & \times & \times \\ \times & \times & \times & \times \\ \times & \times & \times & \times \\ \times & \times & \times & \times \end{bmatrix} & \begin{bmatrix} \times & \times & \times & \times \\ \times & \times & \times & \times \\ \times & \times & \times & \times \\ \times & \times & \times & \times \end{bmatrix} & \begin{bmatrix} D_{41} & & & \\ & D_{42} & & \\ & & D_{43} & \end{bmatrix}
\end{bmatrix}.$$

We now prove our major theorem.

THEOREM 1. *A multicolor* T *matrix and its associated multicolor matrix have* SOR *iteration matrices with the same eigenvalues.*

Proof. Let $\mathscr{L}_{T,\omega}$ be the SOR matrix for the multicolor T matrix; $\mathscr{L}_{c\omega}$ be the SOR matrix for the multicolor matrix and P be the permutation matrix that transforms the multicolor T ordering to the multicolor ordering.

Definition 3 guarantees that a multicolor T matrix is also a banded matrix with semibandwidth equal to $c-1$ with the blocks D_{ij} on the diagonal being diagonal blocks. This means that consecutive iterations of SOR can occur for every node every c time units. Let $Q_i^{(k)}$ represent the set of earliest times, t, that the nodes in blocks D_j in block M_i can be updated on iteration k. Specifically,

$$
\begin{aligned}
&Q_1^{(k)} = \{t \mid (k-1)c+1 \leqq t \leqq kc-f+1\},\\
&Q_i^{(k)} = \{t \mid (k+i-2)c-f+2 \leqq t \leqq (k+i-1)c-f+1\},\ 2 \leqq i \leqq s-1,\\
&Q_s^{(k)} = \{t \mid (k+s-2)c-f+2 \leqq t \leqq (k+s-2)c-f+1+e\},
\end{aligned} \tag{4}
$$

and it follows that iterations $s-1, s-2, \cdots, 1$ can be done on the nodes in blocks M_i, $i=1,2,\cdots,s-1$ respectively before iteration 1 is started on the nodes in block M_s. Also, we can conclude from (4) that at the c times $(s+i-2)c-f+2$ to $(s+i-1)c-f+1$ inclusively, a multicolor $\mathcal{L}_{c,\omega}$ iteration with colors numbered $1,\cdots,c$ can be done. This suggests that we consider the matrix $\mathcal{L}_{T,\omega}$ in the factored form,

$$\mathcal{L}_{T,\omega} = \mathcal{L}_s \mathcal{L}_{s-1} \cdots \mathcal{L}_2 \mathcal{L}_1 \tag{5}$$

where the $N \times N$ matrix $\mathcal{L}_i$ represents one SOR iteration on the nodes in block M_i of the multicolor T matrix. In particular, let

(1) n_i be the number of nodes in block M_i,

(2) $I_{(ij)}$ be the $N \times N$ matrix with a 1 in the diagonal position of the row associated with the jth node of block M_i and zeros elsewhere;

(3) $B_{(ij)}$ be the $N \times N$ matrix with the row associated with the jth node of block M_i being equal to the row of the Jacobi iteration matrix, $B = D^{-1}(L+U)$, associated with this node and all other rows being zero.

Then $\mathcal{L}_i$ can be written as

$$\mathcal{L}_i = \prod_{j=1}^{n_i} (\omega B_{(ij)} + I - \omega I_{(ij)}) \tag{6}$$

and has the form

$$\mathcal{L}_i = \begin{bmatrix} I_1 & & & & & & \\ & I_2 & & & & & \\ & & I_{i-1} & & & & \\ & & X & X & X & & \\ & & & & & I_{i+1} & \\ & & & & & & I_s \end{bmatrix}. \tag{7}$$

Now, $k+s-1$ iterations of multicolor T SOR can be expressed in terms of k iterations of multicolor SOR as

$$\mathcal{L}_{T,\omega}^{k+s-1} = R P^T \mathcal{L}_{c,\omega}^k P S \tag{8}$$

where

$$
\begin{aligned}
R &= (\mathcal{L}_s \cdots \mathcal{L}_3 \mathcal{L}_2) \cdots (\mathcal{L}_s \mathcal{L}_{s-1} \mathcal{L}_{s-2})(\mathcal{L}_s \mathcal{L}_{s-1})(\mathcal{L}_s),\\
S &= (\mathcal{L}_1) \cdots (\mathcal{L}_{s-2} \cdots \mathcal{L}_2 \mathcal{L}_1)(\mathcal{L}_{s-1} \cdots \mathcal{L}_2 \mathcal{L}_1).
\end{aligned} \tag{9}
$$

Since $\det(\mathcal{L}_{T,\omega}) = (1-\omega)^N$ where N is the size of $\mathcal{L}_{T,\omega}$, it follows that $\mathcal{L}_{T,\omega}$ is nonsingular if $\omega \neq 1$. Hence the factors in (5) and R in (8) are nonsingular whenever $\omega \neq 1$.

So, for $\omega \neq 1$, it follows from (8) that

$$\mathcal{L}_{T,\omega}^{k+s-1} = RP^T \mathcal{L}_{c,\omega}^k PR^{-1} RS. \tag{10}$$

Since the multicolor T matrix is block tridiagonal, the matrix $\mathcal{L}_i$ commutes with $\mathcal{L}_j$ if $j \neq i-1, i+1$ as is easily seen by (7). By applying this fact to the product RS in (10), we get

$$RS = \mathcal{L}_{T,\omega}^{s-1}. \tag{11}$$

Hence, for $\omega \neq 1$,

$$\mathcal{L}_{T,\omega}^k = RP^{-1} \mathcal{L}_{c,\omega}^k PR^{-1} \tag{12}$$

and it follows that $\mathcal{L}_{T,\omega}$ and $\mathcal{L}_{c,\omega}$ have the same eigenvalues.

Finally, the case $\omega = 1$ follows because the eigenvalues of a matrix are continuous functions of the coefficients of the matrix, and the theorem is proved.

Now, consider the five sets that would be formed in Fig. 4 by letting $f = 4$. This results in a grouping of nodes with the earliest first update times of 1, 2–5, 6–9, 10–13, and 14–15 into sets 1 to 5 respectively. This corresponds to assigning the numbers 1/2/3/4 to the colors B/G/O/R respectively with the other values in Definition 2 being $c = 4$ and $e = 2$. The resulting T_M matrix is the same as that for the assignment of 1/2/3/4 to the colors R/B/G/O in the previous example; however, the block structure is different, indicating different associated multicolor matrices. Also, we could let $f = 3$ and then $f = 2$ and effectively describe G/O/R/B and O/R/B/G orderings of the equations. This discussion leads to the following corollary.

COROLLARY 1. *If the multicolor T matrix results from a* NR *data flow ordering of the grid points, the c multicolor* SOR *matrices that arise by letting f in Definition* 2 *vary from* 1 *to c have the same eigenvalues as the natural rowwise* SOR *matrix.*

Proof. From Theorem 1 we conclude that the c multicolor SOR matrices have the same eigenvalues as the multicolor T SOR matrix. However, the multicolor T ordering is the NR data flow ordering which is just a nonmigratory permutation of the natural rowwise ordering and the corollary follows.

For our example, the R/B/G/O, B/G/O/R, G/O/R/B, and the O/R/B/G SOR matrices have the same eigenvalues as the natural rowwise SOR matrix, and thus an iteration done with any of these multicolor matrices will converge at the same asymptotic rate as the iteration using the sequential rowwise matrix.

The question arises whether there are other four color orderings for the 9-point stencil of Fig. 3 that lead to multicolor and natural rowwise SOR matrices that have the same eigenvalues. We provide a partial answer to this question with Corollary 2.

COROLLARY 2. *The multicolor* SOR *matrix associated with the matrix T_M in Definition* 3 *and the multicolor* SOR *matrix that results from ordering the equations of T_M in reverse order have the same eigenvalues whenever T_M is symmetric.*

Proof. Let A_1, $\mathcal{L}_{1,\omega}$ and A_2, $\mathcal{L}_{2,\omega}$ be the respective multicolor T and multicolor T SOR matrices for the forward and reverse orders respectively. Then by Theorem 1, the multicolor SOR matrices have the same eigenvalues as $\mathcal{L}_{1,\omega}$ and $\mathcal{L}_{2,\omega}$ respectively. It remains to show that $\mathcal{L}_{1,\omega}$ and $\mathcal{L}_{2,\omega}$ have the same eigenvalues.

Now, let

$$A_1 = D_1 - L_1 - U_1$$

and

$$A_2 = D_1 - L_2 - U_2$$

where D_i, L_i, and U_i are defined by (2).

If P is the permutation matrix from the forward to the reverse ordering, we have,

$$P^T D_1 P = D_2; \quad P^T L_1 P = U_2, \quad P^T U_1 P = L_2$$

and

$$P^T \mathcal{L}_{1,\omega} P = (D_2 - \omega U_2)^{-1}(\omega L_2 + (1-\omega) D_2).$$

But since A_2 is symmetric, $U_2 = L_2^T$ and

$$P^T \mathcal{L}_{1,\omega} P = [\omega U_2 + (1-\omega) D_2)(D_2 - \omega L_2)^{-1}]^T.$$

Since a square matrix has the same eigenvalues as its transpose, and the eigenvalues of a product AB equals the eigenvalues of BA, it follows from (3) that $\mathcal{L}_{1,\omega}$ and $\mathcal{L}_{2,\omega}$ have the same eigenvalues and the corollary follows.

Therefore, for Fig. 4, the O/G/B/R, R/O/G/B, B/R/O/G, G/B/R/O, and the natural rowwise orderings have SOR matrices with the same eigenvalues whenever the matrix A of (1) is symmetric.

Obviously, there are other 4-color topologies of a rectangular grid that is discretized with the 9-point stencil of Fig. 3. For example, consider the coloring shown in Fig. 5.

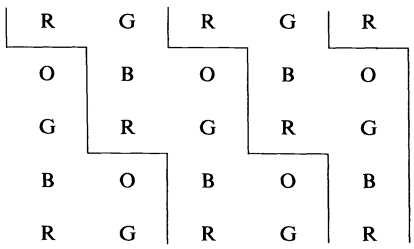


FIG. 5. *Columnwise coloring for the 9-point stencil.*

With the sets indicated in Fig. 5, the R/B/G/O, B/G/O/R, G/O/R/B, O/R/B/G orderings lead to SOR matrices with the same eigenvalues as the natural columnwise (left to right, bottom to top) SOR matrix. Another example, that was given in Adams [1982] is shown in Fig. 6 along with the associated earliest update times for the first two iterations. These update times were obtained by considering two different update rules for the grid. The first rule applies to the R and G points and is identical to the rule in Fig. 3 except the data used from the west neighbor is from iteration k instead of $k+1$. The second rule applies to the O and B points and is like Fig. 3 except the data used from the east neighbor is from iteration $k+1$ instead of iteration k.

The sets indicated above show that $f = 1$, and $e = 4$. Also, the value of f can not be changed and still maintain a multicolor T matrix since all colors must be present in each set for the earliest times shown above. Theorem 1 can be applied to Fig. 6 to prove that R/B/G/O SOR has the same eigenvalues as a multicolor T SOR matrix that is natural rowwise-like in the sense that we update nodes in set 1 before those in set 2, etc., but within sets the ordering is R/B/G/O and is not a nonmigratory permutation of the natural rowwise ordering.

So far, we have shown that the 9-point stencil has multicolor orderings with SOR matrices having eigenvalues identical to those of the SOR matrix for the natural rowwise ordering, the natural columnwise ordering, and a rowwise-like ordering. Next, we turn to the practical questions.

G 11, 15	O 12, 16	G 11, 15	O 12, 16
R 9, 13	B 10, 14	R 9, 13	B 10, 14
G 7, 11	O 8, 12	G 7, 11	O 8, 12
R 5, 9	B 6, 10	R 5, 9	B 6, 10
G 3, 7	O 4, 8	G 3, 7	O 4, 8
R 1, 5	B 2, 6	R 1, 5	B 2, 6

FIG. 6. *Rowwise-like coloring for the 9-point stencil.*

(1) What stencils have multicolor and NR SOR matrices with the same eigenvalues?

(2) How do we find these multicolor orderings?

4. Special stencils and NR-equivalent multicolor orderings. In this section, we define a class of stencils for which a discretized rectangular domain has multicolor orderings with SOR matrices that have the same eigenvalues as the natural rowwise (NR) SOR matrix. In addition, we show how to find these colorings and illustrate the procedure for several well-known stencils. We begin with the following definitions.

DEFINITION 4. A stencil S is a pattern of neighbors for a given node $(\bar{i}, \bar{j})$. The stencil has $(0, 0)$ as its center node and is defined relative to $(\bar{i}, \bar{j})$ as follows: (p, q) is said to be a node in stencil S if for all (i, j) in the region, the node $(i+p, j+q)$ is a neighbor of (i, j) provided it is also in the region.

DEFINITION 5. A stencil that is structurally symmetric about node $(0, 0)$ is a SO-stencil. That is, if node (i, j) is in the stencil, then node $(-i, -j)$ is also in the stencil.

We note that a symmetric stencil leads to a symmetric nonzero pattern in the matrix A of (1), but does not necessarily lead to numerical symmetry of A.

Now, recall from the 9-point stencil example of the last section that every node could be updated every c time units, where c was the number of colors. This fact was reflected in the definition of a multicolor T matrix and was necessary to prove that the R/B/G/O, B/G/O/R, G/O/R/B, and O/R/B/G SOR matrices for Fig. 4 had the same eigenvalues as the NR SOR matrix. However, update rules based on other orderings, as shown in Figs. 5 and 6, also lead to constant update intervals but may not be equivalent to the NR ordering. We now seek a class of stencils for which a constant updating increment is both necessary and sufficient to find multicolor orderings that are equivalent to the NR scheme.

The first step in this direction is to consider stencils for which consecutive nodes in a given row of the grid update one time unit apart. This is ensured by requiring that a SO stencil contain node $(0, 1)$, and likewise, node $(0, -1)$. Such a stencil is called a $(0, 1)$-SO stencil, and the result is proven in Theorem 2.

THEOREM 2. *If a rectangular grid is discretized with a* $(0, 1)$-SO *stencil and an iteration is done at the earliest time according to rule* NR *and requires one time unit to*

complete, then iteration k for node $(i, j+1)$ *begins one time unit later than iteration k for node* (i, j).

Proof. The proof is by induction on k. For each value of k we induct on i and for each value of i we induct on j. The proof is given in its entirety in the Appendix.

We remark that we have been able to find multicolor orderings for SO stencils with the properties that nodes update every c units of time and the SOR matrices have the same eigenvalues as the NR scheme, but do not contain node (0, 1). However, this requirement that a node have an "east" and "west" neighbor is not restrictive for stencils that are commonly used for PDEs as will be illustrated later.

The next step is to further restrict the stencil so that a node updates every c time units. The NR rule says that the time a node can update on iteration $k+1$ is a function of the times its backward neighbors were updated on iteration $k+1$ and the times its forward neighbors were updated on iteration k. However, the times the nodes update on the first iteration are determined by the backward neighbor times only. Since stencils of interest contain a node in row -1, it is convenient to consider stencils for which a node in row -1 is the last backward node to be updated. This will be true for node $(-1, \alpha)$ if nodes strictly below the x-axis and strictly above line L_1 in Fig. 7 are excluded from the stencil as indicated by the darkened nodes in Fig. 7. Node $(-1, \alpha)$ is called the "controlling" backward neighbor. Similarly, we require the last forward neighbor to be updated to be above the x-axis in row γ and column β, $\beta \geqq 0$. This will be true for node (γ, β) if we do not allow nodes above line L_2 to be in the stencil as shown in Fig. 7. The nodes $(-3, 0)$ and $(-3, 1)$ are excluded by symmetry of the stencil since $(3, 0)$ and $(3, -1)$ are excluded as forward neighbors. Node (γ, β) is called the "controlling" forward neighbor. Again, this is not a real restriction for practical stencils. Fig. 7 was the motivation for our major definition below.

DEFINITION 6. A SO-stencil that contains nodes $(0, 1)$, $(-1, \alpha)$, and (γ, β) with $\alpha, \beta \geqq 0$ and $\gamma > 0$ but does not contain nodes

$$\{(y, x), y \leqq -1 | x > -(\alpha+1)y - 1\}$$

or

$$\{(y, x), y \geqq 0 | x > -(\alpha+1)y + \gamma(\alpha+1) + \beta\}$$

is an (α, β, γ)-SO Stencil.

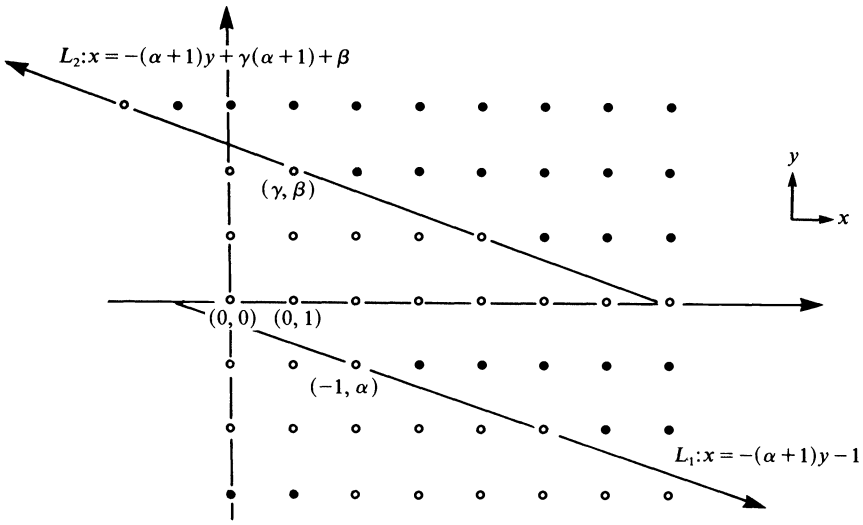


FIG. 7. *Excluded backward and forward neighbors.*

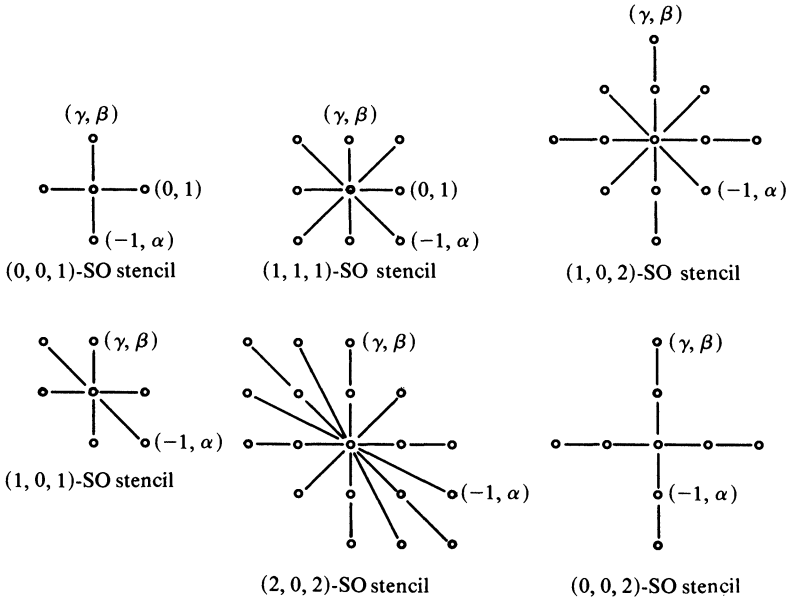


FIG. 8. *The classification of some common stencils.*

The classification of six commonly used stencils as (α, β, γ)-SO stencils is shown in Fig. 8. Hence, we see that it is quite an easy task to find the values of α, β, and γ for a given stencil that satisfy Definition 6. It is also easy to construct many stencils for given values of α, β, and γ.

In Theorem 3 below, we express the value of c in terms of α, β, and γ and describe how to color the grid so that c equals the number of colors.

THEOREM 3. *If a rectangular grid is discretized with an* (α, β, γ)-SO *stencil, the matrix A of* (1) *that results from the* NR *data flow ordering is a multicolor T matrix with*

$$c = \gamma(\alpha+1)+(\beta+1)$$

colors. Furthermore, if the first node is colored f and node (i, j) *is color*

$$[t^{(1)}(i,j)+f-2] \bmod c+1$$

then the blocks M_1, M_r, $r=2, \cdots, s-1$, *and* M_s *contain the nodes in set* S_1, S_r, $r=2, \cdots, s-1$, *and* S_s *below*:

$$S_1 = \{(i,j) | 1 \leqq t^{(1)}(i,j) \leqq c-f+1\},$$

$$S_r = \{(i,j) | (r-1)c-f+2 \leqq t^{(1)}(i,j) \leqq rc-f+1\}, \qquad 2 \leqq r \leqq s,$$

$$S_s = \{(i,j) | (s-1)c-f+2 \leqq t^{(1)}(i,j) \leqq (s-1)c-f+1+e\}.$$

Proof. It is sufficient to prove that all nodes can update every c time units, since the above specifications for M_r, $r=1 \cdots s$, and the coloring rule follow from this fact. The proof is an induction on k and for each k we induct on i and use Theorem 2 to replace the j induction. The proof is given in the Appendix.

Theorem 3 and Corollary 1 show that the c multicolor SOR matrices gotten by letting f vary from 1 to c have the same eigenvalues as the SOR matrix associated with the NR data flow ordering.

5. Implementation of parallel SOR. The data flow view of rowwise sequential SOR as a collection of computations to be scheduled as early as possible makes it clear that a parallel SOR algorithm is possible for a shared memory MIMD machine. It was the investigation of such an algorithm on the HEP computer which stimulated this paper (Patel and Jordan [1984]). The fact that stencils of interest lead to the multicolor scheme which updates fixed, disjoint and exhaustive subsets of the grid points means that implementation on an SIMD or vector computer is also simple. The degree of parallelism of the algorithm, number of processes in MIMD, or vector length in SIMD is $(n \times m)/c$ where c is the number of colors. Since the stencils of interest are limited in extent, multicolor SOR is also suitable for MIMD machines in which data communication costs are significant, such as the processor arrays discussed in Adams [1982].

In implementing multicolor SOR on a vector computer, c vectors of length $(n \times m)/c$ are updated cyclically. Since the vectors consist of subsets of region points and are fixed throughout the iteration, it is appropriate to reorder the data structure to bring all points of a given color into a single vector. If the $n \times m$ array is stored rowwise in memory, the regularity of the coloring pattern for a given stencil on the right sized grid may make it possible to form vectors with a constant stride between elements without reordering the data structure. Some vector architectures will handle constant stride vectors directly. The computation to update the vector of, say, red nodes will be a linear combination of the vectors for nodes of other colors. Two or more shifted versions of a vector may be used, since the stencil centered on a red point may include more than one, say, black neighbor. Other comments on SOR in connection with vector computers can be found in Buzbee et al. [1977] and Adams [1983].

In a shared memory MIMD implementation, the structuring of data is unimportant. Instead, the main issue is that of sychronizing processes so that a scheduling consistent with rule NR results. Potentially, the process updating point (i, j) for the kth time would have to verify that all backward neighbors have been updated k times and all forward neighbors $k-1$ times. However, for (α, β, γ)-SO stencils, it is sufficient to verify that point$(-1, \alpha)$ has been updated k times and point (γ, β) updated $k-1$ times. Synchronization at one point in the forward direction and one point in the backward direction is all that is needed. With shared memory, it is useful to think of the processes moving across the array following a wave of computation. The HEP algorithm mentioned above started with order n parallelism by letting one process sweep each row with the sweeps being as simultaneous as possible under the synchronization rules. Observing that sweeps for subsequent iterations could be started before the current one finished led to the multicolor algorithm with parallelism $(n \times m)/c$. Synchronization on the HEP was done by the producer/consumer mechanism with a computation consuming values from neighbors $(-1, \alpha)$ and (γ, β) and producing two new values, one for each of these neighbors.

On an MIMD machine where the processors are arranged in an array, it is no longer appropriate to let the processes move through the array of processors. A single processor would perform computations for a fixed subregion, preferably containing an equal number of points of each color. Synchronization becomes a by-product of the communications required to pass new iterates to other processors which require them. The storage organization places a node into the local memory of the processor responsible for updating it. This situation is thoroughly covered in Adams [1982].

6. Conclusions. The results of this paper give the practitioner in numerical partial differential equations the assurance that one can use highly parallel multicolor SOR

methods that have exactly the same convergence properties that are associated with the rowwise sequential method, and that these particular multicolor orderings are easily constructed. Thus, in this sense SOR is color-blind.

There are, however, colorings, like those in Figs. 5 and 6, which decouple the stencil on the region which have SOR matrices with the same eigenvalues as columnwise and other, less regular, orderings. The relationship of SOR with these orderings to the natural rowwise SOR is still an open question in general. However, for the 5-point stencil of Fig. 8 the Red/Black, Black/Red, natural rowwise, and natural columnwise orderings all have SOR iteration matrices with equal eigenvalues which corroborates Young's results that were obtained by the use of consistently ordered matrices. In addition, for symmetric matrices, our results show that for the 6-point (1, 0, 1)-SO stencil of Fig. 8 all 6 orderings of the unique 3-color pattern which decouples the stencil lead to NR equivalent SOR matrices. This means that columnwise and rowwise SOR have the same asymptotic convergence rate for this stencil. For the 9-point (1, 1, 1)-SO of Fig. 8, for symmetric matrices, we only exhibited 8 orderings using 4 colors that were equivalent to the rowwise natural ordering. These 8 are a small fraction of the evaluation orderings on several 4-color topologies which decouple the stencil on the grid.

The multicolor T matrix defined here does not appear to lead to the determination of an optimal relaxation factor, ω, for SOR by relating the eigenvalues to the Jacobi iteration matrix as was the case for Young's T matrices. Nevertheless, if the matrix A in (1) is a Stieltjes matrix, a "good" estimate of ω can be found so that any ordering (and hence multicolor orderings) will yield a convergence rate which is at least one-half the convergence rate associated with the optimal ordering. This result is due to Kahan and can be found in Young [1971].

The development also makes clear some of the issues involved in implementing multicolor SOR. Insensitivity to a cyclic permutation of the colors and the equivalence of a reversed color cycle to a backwards rowwise sweep for symmetric matrices are cases in point. A particularly nice correspondence is that of the points $(-1, \alpha)$ and (γ, β) to the backward and forward synchronization points of the shared memory MIMD implementation.

The technique of using data flow ideas to find orderings clearly extends to regions of three or more dimensions as does the shared memory MIMD approach of engaging multiple processes in simultaneous rowwise sweeps. The equivalence of a fast scheduling of the data flow operations to a multicolor scheme will again depend somewhat on stencil geometry so the idea of an (α, β, γ)-SO stencil will need to be generalized.

The extension to block SOR can be done by defining a block multicolor T matrix and by considering the (α, β, γ)-SO stencil to represent connectivity between blocks of nodes. The convergence rate of block SOR for certain block multicolor orderings of mesh problems leading to irreducible Stieltjes matrices has been compared to a 2-line SOR scheme by O'Leary [1983]. Our results can be used to show that her interesting P^3 ordering leads to a block multicolor T matrix for which the associated block multicolor SOR matrix has the same eigenvalues as a block columnwise SOR matrix.

For point SOR with multiple equations per node a multicolor scheme can be formulated in which equation 1 is evaluated for all points of one color, followed by equation 2 for that color, etc., before moving to the next color. An argument equivalent to the current one can be used to show the SOR matrix has the same eigenvalues as that for a rowwise sequential sweep of the grid points with all equations being evaluated at a point in the same order as for the color groups in the parallel method.

Appendix.

Proof of Theorem 2. Let S represent the stencil

$t^{(k)}(i,j)$ = the earliest time that iteration k can begin for node (i,j),

$S_R = \{i | (i,l) \in S \text{ for some } l \geqq 0\}$,

$S_C = \{j < 0 | (0,j) \in S\}$,

$\Delta_j = \max_{l \geqq 0} l$ where $(j,l) \in S$.

The proof is by induction on k.

The proof for $k=1$ is by induction on i. For $i=1$, the (0, 1)-SO stencil guarantees that $t^{(1)}(1,j+1) = t^{(1)}(1,j)+1$. We assume that

$$t^{(1)}(g,j+1) = t^{(1)}(g,j)+1$$

for $g < i$ and prove that

$$t^{(1)}(i,j+1) = t^{(1)}(i,j)+1. \tag{A.1}$$

Now,

$$t^{(1)}(i,j+1) = \max\{\max_{\{h \leqq j | h-j-1 \in S_C\}} [t^{(1)}(i,h)+1], \max_{\{g<i | g-i \in S_R\}} [t^{(1)}(g,j+1)+\Delta_{g-1}+1]\}. \tag{A.2}$$

If $\{g < i | g-i \in S_R\}$ is empty, then (A.1) follows immediately from an induction on j; otherwise, (A.2) can be written as

$$t^{(1)}(i,j+1) = \max\{\max_{\{h \leqq j | h-j-1 \in S_C\}} t^{(1)}(i,h), \max_{\{g<i | g-i \in S_R\}} [t^{(1)}(g,j)+\Delta_{g-i}+1]+1\}. \tag{A.3}$$

Since

$$t^{(1)}(i,1) = \max_{\{g<i | g-i \in S_R\}} [t^{(1)}(g,1)+\Delta_{g-i}+1],$$

(A.1) follows from an induction on j.

Secondly, we assume $t^{(k)}(i,j+1) = t^{(k)}(i,j)+1$ and prove

$$t^{(k+1)}(i,j+1) = t^{(k+1)}(i,j)+1. \tag{A.4}$$

Now,

$$t^{(k+1)}(i,j+1) = \max\{\max_{\{h \leqq j | h-j-1 \in S_C\}} [t^{(k+1)}(i,h)]+1, \max_{\{g<i | g-i \in S_R\}} [t^{(k+1)}(g,j+1)+\Delta_{g-i}+1], \max_{\{g \geqq i | g-i \in S_R\}} [t^{(k)}(g,j+1)+\Delta_{g-i}+1]\}. \tag{A.5}$$

Since $\{g \geqq i | g-i \in S_R\}$ is not empty for (0, 1)-SO stencils, the last term may be written as

$$\max_{\{g \geqq i | g-i \in S_R\}} [t^{(k)}(g,j)+\Delta_{g-i}+1]+1. \tag{A.6}$$

Now, if $\{g < i | g-i \in S_R\}$ is empty, (A.4) follows from an induction on j; otherwise

the second term in (A.5) can be written as

$$\max_{\{g<i|g-i\in S_R\}} [t^{(k+1)}(g,j)+\Delta_{g-i}+1]+1 \tag{A.7}$$

and (A.4) follows from an induction on j and the theorem is proved.

Proof of Theorem 3. It is sufficient to prove that all nodes can update every c time units, since the block definitions and the coloring rule given in the theorem trivially follow this fact. Since an (α, β, γ)-SO stencil is also a $(0, 1)$-SO stencil, Theorem 1 implies that we only need to prove that

$$t^{(k+1)}(i,1)-t^{(k)}(i,1)=\gamma(\alpha+1)+(\beta+1)=c \tag{A.8}$$

for $k\geqq 1$.

Let S_R, S_C, and Δ_j be as defined in the proof of Theorem 2. Now, the first iteration can begin at node $(i, 1)$ at time,

$$t^{(1)}(i,1)=\max_{\{g<i|g-i\in S_R\}} [t^{(1)}(g,1)+\Delta_{g-1}+1]. \tag{A.9}$$

By the definition of an (α, β, γ)-SO stencil,

$$\Delta_{g-i}\leqq\begin{cases}(i-g)(\alpha+1)-1 & \text{if } g-i<0,\\ (\gamma+i-g)(\alpha+1)+\beta & \text{if } g-i\geqq 0\end{cases} \tag{A.10}$$

with equality when $g-i=-1$ and when $g-i=\gamma$.

By using (A.10), (A.9) can be written as

$$t^{(1)}(i,1)=\max\{\max_{\{g<i-1|g-i+1\in S_R\}} [t^{(1)}(g,1)+\Delta_{g-i}+1],\ t^{(1)}(i-1,1)+(\alpha+1)\} \tag{A.11}$$

and if we choose $t^{(1)}(1,1)=1$, it follows from an induction on i that

$$t^{(1)}(i,1)=1+(i-1)(\alpha+1). \tag{A.12}$$

Now,

$$\begin{aligned}t^{(2)}(i,1)-t^{(1)}(i,1)=\max\{&\max_{\{g<i|g-i\in S_R\}} [t^{(2)}(g,1)-t^{(1)}(i,1)+\Delta_{g-i}+1],\\ &\max_{\{g\geqq i|g-i\in S_R\}} [t^{(1)}(g,1)-t^{(1)}(i,1)+\Delta_{g-i}+1]\}.\end{aligned} \tag{A.13}$$

When $i=1$,

$$t^{(2)}(1,1)-t^{(1)}(1,1)=\max\{[t^{(1)}(\gamma+1,1)+\beta],\ \max_{\{g\geqq 1|g-1\in S_R\}} [t^{(1)}(g,1)+\Delta_{g-i}]\}$$

and $t^{(2)}(1,1)-t^{(1)}(1,1)=1+\gamma(\alpha+1)+\beta$ from (A.10), (A.12), and from an induction of i in (A.13), (A.8) follows for $k=1$.

Next, assume (A.8) is true for $k-1$, and prove true for k.

Now,

$$\begin{aligned}t^{(k+1)}(i,1)-t^{(k)}(i,1)=\max\{&\max_{\{g<i|g-i\in S_R\}} [t^{(k+1)}(g,1)-t^{(k-1)}(i,1)-t+\Delta_{g-i}+1],\\ &\max_{\{g\geqq i|g-i\in S_R\}} [t^{(k-1)}(g,1)-t^{(k-1)}(i,1)+\Delta_{g-i}+1]\}.\end{aligned} \tag{A.14}$$

When $i=1$,

$$t^{(k+1)}(1,1)-t^{(k)}(1,1)=\max_{\{g\geqq 1|g-1\in S_R\}} [t^{(k-1)}(g,1)-t^{(k-1)}(1,1)+\Delta_{g-1}+1]$$

and

$$t^{(k+1)}(1,1) - t^{(k)}(1,1) = t^{(k)}(1,1) - t^{(k-1)}(1,1) = c$$

from (A.10) and (A.12). Hence (A.14) is true for $i = 1$, and by induction on i in equation (A.14), the theorem is proven.

Acknowledgments. Thanks are due Nisheeth Patel for the practical side stimulus which led to a discussion between one of us, Burton Smith, Dennis Gannon, Kenneth Batcher and Gary Rodrigue at a forum in Oregon, expertly hosted by Bill Buzbee of Los Alamos and George Michael of Lawrence Livermore National Laboratories. Enthusiastic and valuable support was provided by Milton Rose, Robert Voigt, and Merrell Patrick both personally and through the ICASE working environment. James Ortega also contributed discussion and enthusiasm for the work.

REFERENCES

W. B. ACKERMAN, *Data flow languages*, Computer, 15 (1982), pp. 15-25.

L. M. ADAMS, *Iterative algorithms for large sparse linear systems on parallel computers*, Ph.D. dissertation, University of Virginia; also published as NASA CR-166027, NASA Langley Research Center, Hampton, VA, November, 1982.

L. M. ADAMS AND J. M. ORTEGA, *A multi-color* SOR *method for parallel computation*, Proc. 1982 International Conference on Parallel Processing, Bellaire, MI, August 1982, pp. 53-58.

L. ADAMS, *An M-step preconditioned conjugate gradient method for parallel computation*, Proc. 1983 International Conference on Parallel Processing, Bellaire, MI, August 1983, pp. 36-43.

B. L. BUZBEE, G. H. GOLUB AND J. A. HOWELL, *Vectorization for the* CRAY-1 *of some methods for solving elliptic difference equations*, High Speed Computer and Algorithm Organization, D. J. Kuck, D. H. Lawrie and A. A. Sameh, eds., Academic Press, New York, 1977, pp. 255-272.

DIANNE P. O'LEARY, *Ordering schemes for parallel processing of certain mesh problems*, this Journal, 5 (1984), pp. 620-632.

N. R. PATEL AND H. F. JORDAN, *A parallelized point successive over-relaxation method on a multiple instruction multiple data stream computer*, submitted to Parallel Computing.

D. YOUNG, *Iterative methods for solving partial differential equations of elliptic type*, Doctoral thesis, Harvard Univ., Cambridge, MA, 1950.

———, *Iterative Solution of Large Linear Systems*, Academic Press, New York, 1971.

SIAM J. SCI. STAT. COMPUT.
Vol. 7, No. 2, April 1986

011

A NOTE ON THE STABILITY OF SOLVING A RANK-p MODIFICATION OF A LINEAR SYSTEM BY THE SHERMAN-MORRISON-WOODBURY FORMULA*

E. L. YIP†

Abstract. In this paper, we address the stability of the Sherman-Morrison-Woodbury formula. Our main result states that if the original matrices, A and B, are well conditioned, then there exists matrices U and V such that the Sherman-Morrison-Woodbury formula is stable when applied to $A = B - UV^T$.

Key words. updating Sherman-Morrison-Woodbury updating formula, updating, condition number, stability

1. Introduction. In this note, we address the stability of the Sherman-Morrison-Woodbury updating formula as a method for solving (updated) systems of linear equations $Ax = b$ where there exists a matrix B such that $A - B$ contains only a few nonzero rows (columns) and where $By = b$ can be solved efficiently. Such systems occur quite frequently in practice. For example, Yip [5] describes an out-of-core subprogram designed for the solution of the difference equation for the 2-D transonic flow equation with small disturbances; A is a banded matrix plus eight nonzero columns and B is the banded matrix such that $A - B$ is a matrix with eight nonzero columns. The capacitance matrix methods described by Hockney [6], Buzbee et al. [7], and Proskurowski et al. [8] are equivalent to updating rows of a finite element matrix corresponding to the irregular mesh points. Also, Bunch and Rose [10] discuss a linear equation solution technique known as "tearing" for systems of the form $(B + V\Sigma W^T)$ where B and Σ are respectively $n \times n$ and $r \times r$ matrices, and V and W are $n \times r$ matrices, with r much less than n.

The Sherman-Morrison-Woodbury updating formula is an efficient method for a more general updating problem with $A - B$ of low rank, and not necessarily consisting of a few nonzero rows or columns. However, in this note, we mainly concentrate on the case where $A - B$ contains p linearly independent and $(n - p)$ zero rows (columns), where n is the order of the matrices A and B. We show that in this case we can easily choose U and V such that $A - B = UV^T$ and such that the Sherman-Morrison-Woodbury formula is stable. We show that for the general case when $A - B$ is of rank p, such U and V also exist, but they may not be practical to compute.

For the sake of completeness, we first present the Sherman-Morrison-Woodbury formula and the stability analysis done by Stewart [9] for the case $p = 1$. We present our main results in § 2. In § 3, we prove the lemma on which the proofs of our main results are based. In § 4, we present some numerical examples which confirm our theoretical results.

If A and B are $n \times n$ matrices, and if $A - B$ is a rank p matrix, there exist $n \times p$ matrices U and V such that $A = B - UV^T$. The Sherman-Morrison-Woodbury updating formula expresses A^{-1} in terms of B^{-1}, U and V:

$$A^{-1} = B^{-1} + B^{-1}U(I - V^TB^{-1}U)^{-1}V^TB^{-1}. \tag{1.1}$$

There are many different implementations of the above equation for the solution of

* Received by the editors February 21, 1984, and in revised form January 3, 1985.
† Boeing Aerospace Company, Seattle, Washington 98124.

$Ax = b$. The capacitance matrix method in [6], [7], and [8] is an implementation designed for the special case when $B - A$ is a matrix with p nonzero rows and B is such that systems of the form $By = b$ can be solved efficiently by Fast Poisson Solvers. Algorithm 1 below is an implementation for the most general case, that is, when $B - A$ is an arbitrary rank p matrix.

ALGORITHM 1
Step 1. Solve $BZ = U$.
Step 2. Compute the matrix $K = (I - V^T Z)$, and its LU factorization.
Step 3. Solve $By = b$.
Step 4. Compute $w = V^T y$.
Step 5. Solve $Ks = w$.
Step 6. The solution x for $Ax = b$ can be computed as $x = y + Zs$.

If we postmultiply (1.1) by the vector b, we obtain:

$$A^{-1}b = B^{-1}b + B^{-1}U(I - V^T B^{-1}U)^{-1}V^T B^{-1}b. \tag{1.2}$$

Substituting the matrices Z and K and the vectors y, w, and s, which are defined in Algorithm 1, into (1.2), we see that, in the absence of numerical round-off, the vector x defined in Step 6 of Algorithm 1 equals $A^{-1}b$.

If A and B are full matrices, and if B has already been factored, then the amount of work in Algorithm 1 is of the order $p(n^2 + p^2/3)$; if p is small, this can be much less costly than factoring A.

Assume the matrix B is well scaled and well-conditioned. The stability of Algorithm 1 depends on the scaling between the matrices B, U and V and the right-hand side vector b. Any discussion of the condition of the matrix K in step 2 must include consideration of the scaling of the matrices B, U and V. The following example illustrates the problem relating to scaling:

Example 0. A and B are $n \times n$ scalar matrices $(\varepsilon - 1)I$ and εI respectively, with ε of very small magnitude. Thus $p = n$. We can choose $U = V = I$.

Note that in this example, both of the matrices A and B are well conditioned. However, Algorithm 1 is unstable, because disastrous cancellation occurs in the computation of the matrix K.

We say the method is stable if each step of Algorithm 1 is stable. Table 1 gives a summary of the factors affecting the stability of each step.

TABLE 1

Step	Factors affecting its stability
1. $BZ = U$	condition of B and scaling of B and U
2. $K = (I - V^T Z)$	scaling of V, B and U
3. $By = b$	condition of B and scaling of B and b
4. $w = V^T y$	scaling of V, B and b
5. $Ks = w$	condition of K
6. $x = y + Zs$	accuracy of previous steps

In this paper, we examine the question of assessing the conditioning of $(I - V^T B^{-1}U)$ in terms of the conditioning of A and B. Combined with standard methods of monitoring conditioning and stability in the other steps, we would then have a way of assessing the overall stability of the process.

Stewart [9] used Algorithm 1 for the modification of pivot elements in Gaussian elimination and proved that when $p = 1$, Algorithm 1 is stable provided the solution

method for $By = c$ is stable. In this note, we show that when $A - B$ has exactly $p > 1$ nonzero rows (columns), we can easily choose U and V such that Algorithm 1 is stable. In the section on numerical examples, we also demonstrate (in Example 2) that one can choose U and V such that Algorithm 1 is unstable in spite of A and B being well-conditioned.

2. Main result. We define the condition numbers $\kappa(A) = \|A\| \, \|A^{-1}\|$, with $\| \ \|$ being an arbitrary matrix norm, and $\kappa_2(A) = \|A\|_2 \|A^{-1}\|_2$ where $\| \ \|_2$ is the spectral norm. We also define e_i to be the vector with 1 in its ith entry and zero everywhere else. Our main results are as stated in Theorems 1 and 2, the proofs of which depend on the following Lemma:

LEMMA. *If A and B are nonsingular $n \times n$ matrices, and U and V are $n \times p$ matrices such that $A = B - UV^T$, with U and V of full column rank, then*

$$\kappa(I - V^T B^{-1} U) \leqq \min\{k_1, k_2\} \kappa(A) \kappa(B), \tag{2}$$

where

$$k_1 = (\|U\| \, \|U^+\|)^2, \qquad k_2 = (\|V^T\| \, \|(V^T)^+\|)^2$$

with

$$U^+ = (U^T U)^{-1} U^T, \qquad (V^T)^+ = V(V^T V)^{-1}.$$

We shall prove the lemma in § 3.

THEOREM 1. *Suppose A and B are $n \times n$ nonsingular matrices with $B - A$ of rank p. Then there exist $n \times p$ matrices U and V such that $A = B - UV^T$ and the following inequality holds:*

$$\kappa_2(I - V^T B^{-1} U) \leqq \kappa_2(A) \kappa_2(B). \tag{3}$$

Proof of Theorem 1. There exists a $n \times n$ orthonormal matrix P such that the first p rows of $P^T(B - A)$ are nonzero and the last $(n - p)$ rows are zero. Let U be the $n \times p$ matrix whose columns are the first p columns of P, and let V^T be the $p \times n$ marrix whose rows are the nonzero rows of $P^T(B - A)$. Then $(B - A) = UV^T$. (For example, as suggested by Lewis [2], one can form the singular value decomposition of $B - A = P\Sigma Q$, where P and Q are orthonormal, and Σ is a nonnegative diagonal matrix whose last $(n - p)$ diagonal entries are zeros. $P\Sigma Q = Q_1 S Q_2$, where S a $p \times p$ diagonal matrix with positive diagonal entries and $Q_1 Q_1^T = Q_2 Q_2^T = I_p$. Let $U = Q_1$, and $V^T = SQ_2$; then $(B - A) = UV^T$.) With such a choice of U, $\|U\|_2 = 1$. Thus k_1 defined in the lemma equals 1, and inequality (2) implies inequality (3).

The proof of Theorem 1 suggests that the application of the theorem depends on finding an orthonormal matrix P such that $P^T(B - A)$ has p nonzero rows and $(n - p)$ zero rows or $(B - A)P^T$ has p nonzero columns and $(n - p)$ zero columns. We have proved that such P exists, but the actual computation of such matrices for arbitrary $(B - A)$ seems to be impractical. However, as cited in § 1 of this paper, many practical problems involving rank p modification are such that $B - A$ are has $(n - p)$ zero rows (columns). That is, for these problems, the matrix P is just the identity matrix. We summarize this special case in Theorem 2, which is actually a corollary to Theorem 1.

THEOREM 2. *If A and B are nonsingular $n \times n$ matrices such that $B - A$ has $(n - p)$ zero rows (columns), and the remaining p nonzero rows (columns) are linearly independent, then there exist $n \times p$ matrices U and V such that $(B - A) = UV^T$, and inequality (4) is true.*

$$\kappa(I - V^T B^{-1} U) \leqq \kappa(A) \kappa(B). \tag{4}$$

Proof of Theorem 2. Suppose $(B-A)$ has p nonzero rows with indices $i_1, i_2, \cdots, i_p$. We choose U to be the matrix whose columns are $e_{i_1}, e_{i_2}, \cdots, e_{i_p}$, and V to be the matrix such that the rows of V^T are nonzero rows of $B-A$. If $B-A$ has p columns of nonzeros with the same indices, we choose V to be the matrix whose columns are $e_{i_1}, e_{i_2}, \cdots, e_{i_p}$, and U to be the matrices whose columns are the nonzero columns of $B-A$. Then k_1 defined in the lemma is unity when $B-A$ has p nonzero rows, and k_2 is unity when $B-A$ has p nonzero columns. Thus in both cases $\min\{k_1, k_2\}=1$, and inequality (4) follows from inequality (2).

The proof of Theorem 2 suggests that the structure of $(B-A)$ provides a ready choice of U and V so that the application of the Sherman-Morrison-Woodbury updating formula is stable.

3. Proof of lemma. We define the $p\times n$ matrix U^+ and $n\times k$ matrix $(V^T)^+$ as follows:

$$U^+ = (U^TU)^{-1}U^T, \tag{5a}$$

$$(V^T)^+ = V(V^TV)^{-1}. \tag{5b}$$

Notice that $U^+U = (V^T)(V^T)^+ = I_p$, the identity matrix of order p.

Postmultiplication of both sides of the equation $A = B - UV^T$ by $B^{-1}U$ gives

$$AB^{-1}U = U - UV^TB^{-1}U = U(I - V^TB^{-1}U).$$

Applying U^+ on the left yields

$$(I - V^TB^{-1}U) = U^+AB^{-1}U, \tag{6}$$

so

$$(I - V^TB^{-1}U)^{-1} = U^+BA^{-1}U. \tag{7}$$

Thus

$$\begin{aligned}\kappa(I - V^TB^{-1}U) &= \|(I - V^TB^{-1}U)\|\,\|(I - V^TB^{-1}U)^{-1}\| \\ &= \|U^+AB^{-1}U\|\,\|U^+BA^{-1}U\| \\ &\leqq (\|U^+\|\,\|U\|)^2\|B\|\,\|B^{-1}\|\,\|A\|\,\|A^{-1}\| \\ &= k_1\kappa(B)\kappa(A).\end{aligned} \tag{8}$$

Similarly, we can prove

$$\kappa(I - v^TB^{-1}U) \leqq k_2\kappa(B)\kappa(A). \tag{9}$$

Combining (8) and (9), we obtain (3).

4. Numerical examples. Our lemma provides a bound for the condition number of $(I - V^TB^{-1}U)$. Unfortunately, one can construct examples such that the bound provided in our lemma is very large. Consider the following example:

Example 1. Let B be the 4×4 identity matrix. Let A be the diagonal matrix such that $A = \text{diag}\,(1+10^{-5}, 1+10^{-5}, 1, 1)$. In other words, $B - A = \text{diag}\,(10^{-5}, 10^{-5}, 0, 0)$. Let

$$U = \begin{bmatrix} 10^{-5} & 0 \\ 0 & 1 \\ 0 & 0 \\ 0 & 0 \end{bmatrix}, \qquad V = \begin{bmatrix} 1 & 0 \\ 0 & 10^{-5} \\ 0 & 0 \\ 0 & 0 \end{bmatrix}.$$

Then $k_1 = k_2 = 10^{10}$, and $(I - V^T B^{-1} U) = \text{diag}\,(1+10^{-5}, 1+10^{-5})$. The lemma implies that $\kappa(I - V^T B^{-1} U) \leqq 10^{10}$, but in reality, $\kappa(I - V^T B^{-1} U)$ is close to unity!

There are infinitely many ways to choose the $n \times p$ matrices U and V so that $A - B = UV^T$ since

$$UV^T = UXX^{-1}V^T = (UX)(X^{-1}V^T) = U_X V_X^T \quad \text{with } U_X = UX, \text{ and } V_X^T = X^{-1}V^T,$$

where X is a $p \times p$ arbitrary nonsingular matrix. We want to emphasize the fact that the inequalities (3) and (4) do not hold for all possible choices of U and V. Example 2 demonstrates this point. Let $\|\cdot\|$ be the 1-norm, and consider the following example:

Example 2. Let B be the 4×4 identity matrix, let A be defined as

$$A = \begin{bmatrix} -2 & -10^{-1} & 0 & 0 \\ -10^{-9} & -10^{-10}+1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

Suppose U and V are as follows:

$$U = \begin{bmatrix} 2 & 10^4 \\ 0 & 10^{-5} \\ 0 & 0 \\ 0 & 0 \end{bmatrix}, \qquad V = \begin{bmatrix} 1 & 10^{-4} \\ 0 & 10^{-5} \\ 0 & 0 \\ 0 & 0 \end{bmatrix}.$$

Then

$$(I - V^T B^{-1} U) = \begin{bmatrix} -1 & -10^4 \\ -2 \cdot 10^{-4} & 10^{-5} \end{bmatrix}.$$

We note that $\kappa(B) = 1$, $\kappa(A) = 2$ but $\kappa(I - V^T B^{-1} U)$ is of the order of 10^8. In this example, inequality (4) fails, but inequality (2) of our lemma provides a realistic bound for $\kappa(I - V^T B^{-1} U)$ because $\min\{k_1, k_2\} = k_2 = 10^{10}$.

If we choose U and V according to the proof of Theorem 2, then we have a much better condition number for $(I - V^T B^{-1} U)$. Consider the following example:

Example 3. Let A and B as defined for Example 2, but let

$$U = \begin{bmatrix} 3 & 10^{-1} \\ 10^{-9} & 10^{-10} \\ 0 & 0 \\ 0 & 0 \end{bmatrix}, \qquad V = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \\ 0 & 0 \end{bmatrix}.$$

We see that $(I - V^T B^{-1} U) = (I - U_1)$ where U_1 is the two nonzero rows of U and $\kappa(I - V^T B^{-1} U)$ has condition number of the order 2.

In the remainder of this section, we present the numerical resulrs of some medium size rows (columns) updating matrix problems to demonstrate the relations between $\kappa(I - V^T B^{-1} U)$, $\kappa(A)$, $\kappa(B)$, and the solution error. U and V are computed according to Theorem 1. Problem 1 is due to Grimes in Erisman et al. [1]. It is a tridiagonal system with a P5 ordering applied to it. This is used in [1] as a counterexample to demonstrate that the P5 order is not always acceptable on numerical grounds. The

reordered matrix is of the form:

$$A = \begin{bmatrix} 1 & & & & & -4 \\ -4 & 1 & & & & 1 \\ 0 & -4 & 1 & & & 0 \\ & \cdot & \cdot & \cdot & & \cdot \\ & & \cdot & 1 & & 0 \\ & & & -4 & 1 & 0 \\ & & & & 4 & 0 \end{bmatrix}. \tag{10}$$

Erisman et al. solved the linear system $Ax = b$, where A is defined in equation (10) by applying the Sherman-Morrison updating formula with a rank 1 update. In their application, the matrix B is the lower triangular matrix:

$$B = \begin{bmatrix} 1 & & & & & 0 \\ -4 & 1 & & & & 0 \\ 0 & -4 & 1 & & & 0 \\ & \cdot & \cdot & \cdot & & \cdot \\ & & \cdot & 1 & & 0 \\ & & & -4 & 1 & 0 \\ & & & & 4 & x \end{bmatrix}.$$

Then $A - B$ is a rank 1 matrix with one nonzero column vector which has -4 and 1 in its first and second entries, and $-x$ in its last entry. Systems of the form $By = b$ are being solved by forward substitution. Problems 2-6 are full matrices generated by a random number generator. In problem 6, we add 100 to the diagonals of both A and B to obtain extremely well-conditioned matrices. We use LINPACK [3] subroutine SGECO to compute the condition number of the matrices. All the problems are of order 50. In problem 1, in order to duplicate the computational sequence of Erisman et al., we input the transpose of A and B into SGECO so as to eliminate pivoting. The experiment was done on the CRAY-1 which has 15 digit precision. Table 2 summarizes our results. (The symbols p-nzr and p-nzc mean p nonzero rows and p nonzero columns respectively.)

TABLE 2

Problem	$B-A$	$\kappa(A)$	$\kappa(B)$	$\kappa(I - V^T B^{-1} U)$	Solution error
1	1-nzr	2.96 E0	7.38 E28	1.00 E0	2.81 E14
2	38-nzc	3.80 E2	1.62 E3	4.98 E3	2.33 E−12
3	5-nzc	1.12 E3	4.79 E2	2.03 E2	2.19 E−12
4	25-nzr	1.30 E3	7.47 E2	1.31 E3	1.78 E−11
5	9-nzr	6.68 E3	1.40 E3	1.56 E3	9.12 E−12
6	15-nzc	1.30 E0	1.31 E0	1.08 E0	9.95 E−14

Acknowledgment. The author wants to thank Dr. John G. Lewis for reviewing an earlier draft of this paper and for his contribution in Theorem 1, and Dr. Al Erisman for Example 0 which illustrates the problem with scaling. Part of this work was done under NASA contract NAS1-15128 (see [5] below).

REFERENCES

[1] A. M. ERISMAN, R. G. GRIMES, J. G. LEWIS, W. G. POOLE AND H. D. SIMON, *Evaluation of ordering for unsymmetric sparse matrices*, this Journal.

[2] J. G. LEWIS, *private communication*, 1983.

[3] LINPACK *Users' Guide*, Society for Industrial and Applied Mathematics, Philadelphia, 1979.

[4] A. L. CLINE, C. B. MOLER, G. W. STEWART AND J. H. WILKINSON, *An estimate for the condition number of a matrix*, SIAM J. Numer. Anal., 16 (1979), pp. 368-375.

[5] E. L. YIP, FORTRAN *subroutines for out-of-core solution of large complex linear systems*, prepared for Langley Research Center under contract NAS1-15128, NASS CR 159142, 1979.

[6] R. HOCKNEY, *The potential calculation and some applications*, Math. Comput. Phys., 9 (1970), pp. 135-211.

[7] B. BUZBEE, F. DORR, J. GEORGE AND G. GOLUB, *The direct solution of the discrete Poisson equation for irregular regions*, SIAM J. Numer. Anal., 8 (1971), pp. 722-736.

[8] W. PROSKUROWSKI AND O. WIDLUND, *On the numerical solution of Helmholtz's equation by the capacitance matrix method*, Math. Comput., 30 (1976), pp. 433-468.

[9] G. W. STEWART, *Modifying pivot elements in Gaussian elimination*, Math. Comput., 28 (1974), pp. 527-542.

[10] J. R. BUNCH AND D. J. ROSE, *Partitioning, tearing, and modification of sparse linear systems*, J. Math. Anal. Appl., 48 (1974), pp. 574-593.

SIAM J. SCI. STAT. COMPUT.
Vol. 7, No. 2, April 1986

012

DELAUNAY TRIANGULAR MESHES IN CONVEX POLYGONS*

BARRY JOE†

Abstract. An algorithm for producing a triangular mesh in a convex polygon is presented. It is used in a method for the finite element triangulation of a complex polygonal region of the plane in which the region is decomposed into convex polygons. The interior vertices of the mesh are chosen to be on a quasi-uniform grid, different mesh spacings are specified for the edges of the polygon, and the mesh is a Delaunay triangulation. The correctness of the algorithm is proved and the expected time complexity is shown to be linear in the number of triangles in the mesh.

Key words. mesh generation, finite element method, computational geometry, Delaunay triangulation

AMS (MOS) subject classifications. 65N50, 68Q20, 68Q25

1. Introduction. A common approach for generating triangular meshes in general regions of the plane for the finite element method is to first decompose the region into simpler subregions and then to triangulate each subregion (Cavendish (1974), Shaw and Pitchen (1978), Bykat (1976), Bank (1982)). We have developed a method for the finite element triangulation of a complex polygonal region of the plane in which the region is decomposed into convex polygons such that small interior angles in the polygons are avoided and then a triangular mesh is generated in each convex polygon such that triangules with small angles are avoided (Joe and Simpson (1984), Joe (1984)). Figure 1.1 illustrates a triangulation of a region produced by this method.

In this paper, we describe the algorithm used in this method for generating a triangular mesh in a convex polygon P, and prove its correctness. The triangles of the mesh satisfy an optimal angle criterion, i.e. the mesh is a Delaunay triangulation (Lawson (1977)). Let $e_1, e_2, \cdots, e_m$ be the m edges of P in counterclockwise order. The input of the algorithm consists of the vertices of P in counterclockwise order, triangle size parameter h for the interior of P, and triangle size parameters $h_1, h_2, \cdots, h_m$ for the edges $e_1, e_2, \cdots, e_m$, respectively. Vertices are generated in the interior of P using a quasi-uniform grid of spacing h (cf. Shaw and Pitchen (1978)). Vertices are generated on edge e_i at an equal spacing of approximately h_i. In the Delaunay triangulation of these vertices, the triangles in the interior of P have constant area $h^2/2$. Different h_i are specified on the e_i so that triangles which are graded in size are produced near ∂P (the boundary of P).

In our finite element triangulation method, the triangle size parameters for the interior of the convex polygons in the decomposition of a region are restricted to differ by a factor of at most two in adjacent polygons so that a gradual change of triangle sizes occurs between adjacent polygons. The triangle size parameter for an edge of the decomposition is set to the geometric mean of the triangle size parameters for the interior of the one or two convex polygons with this edge. Hence the parameters h_i for the edges of a convex polygon P in the decomposition satisfy

$$h/\sqrt{2} \leqq h_i \leqq \sqrt{2}h. \tag{1.1}$$

With this restriction, the triangles near ∂P have approximately constant area $h^2/2$ and the number of triangles in P is approximately $2a/h^2$ where a is the area of P, so the

* Received by the editors July 17, 1984, and in revised form December 14, 1984. This work was partially supported by a grant from the Natural Sciences and Engineering Research Council of Canada.

† Department of Computer Science, University of Waterloo, Waterloo, Ontario, Canada N2L 3G1.

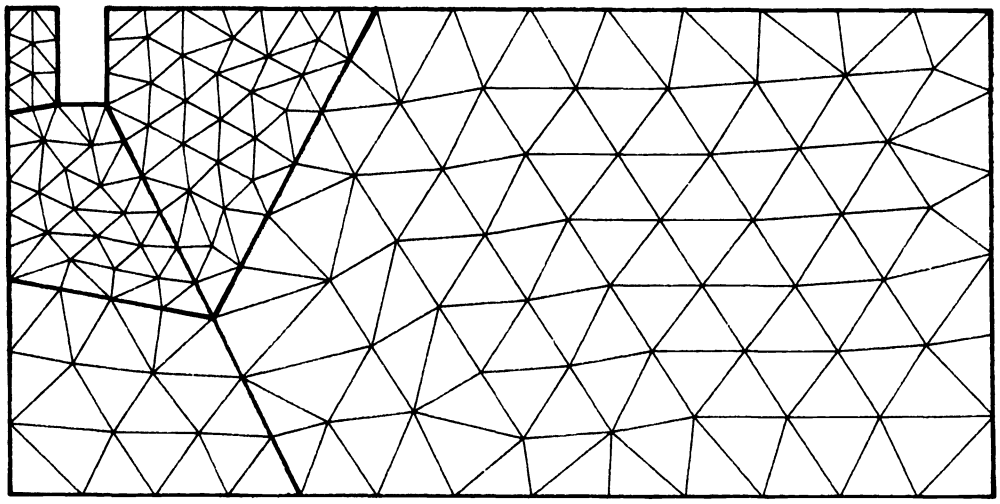


FIG. 1.1. *Illustration of triangulation of a region; thicker lines indicate decomposition into convex polygons.*

triangle size parameter h can be determined from a and the desired number of triangles in the mesh.

In the standard triangulation problem of computational geometry, the vertices are given as input and a (Delaunay) triangulation is constructed to cover the convex hull of the vertices. The construction of a Delunay triangulation (or any triangulation) of the vertices requires $O(n_t \log n_t)$ time where n_t is the number of triangles in the triangulation (Shamos and Hoey (1975), Lee and Schachter (1979)). However, in finite element triangulation, the vertices are generated as well as the triangles. By using information about the location of the vertices relative to each other, our algorithm constructs a Delaunay triangulation of the vertices in an expected time of $O(n_t)$ when the h_i satisfy (1.1) and P contains no "small" interior angles.

In § 2, we define a valid triangulation and a Delaunay triangulation. In § 3, we describe an algorithm for shrinking a convex polygon which we use to determine a convex polygon, $int(P)$, in the interior of P. In our triangulation algorithm, we construct a preliminary triangulation, $VT(P)$, by generating triangles in $int(P)$ and in the strip near ∂P, as described in §§ 4 and 5, respectively. The validity of $VT(P)$ is discussed in § 6. In § 7, we describe how $VT(P)$ is converted into a Delaunay triangulation, $DT(P)$. The time complexity of the triangulation algorithm is discussed in § 8.

2. Valid and Delaunay triangulations. In this section we state conditions for a valid triangulation in a region and state some properties of a Delaunay triangulation of a set of vertices. We say that a collection of triangles is a valid triangulation of a polygonal region of the plane if the triangles form a "tiling" of the region without overlaps or gaps. Let w_1, w_2, w_3 be distinct vertices. We define triangle $\Delta w_1 w_2 w_3$ to be a counterclockwise (CCW) triangle if the interior of the triangle is to the left of the three directed edges $w_1 w_2$, $w_2 w_3$, $w_3 w_1$; otherwise $\Delta w_1 w_2 w_3$ is a clockwise (CW) triangle. If w_1, w_2, w_3 are collinear, we consider $\Delta w_1 w_2 w_3$ to be a CW triangle. Note that the ordering of the vertices of a triangle determines whether it is a CCW or CW triangle. Four conditions that ensure that a collection of triangles, $\Delta_1, \Delta_2, \cdots, \Delta_N$, is a valid triangulation of a region have been established by Simpson (1981). They can be described in geometric terms as follows:

(a) Δ_i is a CCW triangle for all i.

(2.1)
(b) Each edge e of Δ_i is either the only edge joining its vertices or there is one other triangle, Δ_j, having an edge joining these vertices. In the latter case, the direction of e as an edge of Δ_i is opposite to its direction as an edge of Δ_j. In the former case, e is a boundary edge and Δ_i is its boundary triangle.
(c) No interval of a boundary edge intersects a triangle other than its boundary triangle.
(d) A vertex can have at most one boundary edge directed away from it.

Let V be a set of vertices in the plane such that they are not all collinear. A Delaunay triangulation of V is a (valid) triangulation in the convex hull of V which satisfies the max-min angle criterion: for any two triangles in the triangulation that share a common edge, if the quadrilateral formed from the two triangles with the common edge as its diagonal is strictly convex, the replacement of the diagonal by the alternative one does not increase the minimum of the six angles in the two triangles making up the quadrilateral (Sibson (1978)). In other words, the Delaunay triangulation is the triangulation which maximizes the minimum angle in the triangles globally as well as locally in any two adjacent triangles which form a strictly convex quadrilateral.

A Delaunay triangulation also satisfies the circle criterion: the circumcircle of any triangle in the triangulation contains no vertex of V in its interior. The Delaunay triangulation is unique if no four vertices are co-circular. An edge uv, where u and v are vertices of V, is a Delaunay edge if it is an edge in a Delaunay triangulation of V.

LEMMA 2.1 (Lee and Schachter (1979)). *An edge uv is a Delaunay edge if and only if there exists a point c such that the circle centered at c and passing through u and v does not contain any other vertex of V in its interior.*

The following local optimization procedure (LOP) of Lawson (1977) can be used to convert a triangulation of V into a Delaunay triangulation. Let e be an internal edge (i.e. an edge not on the boundary of the convex hull) of a triangulation of V and Q be the quadrilateral formed by the two triangles having e as a common edge. If the circumcircle of one of the triangles contains the fourth vertex of Q in its interior, then e is replaced by the other diagonal of Q (so that the minimum of the angles in the two triangles is increased). Edge e is said to be *locally optimal* if an application of LOP would not swap it. Note that e is locally optimal if Q is a nonconvex quadrilateral, and that the validity conditions (2.1) are unaffected by the LOP.

THEOREM 2.1 (Lawson (1977)). *All internal edges of a triangulation T of V are locally optimal if and only if T is a Delaunay triangulation of V.*

In our algorithm, we generate a preliminary valid triangulation, $VT(P)$, in P which satisfies the four conditions of (2.1). Then $VT(P)$ is converted into a Delaunay triangulation, $DT(P)$, by applying LOP to the internal edges of $VT(P)$. We have constructed $VT(P)$ so that LOP only has to be applied to a subset of the internal edges.

3. Shrinking a convex polygon. Let $p_0, p_1, \cdots, p_{m-1}, p_m$ be the vertices of convex polygon P in counterclockwise order, where $p_m = p_0$ and all interior angles are less than 180°. Let Q be obtained by shrinking P by a distance of $r > 0$, i.e. if $v \in Q$ then the distance from v to $\partial P \geqq r$. In this section, we present an algorithm for constructing Q which uses the same approach as the algorithm of Lee and Preparata (1979) for finding the kernel of a simple polygon. They exploit the ordering of the half-planes corresponding to the polygon edges to obtain a linear time algorithm.

First we give some notation for representing Q and ∂Q. Let l_i be the directed line from p_i to p_{i+1}, L_i be the directed line parallel to l_i and at distance r to the left of l_i, and H_i be the half-plane to the left of and including L_i. Let qL_iq' denote the directed line segment from q to q' where q and q' are two points on L_i, and let $qL_i\infty$ and ∞L_iq

denote the directed half-lines starting and ending at q, respectively, on L_i. Q is the intersection of the half-planes $H_0, H_1, \cdots, H_{m-1}$. If r is sufficiently small then Q is a convex polygon; otherwise Q is degenerate, i.e. Q is either empty or a single point or a line segment. In the degenerate case, we treat Q as empty. If Q is a convex polygon then ∂Q consists of line segments from $n \geqq 3$ of the lines L_i (if $\bigcap_{i \neq j} H_i = Q$ then no line segment of L_j is part of ∂Q). Using the above notation

$$\partial Q = q_0 L_{k_0} q_1 L_{k_1} q_2 \cdots q_{n-1} L_{k_{n-1}} q_n, \qquad 0 \leqq k_0 < k_1 < \cdots < k_{n-1} \leqq m-1,$$

where $q_n = q_0$ is the intersection of L_{k_0} and $L_{k_{n-1}}$ and q_i is the intersection of $L_{k_{i-1}}$ and L_{k_i}, $1 \leqq i \leqq n-1$ (see Fig. 3.1).

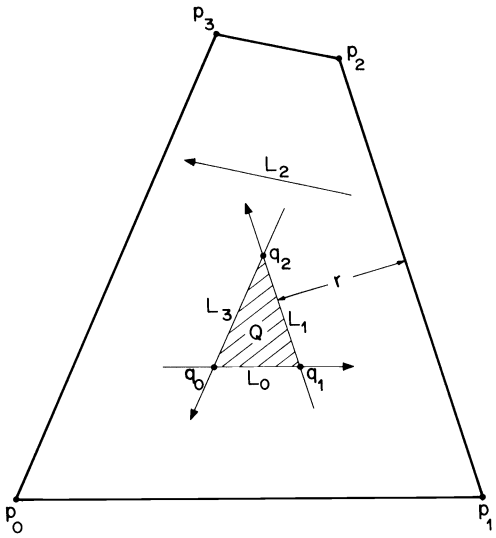


FIG. 3.1. *Illustration of convex polygon Q.*

Our algorithm for constructing Q scans in order the edges $p_i p_{i+1}$ of P and constructs a sequence of convex polygons $Q_1, Q_2, \cdots, Q_{m-1}$ such that Q_i is the intersection of half-planes $H_0, H_1, \cdots, H_i$. Since P is a convex polygon, the polar angles, θ_i, of the vectors $p_{i+1} - p_i$ (or lines l_i) are ordered. Without loss of generality assume $\theta_0 = 0°$, i.e. l_0 is a horizontal line directed from left to right, so that $0° = \theta_0 < \theta_1 < \cdots < \theta_{m-1} < 360°$. Let s be the smallest index such that $\theta_s > 180°$ (note that $2 \leqq s \leqq m-1$). If $\theta_{s-1} = 180°$ then let $s' = s-2$; otherwise let $s' = s-1$. The following facts can be seen from elementary geometry:

(3.1)
(a) For $1 \leqq i \leqq s'$, Q_i is an unbounded convex polygon.
(b) If $\theta_{s-1} = 180°$, Q_{s-1} is either an unbounded convex polygon or degenerate.
(c) For $s \leqq i \leqq m-1$, Q_i is either a bounded convex polygon or degenerate.

Now we describe our algorithm for constructing Q. Initially Q_1 is the intersection of H_0 and H_1 and $\partial Q_1 = \infty L_0 q_1 L_1 \infty$ where q_1 is the intersection of L_0 and L_1. For $2 \leqq i \leqq s'$, Q_i is the intersection of Q_{i-1} and H_i and ∂Q_i is obtained by modifying ∂Q_{i-1}. Suppose

$$(3.2) \qquad \partial Q_{i-1} = q_0 L_{k_0} q_1 \cdots q_n L_{k_n} q_{n+1}, \qquad q_0 = q_{n+1} = \infty, \qquad 0 = k_0 < k_1 < \cdots < k_n = i-1.$$

Since Q_{i-1} is convex and $\theta_j < \theta_i < 180°$ for $j < i$, ∂Q_{i-1} and L_i intersect at exactly one point, t (see Fig. 3.2). Let $j \geqq 0$ be the index such that t is on the line segment $q_j L_{k_j} q_{j+1}$

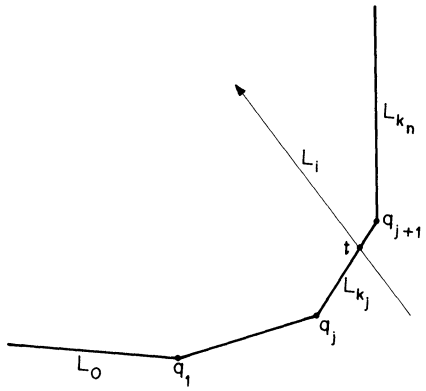


FIG. 3.2. *∂Q_{i-1} and L_i intersect at point t.*

and $t \neq q_j$. j can be found by scanning backwards from n for the first q_j to the left of L_i where $q_0 = \infty$ is considered to be to the left of L_i. The polygonal curve $tL_{k_j}q_{j+1} \cdots q_{n+1}$ is to the right of L_i so it is not part of ∂Q_i. It is replaced by $tL_i\infty$ to obtain $\partial Q_i = q_0 \cdots q_j L_{k_j} t L_i \infty$ which can be expressed in the same form as (3.2). If $\theta_{s-1} = 180°$ then if all points of ∂Q_{s-2} (in particular $q_1, q_2, \cdots, q_n$ are to the right of or on L_{s-1}) then Q_{s-1} is degenerate and the algorithm terminates otherwise ∂Q_{s-1} is obtained by modifying ∂Q_{s-2} as above.

Suppose Q_{s-1} is not degenerate. Q_s is either a bounded convex polygon or degenerate by (3.1c) and ∂Q_s is obtained by modifying ∂Q_{s-1}. If all points of ∂Q_{s-1} are to the right of or on L_s then Q_s is degenerate and the algorithm terminates. Otherwise L_s intersects ∂Q_{s-1} at exactly two points, t_1 and t_2, since Q_{s-1} is convex and $\theta_s > 180°$. Suppose ∂Q_{s-1} is in the form of (3.2) with $i = s$ and t_1 occurs before t_2 in the polygonal curve (see Fig. 3.3). Suppose j and l are the indices $(j < l)$ such that t_1 is on line

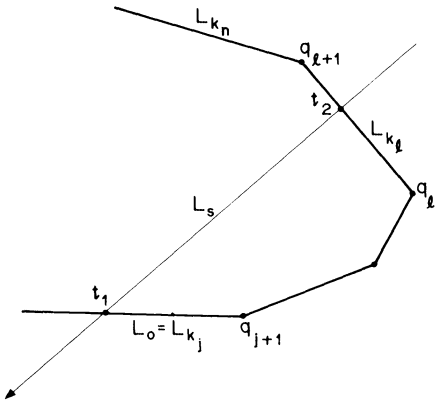


FIG. 3.3. *∂Q_{s-1} and L_s intersect at points t_1 and t_2.*

segment $q_j L_{k_j} q_{j+1}$ and $t_1 \neq q_{j+1}$ and t_2 is on line segment $q_l L_{k_l} q_{l+1}$ and $t_2 \neq q_l$. l can be found by scanning backwards from n for the first q_l to the left of L_s and j can be found by scanning forward from 0 for the first q_{j+1} to the left of L_s. The polygonal curves $t_2 L_{k_l} q_{l+1} \cdots q_{n+1}$ and $q_0 \cdots q_j L_{k_j} t_1$ are to the right of L_s so they are not part of ∂Q_s. They are replaced by $t_2 L_s t_1$ to obtain $\partial Q_s = t_1 L_{k_j} q_{j+1} \cdots q_l L_{k_l} t_2 L_s t_1$ which can be expressed in the same form as (3.3) below.

Suppose $s+1 \leqq i \leqq m-1$ and Q_{i-1} is not degenerate. ∂Q_i is obtained by modifying

$$\partial Q_{i-1} = q_0 L_{k_0} q_1 \cdots q_n L_{k_n} q_{n+1}, \qquad q_0 = q_{n+1}, \qquad 0 \leqq k_0 < k_1 < \cdots < k_n \leqq i-1. \tag{3.3}$$

Consider the position of L_i with respect to Q_{i-1}. There are three cases (see Fig. 3.4):

(a) Q_{i-1} is to the right of or on L_i,

(b) Q_{i-1} is to the left of or on L_i,

(c) ∂Q_{i-1} and L_i intersect at exactly two points, t_1 and t_2.

In case (a), $q_0, q_1, \cdots, q_n$ are to the right of or on L_i, Q_i is degenerate, and the algorithm terminates. In case (b), $Q_i = Q_{i-1}$ and the condition that q_0 is to the left of or on L_i is sufficient to determine this case since $\theta_{k_0} \geqq 0°$ and $\theta_{k_n} < \theta_i$, i.e. the slope of L_i lies between the slopes of L_{k_n} and L_{k_0}. In case (c), ∂Q_i is determined in the same way as ∂Q_s above. If Q_i is not degenerate for $s'+1 \leqq i \leqq m-1$, then the algorithm terminates with $\partial Q = \partial Q_{m-1}$.

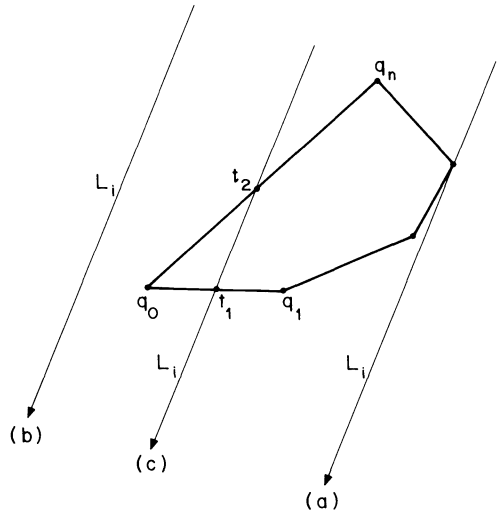


FIG. 3.4. *Three cases of position of L_i with respect to Q_{i-1}.*

The pseudo-code for our algorithm is given in procedure SHRINK.

```
Procedure SHRINK (P, m, r, Q, n);
# Input: convex polygon P = p_0 p_1 ··· p_{m-1} p_m and shrinking distance r
# Output: Q = φ and n = 0 or convex polygon Q = q_0 q_1 ··· q_{n-1} q_n and 3 ≦ n ≦ m
  α := polar angle of p_1 − p_0;
  q_1 := intersection of L_0 and L_1;
  k_0 := 0; k_1 := 1;
  i := 2; n := 1;
  θ := (polar angle of p_3 − p_2) − α;
  Translate angle θ to the interval [0°, 360°);
  while θ ≦ 180° do
    # polygonal boundary curve is ∞ L_{k_0} q_1 ··· q_n L_{k_n} ∞ and next line is L_i
    while n ≧ 1 and q_n is to the right of or on L_i do n := n − 1;
    if θ = 180° and n = 0 then return;
    t := intersection of L_{k_n} and L_i;
    n := n + 1; q_n := t; k_n := i;
    i := i + 1; θ := (polar angle of p_{i+1} − p_i) − α;
    Translate angle θ to the interval [0°, 360°);
  j := 0; q_j := ∞, q_{n+1} := ∞;
  while i ≦ m − 1 do
```

```
    # polygonal boundary curve is q_j L_{k_j} q_{j+1} ··· q_n L_{k_n} q_{n+1} and next line is L_i
    if q_j = ∞ or q_j is to the right of L_i then
      while n ≧ j+1 and q_n is to the right of or on L_i do n := n-1;
      if n = j then n := 0; return;
      t_2 := intersection of L_{k_n} and L_i;
      n := n+1; q_n := t_2; k_n := i;
      while q_{j+1} is to the right of or on L_i do j := j+1;
      t_1 := intersection of L_{k_j} and L_i;
      q_j := t_1; q_{n+1} := t_1;
    i := i+1;
  for i := 0 to n+1-j do q_i := q_{i+j};
  n := n+1-j;
  return;
```

We now derive the time complexity for the algorithm. For each $i \geqq 2$, ∂Q_i is obtained from ∂Q_{i-1} by removing zero or more line segments from the beginning and end of ∂Q_{i-1}. Each line segment is parallel to an edge of ∂P and when it is removed from ∂Q_{i-1} it is not involved in any further computation. Therefore at most m line segments are removed from the ∂Q_i, $1 \leqq i \leqq m-2$. All other computations clearly require $O(m)$ time. Therefore we have shown the following.

THEOREM 3.1. *Procedure* SHRINK *determines* Q *in* $O(m)$ *time.*

4. Triangulation of the interior of P. Let $int(P)$ be obtained by shrinking P by a distance of $h/\sqrt{2}$, i.e.

$$int(P) = \{v \mid v \in P \text{ and distance from } v \text{ to } \partial P \geqq h/\sqrt{2}\}. \tag{4.1}$$

If $h/\sqrt{2}$ is sufficiently small then $int(P)$ is a convex polygon, otherwise $int(P)$ is degenerate, i.e. it is empty, a point, or a line segment. In the degenerate case, no interior triangulation is done, so we assume that $int(P)$ is a convex polygon in this section.

Let v_b and v_t be the endpoints of a diameter of $int(P)$. The algorithm of Shamos (1975) is used to compute the diameter in linear time. We rotate the coordinate system so that line segment $v_b v_t$ is parallel to the y-axis with $y(v_t) > y(v_b)$, where $y(v)$ denotes the y-coordinate of vertex v. We introduce $n+1$ horizontal lines, $y = y_i$, through $int(P)$, evenly spaced h apart with $n = \lfloor (y(v_t) - y(v_b))/h \rfloor$; $dy = (y(v_t) - y(v_b) - nh)/2$; and $y_i = y(v_t) - dy - ih$, $0 \leqq i \leqq n$. Let a_i and b_i be the x-coordinates of the left and right endpoints of $y = y_i$ in $int(P)$ and let $m(i) = \lfloor (b_i - a_i)/h \rfloor$; $dx_i = (b_i - a_i - m(i)h)/2$; and $x_{i,j} = a_i + dx_i + jh$, $0 \leqq j \leqq m(i)$. On each line, we introduce a sequence of mesh vertices $(x_{i,j}, y_i)$ for $0 \leqq j \leqq m(i)$, spaced h apart. Note that at least one vertex is generated on each line and that the vertex which is nearest to ∂P is between $h/\sqrt{2}$ and $h/2 + h/\sqrt{2}$ distance from ∂P (see Fig. 4.1). These vertices do not lie on a uniform square grid of spacing h because those on $y = y_{i+1}$ may be shifted horizontally with respect to those on $y = y_i$ or $y = y_{i+2}$. Consequently we have referred to them as being on a *quasi-uniform* grid.

A subset (possibly empty) of these vertices are then triangulated in a scan down the strips $y_{i+1} \leqq y \leqq y_i$ (see Fig. 4.2). For each pair of lines, let $a = \max(x_{i,0}, x_{i+1,0})$ and $b = \min(x_{i,m(i)}, x_{i+1,m(i+1)})$. (Note that it is possible that $b < a$. In this case $a - b \leqq h$ since on lines $y = y_i$ and $y = y_{i+1}$ there exists a vertex at distance $\leqq h/2$ from diameter $v_b v_t$.) The vertices on the two lines for which the x-coordinate is in the interval $[a-h, b+h]$ are then connected up to form a sequence of similar triangles in which the area is $h^2/2$ and the angles are between 45° and 90° inclusive. This process is carried out for each pair of lines in which $a \leqq b$ and $m(i) + m(i+1) > 0$. If $b < a$ or

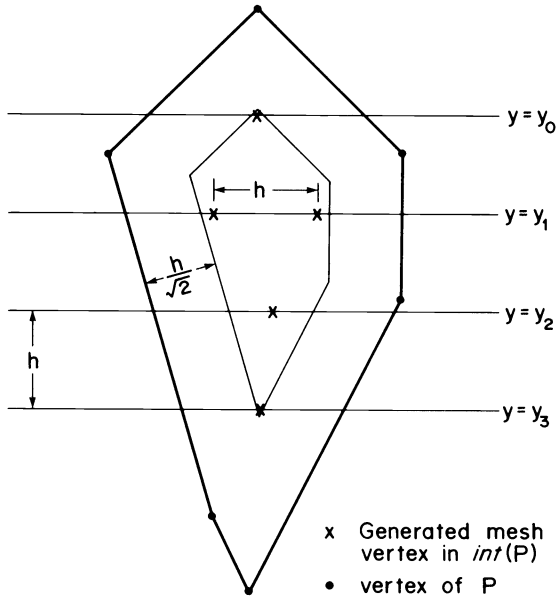


FIG. 4.1. *Generation of mesh vertices in int(P).*

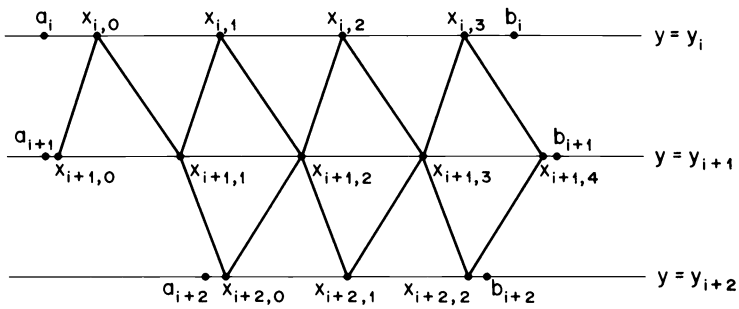


FIG. 4.2. *Triangulation of mesh vertices in int(P).*

$m(i) = m(i+1) = 0$, then no triangles are formed between $y = y_i$ and $y = y_{i+1}$. In this case, the shortest edge joining a vertex on each line is introduced.

We need to identify a boundary of this set of vertices, so that we can triangulate the strip between it and ∂P. Let V_I be the set of vertices, and E_I be the following set of edges:

(4.2)
(a) edges of triangles formed in the scans described above,
(b) additional edges joining consecutive vertices at the ends of lines $y = y_i$,
(c) the shortest edge joining a vertex on line $y = y_i$ to a vertex on line $y = y_{i+1}$, in the case that no triangles are formed between these lines.

Then $G_I = (V_I, E_I)$ is a connected planar graph, and if it contains at least two vertices, then a counterclockwise closed walk, C, can be formed to identify the boundary of G_I by including an edge e of E_I k times in C if e occurs in $2-k$ triangles, $0 \leqq k \leqq 2$. If an edge e occurs twice in C, the direction is opposite in its two occurrences (see Fig. 5.1). If V_I contains only one vertex then let C be this single vertex.

C consists of a walk, C_L, down the left side of G_I followed by the sequence of edges on the line $y = y_n$ from left to right, and a walk, C_R, up the right side of G_I followed by the sequence of edges on the line $y = y_0$ from right to left. C_L consists of paths from $(x_{i,0}, y_i)$ to $(x_{i+1,0}, y_{i+1})$, $i = 0, 1, \cdots, n-1$, which are constructed as follows. The leftmost edge between lines $y = y_i$ and $y = y_{i+1}$ is either part of the triangulation or the edge formed in (4.2c). This edge joins vertices $(x_{i,j}, y_i)$ and $(x_{i+1,k}, y_{i+1})$ with either $j = 0$ or $k = 0$ (or both). If $j = 0$ then the path formed by the vertices $(x_{i,0}, y_i)$, $(x_{i+1,k}, y_{i+1})$, $(x_{i+1,k-1}, y_{i+1}), \cdots, (x_{i+1,0}, y_{i+1})$ is a subwalk of C_L, otherwise the path formed by the vertices $(x_{i,0}, y_i)$, $(x_{i,1}, y_i), \cdots, (x_{i,j}, y_i)$, $(x_{i+1,0}, y_{i+1})$ is a subwalk of C_L. For example, $(x_{i,0}, y_i)$, $(x_{i+1,0}, y_{i+1})$, $(x_{i+1,1}, y_{i+1})$, $(x_{i+2,0}, y_{i+2})$ is a subwalk of C_L in Fig. 4.2. Similarly C_R consists of paths from $(x_{i+1,m(i+1)}, y_{i+1})$ to $(x_{i,m(i)}, y_i)$, $i = n-1$, $n-2, \cdots, 0$. C_L and the reverse of C_R are constructed during the scan of lines to form the triangles of $int(P)$.

The following lemma implies that the edges of E_I (see (4.2)) are locally optimal in any triangulation of the mesh vertices (including those on ∂P) which contains these edges.

LEMMA 4.1. *If $e \in E_I$, then e is a Delaunay edge.*

Proof. E_I can be partitioned into two disjoint sets of edges E_1 and E_2 where E_1 contains the horizontal edges and E_2 contains the edges which join vertices on two consecutive lines $y = y_i$ and $y = y_{i+1}$. First suppose $e \in E_1$. Let the vertices of edge e be $v_1 = (x_{i,j}, y_i)$ and $v_2 = (x_{i,j} + h, y_i)$. Let S be the circle of radius $h/2$ with centre at $(x_{i,j} + h/2, y_i)$ (see Fig. 4.3a). Clearly no vertices of V_I can be in the interior of S. Mesh vertices on ∂P lie a distance $\geqq h/\sqrt{2}$ from the centre of S by (4.1), so they are not in the interior of S. By Lemma 2.1, e is a Delaunay edge.

Now suppose $e \in E_2$. Let the vertices of edge e be $v_1 = (x_{i,j}, y_i)$ and $v_2 = (x_{i+1,k}, y_{i+1})$, and let $d = |x_{i,j} - x_{i+1,k}|$. Let S be the circle whose diameter is e. It is apparent from Fig. 4.3b that no vertices of V_I can be in the interior of S if $d \leqq h$. To establish this, we note that e may have been formed by either (4.2a) or (4.2c). In the first case, it is clear that $d \leqq h$ as a consequence of the triangulation process in $int(P)$ (see Fig. 4.2). In the second case, $d = a - b \leqq h$ as discussed in the paragraph above (4.2). The radius of S is $\sqrt{h^2 + d^2}/2 \leqq h/\sqrt{2}$. As above, mesh vertices on ∂P are not in the interior of S. By Lemma 2.1, e is a Delaunay edge. □

The pseudo-code for the generation of mesh vertices, triangles, and closed walk C in $int(P)$ is given in procedure INTTRIANG. In this procedure, a triangle is represented by a list of three vertex coordinates in counterclockwise order, the function

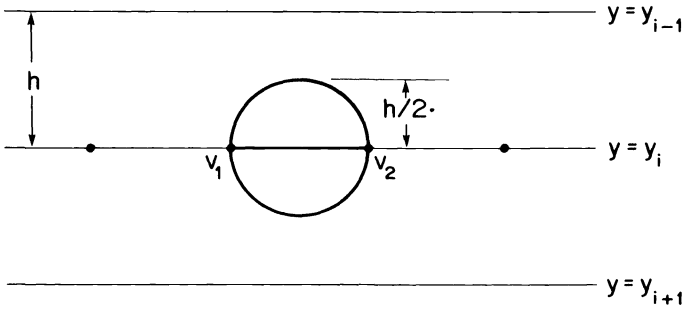


FIG. 4.3a. *Edge v_1v_2 is in E_1.*

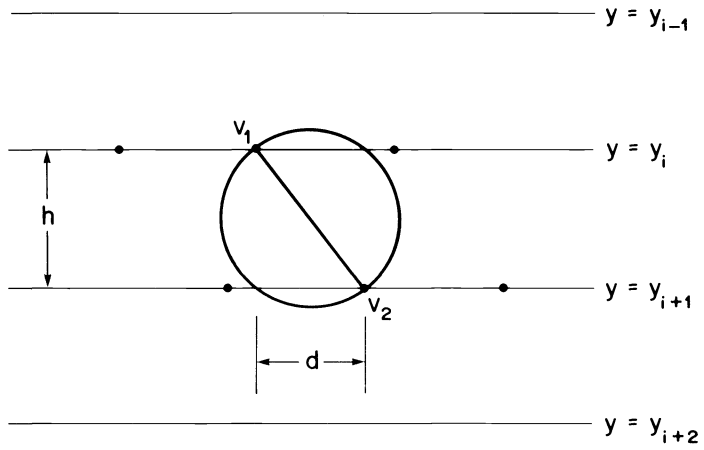


FIG. 4.3b. *Edge v_1v_2 is in E_2.*

append(*list*1, *list*2) appends the elements of the second list at the end of the first list, and the function *reverse*(*list*) reverses the elements of the list.

```
Procedure INTTRIANG (int(P), h, TI, nt, C, nc);
# Input: convex polygon int(P) with diameter parallel to y-axis and triangle size parameter h
# Output: list of triangles TI, number of triangles nt, closed walk C, number of edges nc in C
  # Generate mesh vertices in int(P)
  ymax := maximum y-coordinate of vertices of int(P);
  ymin := minimum y-coordinate of vertices of int(P);
  n := trunc ((ymax - ymin)/h);
  dy := (ymax - ymin - nh)/2;
  for i := 0 to n do y_i := ymax - dy - ih;
  Scan down the left and right sides of ∂int(P) to determine the points (a_i, y_i) and (b_i, y_i);
  for i := 0 to n do
    m(i) := trunc ((b_i - a_i)/h);
    dx_i := (b_i - a_i - m(i)h)/2;
    for j := 0 to m(i) do x_{i,j} := a_i + dx_i + jh;
  nt := 0; TI := [ ];
  if n = 0 then nc := 0; C := [(x_{0,0}, y_0)]; return;
  C_L := [(x_{0,0}, y_0)]; C_R := [(x_{0,0}, y_0), (x_{0,1}, y_0), · · · , (x_{0,m(0)}, y_0)];
  for i := 0 to n - 1 do
    a := max (x_{i,0}, x_{i+1,0});
    b := min (x_{i,m(i)}, x_{i+1,m(i+1)});
    l0 := smallest integer such that x_{i,l0} ≧ a - h;
    l1 := smallest integer such that x_{i+1,l1} ≧ a - h;
    r0 := largest integer such that x_{i,r0} ≦ b + h;
    r1 := largest integer such that x_{i+1,r1} ≦ b + h;
    # Generate the triangles between y = y_i and y = y_{i+1}
    if l0 < r0 or l1 < r1 then
      j = l0; k := l1;
      while j < r0 and k < r1 do
        if x_{i+1,k} ≦ x_{i,j} then
          nt := nt + 1; TI(nt) := Δ[(x_{i+1,k}, y_{i+1}), (x_{i+1,k+1}, y_{i+1}), (x_{i,j}, y_i)];
          k := k + 1;
        else
          nt := nt + 1; TI(nt) := Δ[(x_{i,j+1}, y_i), (x_{i,j}, y_i), (x_{i+1,k}, y_{i+1})];
          j := j + 1;
      if j = r0 then while k < r1 do
          nt := nt + 1; TI(nt) := Δ[(x_{i+1,k}, y_{i+1}), (x_{i+1,k+1}, y_{i+1}), (x_{i,j}, y_i)];
          k := k + 1;
```

```
    else while j < r0 do
        nt := nt+1; TI(nt) := Δ[(x_{i,j+1}, y_i), (x_{i,j}, y_i), (x_{i+1,k}, y_{i+1})];
        j := j+1;
  # Generate the subwalks of C_L and C_R between y = y_i and y = y_{i+1}
  if x_{i,l0} ≦ x_{i+1,l1} then
    C_L := append (C_L, [(x_{i,1}, y_i), (x_{i,2}, y_i), ···, (x_{i,l0}, y_i), (x_{i+1,0}, y_{i+1})])
  else C_L := append (C_L, [(x_{i+1,l1}, y_{i+1}), (x_{i+1,l1-1}, y_{i+1}), ···, (x_{i+1,0}, y_{i+1})]);
  if x_{i,r0} ≦ x_{i+1,r1} then
    C_R := append (C_R, [(x_{i+1,r1}, y_{i+1}), (x_{i+1,r1+1}, y_{i+1}), ···, (x_{i+1,m(i+1)}, y_{i+1})])
  else C_R := append (C_R, [(x_{i,m(i)-1}, y_i), (x_{i,m(i)-2}, y_i), ···, (x_{i,r0}, y_i), (x_{i+1,m(i+1)}, y_{i+1})]);
C_L := append (C_L, [(x_{n,1}, y_n), (x_{n,2}, y_n), ···, (x_{n,m(n)-1}, y_n)]);
C := append (C_L, reverse (C_R));
nc := number of edges in C;
return;
```

5. Triangulation near the boundary of P. Mesh vertices are generated on the edges of ∂P as follows. Let e_i be an edge of ∂P with triangle size parameter h_i (see § 1). Mesh vertices are generated on e_i at an equal spacing of $\bar{h}_i$, the nearest length to h_i which is an integral submultiple of $|e_i|$ (the length of e_i). $\bar{h}_i$ is computed as follows:

$$
\begin{aligned}
&k := \text{trunc}\,(|e_i|/h_i);\\
&r := |e_i|/h_i - k;\\
&\text{if } r > k/(2k+1) \text{ then } k := k+1;\\
&\bar{h}_i := |e_i|/k;
\end{aligned}
\tag{5.1}
$$

If h_i satisfies (1.1), then the restriction of $\bar{h}_i$ compared to h is

$$
\min\,(|e_i|, \sqrt{2}h/3) \leqq \bar{h}_i \leqq 4\sqrt{2}h/3 \tag{5.2}
$$

from (5.1).

In this section we describe a procedure for triangulating the strip, A, between ∂P and C in the case of at least one vertex in the interior of P. (The case in which $int(P)$ is degenerate and there are no vertices in the interior of P will be discussed later in this section.) We believe that the spacing of mesh vertices on ∂P restricted by (5.2) is small enough for the following procedure to generate a valid triangulation in A. However, we have been unable to prove this. On the other hand, as Fig. 5.4 below shows, if the spacing of vertices on ∂P is too large relative to h then an invalid triangulation can be formed. We show how the procedure can be modified to produce a valid triangulation in this case. In § 7 we describe how to modify the triangulation of A so that the total triangulation of P is a Delaunay triangulation.

If there are at least two vertices in $int(P)$, then let the closed walk C determined in the preceding section be represented by the list of vertices $C = [v_0, v_1, \cdots, v_{nc}]$ where $v_0 = v_{nc} = (x_{0,0}, y_0)$ and $nc \geqq 2$ is the number of edges $v_j v_{j+1}$ in C; otherwise let $C = [v_0]$ and $nc = 0$. Let the counterclockwise cycle of edges on ∂P be represented by the list of mesh vertice $B = [u_0, u_1, \cdots, u_{nb}]$ where $u_0 = u_{nb}$, $nb \geqq 3$ is the number of edges $u_i u_{i+1}$ in B, and the numbering of the u_i is done as follows (see Fig. 5.1). If there are at least two vertices in $int(P)$ then we number the u_i so that u_0 is closest to v_0 among the vertices on ∂P with y-coordinate greater than y_0 (note that edge $u_0 v_0$ is entirely in A). Otherwise we number the u_i so that u_0 is the vertex on ∂P which is closest to v_0. In the former case the selection of u_0 is a heuristic for finding a Delaunay edge $u_0 v_0$. In the latter case,

LEMMA 5.1. *If there is only one vertex v_0 in $int(P)$ and u_0 is the vertex on ∂P closest to v_0 then $u_0 v_0$ is a Delaunay edge.*

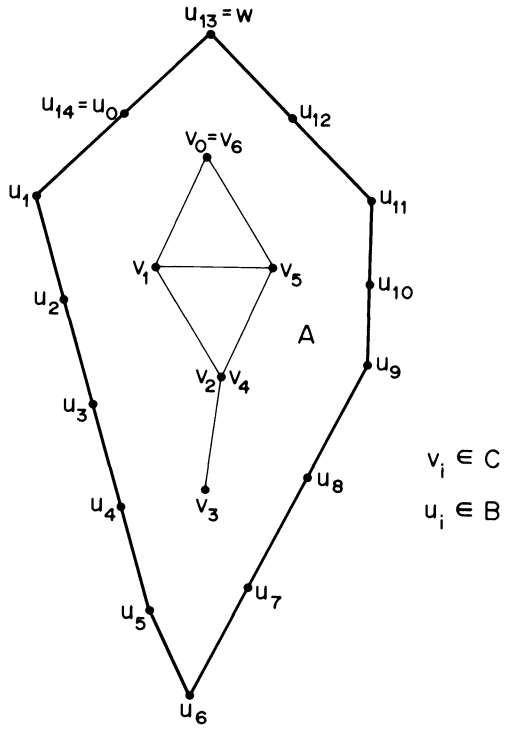


FIG. 5.1. *Illustration of closed walk C, cycle B, and strip A.*

Proof. Let S be the circle of radius $|u_0v_0|$ centred at v_0. S contains no vertices in its interior since u_0 is the closest vertex to v_0. Let S' be the circle of radius $|u_0v_0|/2$ whose diameter is u_0v_0. S' is contained inside S, therefore S' contains no vertices in its interior. By Lemma 2.1, u_0v_0 is a Delaunay edge. □

We now describe how to generate edges of the type u_iv_j and triangles of the types $\Delta u_iu_{i+1}v_j$ and $\Delta v_{j+1}v_ju_i$ in the strip A by "merging" B and C and "zigzagging" counterclockwise around A starting and ending at edge u_0v_0. Note that A is to the left of B and to the right of C so that for a valid triangulation of A (see § 2), CCW triangles $\Delta u_iu_{i+1}v_j$ and $\Delta v_{j+1}v_ju_i$ must be generated. Suppose u_iv_j is the last edge generated and $i < nb$ or $j < nc$ (initially $i = 0$ and $j = 0$). The direction of edge u_iv_j is from u_i to v_j in the last triangle and its direction in the next triangle will be from v_j to u_i. If $i = nb$ then the next edge and triangle generated are $u_{nb}v_{j+1}$ and $\Delta v_{j+1}v_ju_{nb}$. If $j = nc$ then the next edge and triangle generated are $u_{i+1}v_{nc}$ and $\Delta u_iu_{i+1}v_{nc}$. Otherwise either edge u_iv_{j+1} and triangle $\Delta v_{j+1}v_ju_i$ or edge $u_{i+1}v_j$ and triangle $\Delta u_iu_{i+1}v_j$ are generated next based on the following test.

Consider the quadrilateral $Q = u_iu_{i+1}v_{j+1}v_j$. v_j and v_{j+1} are to the left of the directed line from u_i to u_{i+1}, but the positions of u_i and u_{i+1} relative to the directed line, $l(v_jv_{j+1})$, from v_j to v_{j+1} may be one of the following four cases (see Fig. 5.2):

(5.3)
(a) u_i and u_{i+1} are both to the right of $l(v_jv_{j+1})$,
(b) $u_i(u_{i+1})$ is to the left of or on (right of) $l(v_jv_{j+1})$,
(c) $u_i(u_{i+1})$ is to the right of (left of or on) $l(v_jv_{j+1})$,
(d) u_i and u_{i+1} are both to the left of or on $l(v_jv_{j+1})$.

In case (a), Q is a strictly convex quadrilateral and if the circle through the vertices

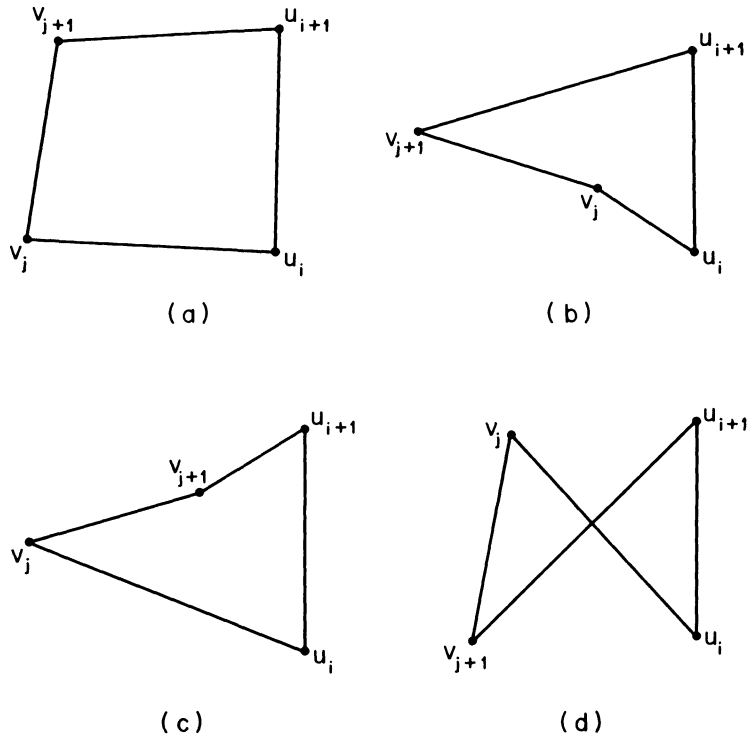


FIG. 5.2. *Four cases of quadrilateral Q.*

u_i, u_{i+1}, v_j does not contain vertex v_{j+1} in its interior then edge $u_{i+1}v_j$ and triangle $\Delta u_i u_{i+1} v_j$ are chosen else edge $u_i v_{j+1}$ and triangle $\Delta v_{j+1} v_j u_i$ are chosen. In case (b), Q is a nonconvex quadrilateral and edge $u_{i+1}v_j$ and triangle $\Delta u_i u_{i+1} v_j$ are chosen. In case (c), Q is a nonconvex quadrilateral and edge $u_i v_{j+1}$ and triangle $\Delta v_{j+1} v_j u_i$ are chosen. In case (d), Q is a nonsimple quadrilateral and edge $u_{i+1}v_j$ and triangle $\Delta u_i u_{i+1} v_j$ are chosen; the other triangle $\Delta v_{j+1} v_j u_i$ is oriented in the wrong (clockwise) direction (see (2.1a)). In cases (a), (b), and (c), the chosen edge is locally optimal in Q (see § 2).

Let $T(P)$ be the collection of triangles formed in A and $int(P)$ as described above and in § 4. $T(P)$ is illustrated in Fig. 5.3. It is not clear that the triangles in $T(P)$ do not overlap, i.e. that they produce a valid triangulation. If $|u_i u_{i+1}|$ is too large relative to h for some i, then $T(P)$ can be an invalid triangulation (e.g. see Fig. 5.4). After discussing the case of no interior vertices, we indicate how this merge procedure can be modified to generate a valid triangulation, $VT(P)$, even if the $|u_i u_{i+1}|$ are large relative to h.

Now we consider the case of no vertices in the interior of P. Let v_b and v_t be the endpoints of a diameter of P and suppose the coordinate system is rotated so that line segment $v_b v_t$ is parallel to the y-axis with $y(v_t) > y(v_b)$ (as was done for $int(P)$ in § 4). Let $B = [u_0, u_1, \cdots, u_{mb}, \cdots, u_{nb-1}, u_{nb}]$ be the counterclockwise cycle of vertices on ∂P where $u_0 = u_{nb} = v_t$ and $u_{mb} = v_b$. B can be split into two chains $B_L = [u_0, u_1, \cdots, u_{mb}]$ and $B_R = [u_{nb}, u_{nb-1}, \cdots, u_{mb}] = [v_0, v_1, \cdots, v_{mc}]$ on the left and right sides of ∂P, respectively, such that the y-coordinates of the vertices strictly decrease from $y(v_t)$ to $y(v_b)$. Note that mb and mc are both at least one. B_L and B_R can be merged to produce a valid triangulation, $VT(P)$, in P in a way similar to the procedure described above

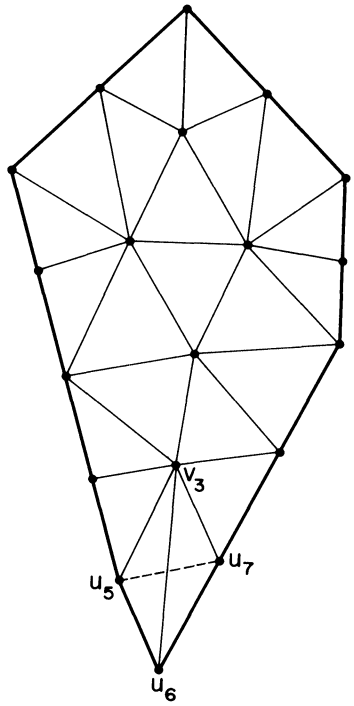


FIG. 5.3. *Illustration of* $T(P)$ *and* $DT(P)$; u_6v_3 *is in* $T(P)$; u_5u_7 *is in* $DT(P)$.

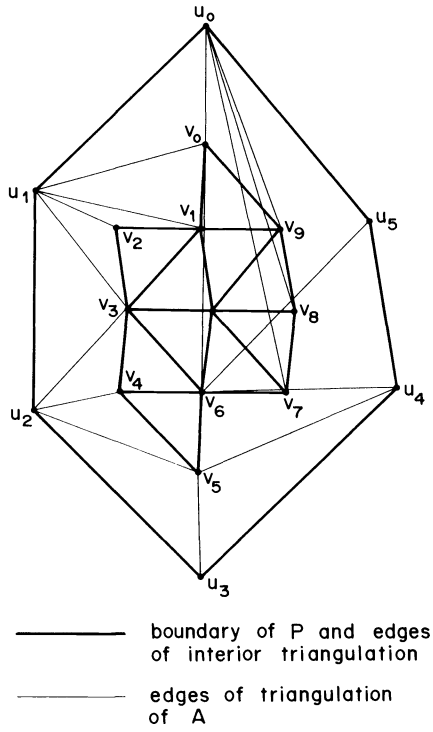


FIG. 5.4. *Illustration of invalid triangulation* $T(P)$; *note overlap by triangles* $\Delta v_6v_5u_4$ *and* $\Delta v_9v_8u_0$.

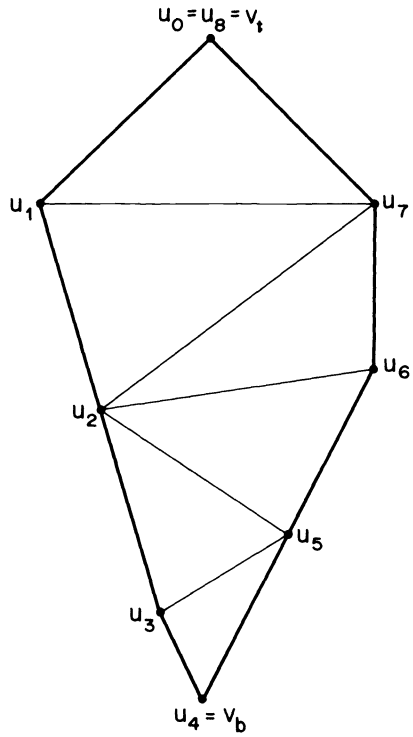


FIG. 5.5. *Illustration of chains B_L, B_R and $VT(P)$ in the case of no interior vertices.*

(see Fig. 5.5), and $VT(P)$ can be converted into a Delaunay triangulation in a way similar to that described in § 7. The validity of $VT(P)$ is discussed in § 6.

The pseudo-code for the merge of B and C or B_L and B_R to generate triangles in the strip A or polygon P, respectively, is given in procedure MERGE. The list of triangles, *TM*, produced by this procedure is modified by the procedure of § 7 and then, in the case of at least one vertex in the interior of P, appended to the list of triangles, *TI*, produced by procedure INTTRIANG.

```
Procedure MERGE (inter, L, nl, M, nm, TM, EM);
# Input: if inter = true then L = B, nl = nb, M = C, nm = nc
#        else L = B_L, nl = mb, M = B_R, nm = mc
# Output: list of triangles TM and list of edges EM;
#         if inter = true then TM and EM each contain nb + nc elements
#           else TM and EM each contain mb + mc - 2 elements
  nll := nl; nmm := nm;
  nt := 0;
  if not inter then
    nt := nt + 1;
    TM(nt) := Δu_0 u_1 v_1; EM(nt) := u_1 v_1;
    if nl + nm = 3 then return;
    nl := nl - 1; nm := nm - 1;
    i := 1; j := 1;
  else i := 0 j := 0;
  while i < nl and j < nm do
    if (u_i and u_{i+1} are both to the left of or on l(v_j v_{j+1})) or
       (v_{j+1} is not in the circle through u_i, u_{i+1}, v_j) then
       nt := nt + 1;
       TM(nt) := Δu_i u_{i+1} v_j; EM(nt) := u_{i+1} v_j;
       i := i + 1;
```

```
    else
      nt := nt + 1;
      TM(nt) := Δv_{j+1}v_j u_i; RM(nt) := u_i v_{j+1};
      j := j + 1;
  if i < nl then
    if not inter and j = nm then nl := nl + 1;
    while i < nl do
      nt := nt + 1;
      TM(nt) := Δu_i u_{i+1} v_j; EM(nt) := u_{i+1} v_j;
      i := i + 1;
  else # j < nm
    if not inter and i = nl then nm := nm + 1;
    while j < nm do
      nt := nt + 1;
      TM(nt) := Δv_{j+1}v_j u_i; EM(nt) := u_i v_{j+1};
      j := j + 1;
  nl := nll; nm := nmm;
  return;
```

As mentioned above, procedure MERGE can fail to produce a valid triangulation of P in the case of at least one interior vertex if the spacing of the u_i on ∂P is too large relative to h. In § 6, we examine procedure MERGE to see how it can be modified to always produce a valid triangulation. We will show that an invalid triangulation $T(P)$ is due to an overlapping triangle $\Delta v_j v_{j-1} u_i$ in which v_{j+1} is to the right of $l(v_{j-1}v_j)$ and u_i is to the left of or on $l(v_j v_{j+1})$. Intuitively, we can think of this as resulting from there being too few vertices in B for the number of vertices in C. The modification consists of generating CCW triangle $\Delta v_{j+1} v_j v_{j-1}$ instead of $\Delta v_j v_{j-1} u_i$, replacing subwalk $v_{j-1}v_j v_{j+1}$ of C with edge $v_{j-1}v_{j+1}$, and restarting the merge of B and the new shortened C from edge $u_r v_{j-1}$ where $\Delta v_{j-1} v_{j-2} u_r$ is formed by the merge procedure. The pseudocode for the modification is given procedure MMERGE. Let $VT(P)$ be the triangulation produced by procedures INTTRIANG and MMERGE. Procedure MMERGE and the validity of $VT(P)$ will be discussed in § 6. In procedure DELTRIANG of § 8, procedure MMERGE is used to produce a valid triangulation of A in the case of at least one interior vertex, and procedure MERGE is used to produce a valid triangulation of P in the case of no interior vertices.

```
Procedure MMERGE (B, nb, C, nc, TM, EM, ne);
# Input: boundary cycle B, closed walk C, number of edges nb and nc in B and C
# Output: list of triangles TM and edges EM in A, each containing nb + nc elements,
#       and the number of edges ne of type v_j v_{j+m}, m ≧ 2, generated in A.
#       The edges of type v_j v_{j+m} are stored at the end of EM in the reverse order
#       that they are generated. The edges of type u_i v_j are stored at the front of EM
#       in the order that they are generated. The triangles are stored in a similar way.
# The working arrays s, p, r, and n are used as follows:
#       v_{s(j)} and v_{p(j)} are the successor and predecessor vertices of v_j in C.
#       s and p are updated when a vertex is removed from C.
#       u_{r(j)} is the vertex of B in the triangle Δv_{s(j)} v_j u_{r(j)} and
#       n(j) is the index of this triangle in list TM.
#       r and n are used to determine how far to backtrack the merge of
#       B and C when a vertex is removed from C.
  for j := 0 to nc - 1 do
    s(j) := j + 1; p(j + 1) := j; r(j) := 0; n(j) := 0;
  nt := 0; ne := 0;
  i := 0; j := 0;
  while i ≦ nb and j ≦ nc and i + j < nb + nc do
    if (j = nc) or (u_i and u_{i+1} are both to the left of or on l(v_j v_{s(j)}))
      or (v_{s(j)} is not in the circle through u_i, u_{i+1}, v_j) then
```

```
    nt := nt + 1;
    TM(nt) := Δu_i u_{i+1} v_j; EM(nt) := u_{i+1} v_j;
    i := i + 1;
  else
    if (s(j) = nc) or (v_{s(s(j))} is to the left of or on l(v_j v_{s(j)}))
      or (u_i is to the right of or on l(v_{s(j)} v_{s(s(j))})) then
      nt := nt + 1;
      TM(nt) := Δv_{s(j)} v_j u_i; EM(nt) := u_i v_{s(j)};
      r(j) := i; n(j) := nt;
      j := s(j);
    else
      flag := false;
      repeat # remove v_{s(j)} from C
        TM(nb + nc - ne) := Δv_{s(s(j))} v_{s(j)} v_j;
        EM(nb + nc - ne) := v_j v_{s(s(j))};
        ne := ne + 1;
        s(j) := s(s(j)); p(s(j)) := j;
        if j = 0 then i := 0; nt := 0;
          else i := r(p(j)); nt := n(p(j));
        if (j > 0) and (v_{s(j)} is to the right of l(v_{p(j)} v_j))
          and (u_i is to the left of or on l(v_j v_{s(j)})) then
          j := p(j)
        else flag := true;
      until flag;
return;
```

6. Validity of triangulation *VT(P)*. $VT(P)$ is a valid triangulation of P if the triangles form a "tiling" of P without overlaps or gaps. We will show that $VT(P)$ is valid for the three cases of (i) no interior vertices, (ii) one interior vertex, and (iii) two or more interior vertices. In case (ii), $VT(P)$ (or $T(P)$) is valid since the triangles $\Delta u_i u_{i+1} v_0$, $0 \leqq i \leqq nb-1$, are formed and they clearly tile P without overlaps or gaps.

In case (i), each edge added to $VT(P)$ in procedure MERGE subdivides an untriangulated convex subregion of P into two convex subregions—a triangle and a smaller untriangulated subregion. For example, in Fig. 5.5, edge u_1u_7 subdivides P into triangle $\Delta u_0u_1u_7$ and convex subpolygon $u_1u_2u_3u_4u_5u_6u_7u_1$. If all the untriangulated convex subregions are nondegenerate (i.e. not a line segment) then the triangles of $VT(P)$ clearly tile P without overlaps or gaps so $VT(P)$ is valid. An untriangulated convex subregion can be degenerate if one side of ∂P contains only two vertices and the other side contains three or more collinear vertices at the bottom. In this case degenerate triangles (i.e. with three collinear vertices) are formed. For example, in Fig. 6.1a, triangles $\Delta u_0u_1u_3$ and $\Delta u_2u_3u_1$ are formed by procedure MERGE and the latter triangle is degenerate. If the collinear vertices at the bottom are perturbed slightly so that no three vertices are collinear and the polygon remains convex then all the untriangulated subregions are nondegenerate and the triangulation is valid (e.g. see Fig. 6.1b). Therefore we still consider the triangulations with the degenerate triangles to be "valid". The procedure described in the next section will convert these triangulations into Delaunay triangulations.

In the rest of this section we will consider the case of two or more interior vertices. We will show that $VT(P)$ is valid by showing how $T(P)$ (the triangulation produced by procedures INTTRIANG and MERGE) can sometimes be invalid and how a modification can be made to procedure MERGE to always produce a valid triangulation. In § 2, we mentioned that a triangulation of a region is valid if it satisfies the four conditions of (2.1).

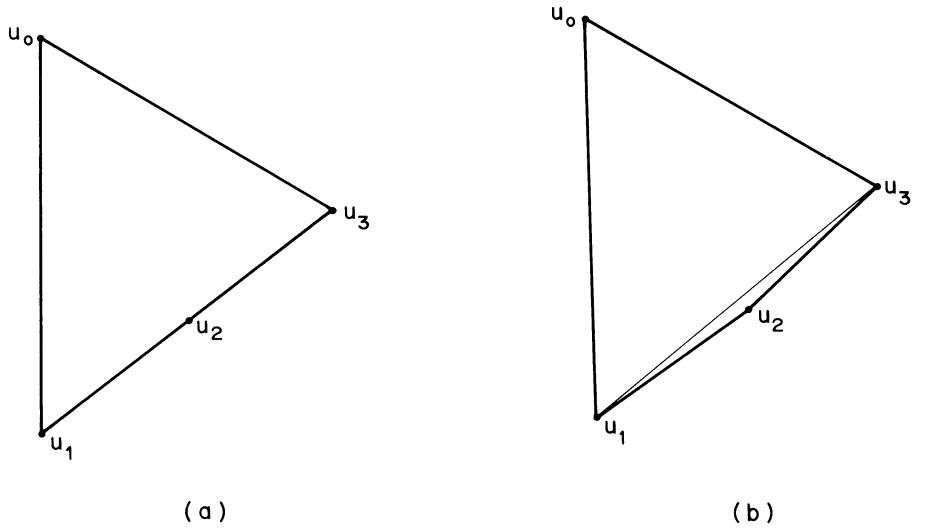


FIG. 6.1. (a) *Example where degenerate triangle* $\Delta u_2u_3u_1$ *is formed by procedure* MERGE; (b) *Perturbation of chain* $u_1u_2u_3$ *to obtain valid triangulation.*

LEMMA 6.1.

(a) $T(P)$ *satisfies conditions* (b), (c), *and* (d) *of* (2.1).

(b) *If* $\Delta v_{j+1}v_ju_i$ *is produced by procedure* MERGE *and* $i < nb$, *then* $\Delta v_{j+1}v_ju_i$ *is* CCW.

(c) *If all triangles of type* $\Delta v_{j+1}v_ju_i$ *produced by procedure* MERGE *are* CCW, *then* $T(P)$ *is a valid triangulation of* P.

(d) $T(P)$ *is not a valid triangulation of* P *if and only if there is a* CW *triangle of type* $\Delta v_{j+1}v_ju_{nb}$ *produced by procedure* MERGE.

Proof. The boundary edges of P are the edges u_iu_{i+1} of B. Conditions (c) and (d) of (2.1) are clearly satisfied by $T(P)$ since the triangles formed in A by procedure MERGE are of the types $\Delta u_iu_{i+1}v_j$ and $\Delta v_{j+1}v_ju_i$. We now show that condition (2.1b) is satisfied by all edges of $T(P)$. The edges of E_I (see (4.2)) which are not edges of C clearly satisfy condition (2.1b) (see Fig. 4.2). An edge $e = v_jv_{j+1}$ of C occurs either once or twice in C. In the latter case, $v_j = v_{k+1}$ and $v_{j+1} = v_k$ for some $k \neq j$ and e occurs in opposite directions in the triangles $\Delta v_{j+1}v_ju_i$ and $\Delta v_{k+1}v_ku_l$ formed in A. In the former case, e occurs in opposite directions in the triangles $\Delta v_{j+1}v_ju_i$ in A and $\Delta v_jv_{j+1}w$ in $int(P)$. A boundary edge u_iu_{i+1} of B occurs only once in the triangle $\Delta u_iu_{i+1}v_j$ formed in A. The other edges of $T(P)$ are of the type u_iv_j, $u_i \in B$ and $v_j \in C$, formed in A. One of the triangles with edge u_iv_j is either $\Delta u_{i-1}u_iv_j$ or $\Delta v_jv_{j-1}u_i$; the other triangle with edge u_iv_j is either $\Delta u_iu_{i+1}v_j$ or $\Delta v_{j+1}v_ju_i$, where the indices of u_i and v_j are taken modulo nb and nc respectively. In all four possibilities, edge u_iv_j occurs in opposite directions in the two triangles. Therefore condition (2.1b) is satisfied by all edges of $T(P)$.

Now we determine when condition (2.1a) is satisfied or not satisfied. The triangles generated by procedure INTTRIANG are all CCW. The triangles generated by procedure MERGE are of the types $\Delta u_iu_{i+1}v_j$ or $\Delta v_{j+1}v_ju_i$. $\Delta u_iu_{i+1}v_j$ is a CCW triangle since v_j is to the left of directed edge u_iu_{i+1}. $\Delta v_{j+1}v_ju_i$ is a CCW triangle if it is formed from case (a) or (c) of (5.3) (see Fig. 5.2). It cannot be formed from case (b) or (d) of (5.3). So $\Delta v_{j+1}v_ju_i$ may be a CW triangle only when there are no more quadrilaterals $u_iu_{i+1}v_{j+1}v_j$ and $i = nb$. By condition (2.1a), $T(P)$ is a valid triangulation if the triangles of type $\Delta v_{j+1}v_ju_{nb}$ are all CCW.

If $T(P)$ is not a valid triangulation then condition (2.1a) is not satisfied and there is a CW triangle of the type $\Delta v_{j+1}v_ju_{nb}$. To show that the converse is also true, suppose

$\Delta v_{j+1}v_ju_{nb}$ is a CW triangle produced by procedure MERGE. Edge v_jv_{j+1} occurs either once or twice in C. In the former case, CCW triangle $\Delta v_jv_{j+1}w$ is formed by procedure INTTRIANG, and $\Delta v_{j+1}v_ju_{nb}$ and $\Delta v_jv_{j+1}w$ overlap. In the latter case, $v_j = v_{k+1}$ and $v_{j+1} = v_k$ for some $k \neq j$ and $\Delta v_{k+1}v_ku_l$ is formed by procedure MERGE; either $\Delta v_{k+1}v_ku_l$ is CCW if $l < nb$ or $\Delta v_{k+1}v_ku_l \equiv \Delta v_{j+1}v_ju_{nb}$ if $l = nb$; in both subcases $\Delta v_{j+1}v_ju_{nb}$ and $\Delta v_{k+1}v_ku_l$ overlap. Therefore $T(P)$ is not a valid triangulation. □

We now examine the triangles produced in A by procedure MERGE to see how CW triangles $\Delta v_{j+1}v_ju_{nb}$ can be formed and how the procedure can be modified to produce a valid triangulation in this case. Since the triangles of type $\Delta u_iu_{i+1}v_j$ are always CCW, we concentrate on the triangles of type $\Delta v_{j+1}v_ju_i$. Let $\Delta v_{j+1}v_ju_{r(j)}$, $0 \leqq j \leqq nc-1$, be the triangles of type $\Delta v_{j+1}v_ju_i$ produced by procedure MERGE, where $0 \leqq r(0) \leqq r(1) \leqq \cdots \leqq r(nc-1) \leqq nb$.

We first make some definitions which will be used in the following lemmas. We define v_j to be a reflex vertex of C if v_{j+1} is to the right of $l(v_{j-1}v_j)$ and define v_j to be a convex vertex of C otherwise. Note that v_0 is a convex vertex of C since it is the leftmost interior mesh vertex on the top line $y = y_0$. Let v_{mid} be the rightmost interior mesh vertex on the bottom line $y = y_n$. v_{mid} is also a convex of C. Let C_L be the subwalk of C from v_0 to v_{mid} and C_R be the subwalk of C from v_{mid} to v_{nc}. Let $[w, z]$ denote the "interval" of ∂P going counterclockwise from point w to point z inclusive. A parenthesis will be used in place of the bracket if the endpoint w or z is not included, e.g. (w, z), $(w, z]$, $[w, z)$. Let w_j and z_j be the intersections of line $l(v_jv_{j+1})$ with ∂P where w_j is the intersection closer to v_j and z_j is the intersection closer to v_{j+1}. (w_j, z_j) is the interval of ∂P to the right of $l(v_jv_{j+1})$. Define $I_j = (w_j, z_j)$ if u_0 is not in (w_j, z_j), $I_j = [u_0, z_j)$ if u_0 is in (w_j, z_j) and $v_jv_{j+1} \in C_L$, and $I_j = (w_j, u_{nb}]$ if $u_0 = u_{nb}$ is in (w_j, z_j) and $v_jv_{j+1} \in C_R$. Recall that $u_{r(j)}$ is a a vertex of the triangle $\Delta v_{j+1}v_ju_{r(j)}$ produced by procedure MERGE.

LEMMA 6.2. *If $u_{r(j)}$ is not in I_j for some j, then $T(P)$ is not a valid triangulation.*

Proof. If $u_{r(j)}$ is not in (w_j, z_j) then $\Delta v_{j+1}v_ju_{r(j)}$ is CW so $r(j) = nb$ and $T(P)$ is not valid by Lemma 6.1. Suppose u_0 is in (w_j, z_j) so that $I_j \neq (w_j, z_j)$ and suppose $u_{r(j)}$ is in (w_j, z_j) but not in I_j. First suppose $v_jv_{j+1} \in C_L$. Then $u_{r(j)}$ is in (w_j, u_{nb}) and CCW triangle $\Delta v_{j+1}v_ju_{r(j)}$ must intersect the walk $u_0v_0v_1 \cdots v_j$ (see Fig. 6.2). Now suppose $v_jv_{j+1} \in C_R$. Then $u_{r(j)}$ is in (u_0, z_j) and CCW triangle $\Delta v_{j+1}v_ju_{r(j)}$ must intersect the walk $v_{j+1}v_{j+2} \cdots v_{nc}u_{nb}$. In both cases, $\Delta v_{j+1}v_ju_{r(j)}$ overlaps other triangles of $T(P)$ so $T(P)$ is not valid. □

LEMMA 6.3. *If v_j is a reflex vertex of C and $\Delta v_jv_{j-1}u_{r(j-1)}$ is* CCW *with $u_{r(j-1)}$ to the left of or on $l(v_jv_{j+1})$, then $\Delta v_jv_{j-1}u_{r(j-1)}$ is an overlapping triangle and $T(P)$ is not a valid triangulation.*

Proof. If $u_{r(j-1)}$) is to the left of or on $l(v_jv_{j+1})$ then part of edge v_jv_{j+1} lies in the interior or on the boundary of $\Delta v_jv_{j-1}u_{r(j-1)}$ (see Fig. 6.3). Therefore $\Delta v_jv_{j-1}u_{r(j-1)}$ and $\Delta v_{j+1}v_ju_{r(j)}$ overlap and $T(P)$ is not a valid triangulation. □

LEMMA 6.4.

(a) *If v_j is a reflex vertex of C and $\Delta v_jv_{j-1}u_{r(j-1)}$ is CCW with $u_{r(j-1)}$ in I_{j-1} and to the right of $l(v_jv_{j+1})$ then $\Delta v_{j+1}v_ju_{r(j)}$ is CCW with $u_{r(j)}$ in I_j.*

(b) *If v_j is a convex vertex of C, $0 < j < nc$, and $\Delta v_jv_{j-1}u_{r(j-1)}$ is CCW with $u_{r(j-1)}$ in I_{j-1} then $\Delta v_{j+1}v_ju_{r(j)}$ is CCW with $u_{r(j)}$ in I_j.*

(c) *If $j = 0$ then $\Delta v_{j+1}v_ju_{r(j)}$ is CCW with $u_{r(j)}$ in I_j.*

Proof. Define $u_{t(j)}$ to be the vertex of largest index in I_j.

(a) Let u_l, $r(j-1) \leqq l \leqq nb$, be the vertex of largest index to the right of $l(v_jv_{j+1})$ such that u_i, $i = r(j-1), \cdots, l$, are to the right of $l(v_jv_{j+1})$. If $l > r(j-1)$ then quadrilaterals $u_iu_{i+1}v_{j+1}v_j$, $i = r(j-1), \cdots, l-1$, are case (a) of (5.3) and CCW triangle

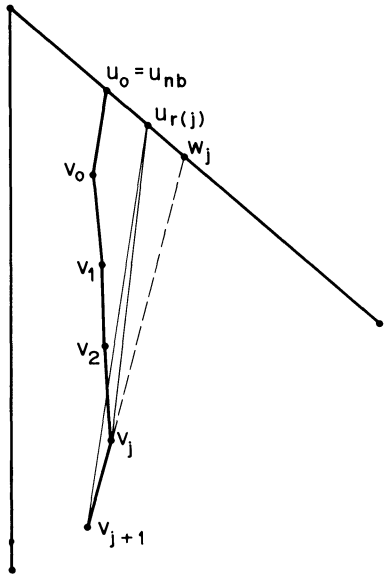


FIG. 6.2. $\Delta v_{j+1}v_ju_{r(j)}$ *intersects walk* $u_0v_0v_1\cdots v_j$.

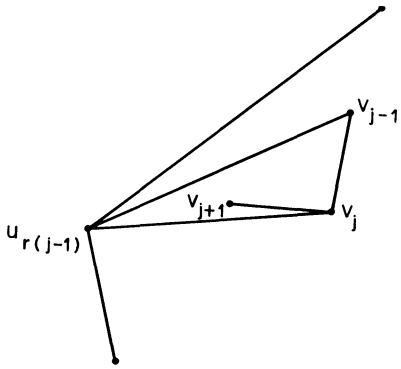


FIG. 6.3. $\Delta v_jv_{j-1}u_{r(j-1)}$ *intersects edge* v_jv_{j+1}.

$\Delta v_{j+1}v_ju_{r(j)}$ may be formed from one of these quadrilaterals. If not, then CCW triangle $\Delta v_{j+1}v_ju_{r(j)}$, $r(j)=l$, is formed because either $l=nb$ or quadrilateral $u_lu_{l+1}v_{j+1}v_j$ is case (c) of (5.3). v_j is a reflex vertex and $u_{r(j-1)}$ is in I_{j-1} and to the right of $l(v_jv_{j+1})$ imply that $u_{r(j-1)}$ is in I_j and $r(j-1)\leqq r(j)\leqq t(j)=l$. Therefore $u_{r(j)}$ is in I_j.

(b) v_j is a convex vertex and u_0 is chosen so that $y(u_0)>y(v_0)$ imply that $t(j)\geqq t(j-1)$. $u_{r(j-1)}$ is in I_{j-1} implies that $r(j-1)\leqq t(j-1)$. Let u_k, $r(j-1)\leqq k\leqq t(j)$, be the vertex of smallest index to the right of $l(v_jv_{j+1})$ such that u_i, $i=k,\cdots,t(j)$, are to the right of $l(v_jv_{j+1})$. If $k>r(j-1)$ then $\Delta u_iu_{i+1}v_j$, $i=r(j-1),\cdots,k-1$ are formed from case (b) or (d) of (5.3). If $k<t(j)$ then quadrilaterals $u_iu_{i+1}v_{j+1}v_j$, $i=k,\cdots,t(j)-1$, are case (a) of (5.3) and CCW triangle $\Delta v_{j+1}v_ju_{r(j)}$ may be formed from one of these quadrilaterals. If not, then CCW triangle $\Delta v_{j+1}v_ju_{r(j)}$, $r(j)=t(j)$, is formed because either $t(j)=nb$ or quadrilateral $u_{t(j)}u_{t(j)+1}v_{j+1}v_j$ is case (c) of (5.3). $u_{r(j-1)}$ is in I_{j-1} and $k\leqq r(j)\leqq t(j)$ imply that $u_{r(j)}$ is in I_j.

(c) Let u_k, $0 \leqq k \leqq t(j)$, be the vertex of smallest index to the right of $l(v_jv_{j+1})$ such that u_i, $i = k, \cdots, t(j)$, are to the right of $l(v_jv_{j+1})$. The rest of the proof is the same as (b) with $r(j-1)$ replaced by 0 (also omit "$u_{r(j-1)}$ is in I_{j-1}" from the last sentence). □

LEMMA 6.5. *If all vertices v_j of C are convex then $T(P)$ is a valid triangulation.*

Proof. By parts (c) and (b) of Lemma 6.4 and induction, $\Delta v_{j+1}v_ju_{r(j)}$, $j = 0, 1, \cdots, nc-1$, are CCW triangles. By Lemma 6.1, $T(P)$ is a valid triangulation. □

LEMMA 6.6. *If for all reflex vertices v_j of C, $u_{r(j-1)}$ is to the right of $l(v_jv_{j+1})$, then $T(P)$ is a valid triangulation.*

Proof. We will show by induction that for all j, $\Delta v_{j+1}v_ju_{r(j)}$ is CCW with $u_{r(j)}$ in I_j. By Lemma 6.4(c), $\Delta v_1v_0u_{r(0)}$ is CCW with $u_{r(0)}$ in I_0. Suppose $0 < j < nc$ and $\Delta v_jv_{j-1}u_{r(j-1)}$ is CCW with $u_{r(j-1)}$ in I_{j-1}. If v_j is a convex vertex then $\Delta v_{j+1}v_ju_{r(j)}$ is CCW with $u_{r(j)}$ in I_j by Lemma 6.4(b). If v_j is a reflex vertex then it is given that $u_{r(j-1)}$ is to the right of $l(v_jv_{j+1})$ so $\Delta v_{j+1}v_ju_{r(j)}$ is CCW with $u_{r(j)}$ in I_j by Lemma 6.4(a). Therefore $\Delta v_{j+1}v_ju_{r(j)}$, $j = 0, 1, \cdots, nc-1$, are CCW triangles. By Lemma 6.1, $T(P)$ is a valid triangulation. □

CONJECTURE 6.1. If $|u_iu_{i+1}|$ is sufficiently small relative to h for all i (e.g. as produced by the spacing from (1.1), (5.1), (5.2)), then $T(P)$ is a valid triangulation.

Rationale. If $|u_iu_{i+1}|$ is sufficiently small relative to h for all i then for all reflex vertices v_j of C there exists a vertex u_k in I_{j-1} such that u_k is to the right of $l(v_jv_{j+1})$; in order for CCW triangle $\Delta v_jv_{j-1}u_k$ to be generated by procedure MERGE, either $u_k = u_{nb}$ or u_{k+1} must not be in the interior of the circle through the vertices v_{j-1}, v_j, u_k. It appears from the way that C is constructed from $int(P)$ that such a vertex u_k exists for all reflex vertices v_j and the hypothesis of Lemma 6.6 is satisfied, so $T(P)$ is a valid triangulation. □

If $|u_iu_{i+1}|$ is too large relative to h for some i, then procedure MERGE can produce an invalid triangulation $T(P)$ (see Fig. 5.4). Lemmas 6.3, 6.4, 6.5, and 6.6 suggest how procedure MERGE can be modified to produce a valid triangulation in this case. By Lemma 6.6, if $T(P)$ is not valid then there exists a reflex vertex v_j such that $u_{r(j-1)}$ is to the left of or on $l(v_jv_{j+1})$. Suppose v_j is the reflex vertex of smallest index such that $u_{r(j-1)}$ is to the left of or on $l(v_jv_{j+1})$ (e.g. in Fig. 5.4, $v_j = v_6$). Then $\Delta v_{k+1}v_ku_{r(k)}$, $k = 0, \cdots, j-1$, are CCW with $u_{r(k)}$ in I_k by Lemma 6.4 but $\Delta v_jv_{j-1}u_{r(j-1)}$ is an overlapping triangle by Lemma 6.3. Therefore instead of generating $\Delta v_jv_{j-1}u_{r(j-1)}$, generate CCW triangle $\Delta v_{j+1}v_jv_{j-1}$ which does not overlap any triangles in $C \cup$ (interior of C), replace subwalk $v_{j-1}v_jv_{j+1}$ of C with edge $v_{j-1}v_{j+1}$, and rerun procedure MERGE with B and the new shortened C which still lies in $int(P)$. If this process is repeated enough times when overlapping triangles $\Delta v_jv_{j-1}u_{r(j-1)}$ are detected, then the resulting triangulation is valid because either the hypothesis of Lemma 6.6 is satisfied or $C \cup$ (interior of C) eventually becomes a convex set so the hypothesis of Lemma 6.5 is satisfied. Procedure MMERGE of § 5 contains this modification to procedure MERGE in which the merge is restarted from edge $u_{r(j-2)}v_{j-1}$ instead of edge u_0v_0 (if $j \geqq 2$) when it is determined that a CCW triangle $\Delta v_{j+1}v_jv_{j-1}$ must be added to the triangulation, since the triangles in A between u_0v_0 and $u_{r(j-2)}v_{j-1}$ would remain the same. We have shown in the above discussion the following.

THEOREM 6.1. *The triangulation, $VT(P)$, produced by procedures* INTTRIANG *and* MMERGE *is a valid triangulation.*

7. Converting $VT(P)$ into a Delaunay triangulation. We describe a procedure for converting $VT(P)$ into a Delaunay triangulation, $DT(P)$. We discuss only the case of at least one interior vertex, since the case of no interior vertices is similar. From Lemma

4.1, the edges of the interior triangulation and C (as defined in § 4) are locally optimal in any triangulation of the mesh vertices which contains these edges. From Theorem 2.1, $VT(P)$ can be converted into a Delaunay triangulation by applying LOP to the internal edges in the triangulation of A until they are all locally optimal.

Procedure MMERGE produces $nt = nb + nc$ triangles, $TM(1)$, $TM(2), \cdots,$ $TM(nt)$, and nt edges, $EM(1), EM(2), \cdots, EM(nt)$, in A, where $ne \geqq 0$ edges are of the type $v_j v_{j+m}$, $m \geqq 2$, and the triangles and edges are ordered as follows. Edges $EM(k)$, $k = 1, \cdots, nt - ne$, are of the type $u_i v_j$ and are in the order that they are generated. Edges $EM(k)$, $k = nt - ne + 1, \cdots, nt$, are of the type $v_j v_{j+m}$, $m \geqq 2$, and are in the reverse order that they are generated (these edges are added to ensure the validity of the triangulation of A). For $k = 1, \cdots, nt - ne - 1$, $EM(k)$ is the common edge of $TM(k)$ and $TM(k+1)$. $EM(nt - ne) = u_0 v_0 = u_{nb} v_{nc}$ is the common edge of $TM(nt - ne)$ and $TM(1)$. For $k = nt - ne + 1, \cdots, nt$, $EM(k)$ is the common edge of $TM(k)$ and $TM(l)$ for some $l < k$. For $1 \leqq k \leqq nt$, let A_k be the region formed by the adjacent triangles $TM(1), \cdots, TM(k)$. We define an *internal* edge in a triangulation of A_k to be an edge of the type $u_i v_j$, $u_i u_{i+m}$, or $v_j v_{j+m}$, $m \geqq 2$, which is a common edge of two triangles in A_k.

The following step is performed for $k = 1, 2, \cdots, nt - 1$ to obtain locally optimal internal edges in the triangulation of A. Suppose the internal edges in the triangulation of A_k are locally optimal (this is trivially true for $k = 1$). Then the internal edges in the triangulation of A_{k+1} are made locally optimal as follows (with the exception mentioned in the next paragraph when $k = nt - ne - 1$). The triangulation of A_{k+1} consists of the triangles in A_k plus the triangle $TM(k+1)$. Let $e = EM(k)$ if $k \leqq nt - ne - 1$ and $e = EM(k+1)$ if $k \geqq nt - ne$. Then e is the only internal edge in the triangulation of A_{k+1} which may not be locally optimal. Therefore apply LOP to e. If it is swapped for the other diagonal edge of the quadrilateral, Q, formed by the two triangles having e as a common edge, then the edges of Q which are internal edges of A_{k+1} may no longer be locally optimal so they are placed in a stack of edges for which LOP must be applied. If the stack is not empty, then the top edge is popped from the stack and LOP is applied to this edge. This may cause another swap and more internal edges of A_{k+1} to be pushed onto the stack. This process of applying LOP is repeated until the stack is empty which implies that the internal edges in the triangulation of A_{k+1} are locally optimal. Lawson (1977) shows that this process must always terminate. (There are only a finite number of possible triangulations of A_{k+1}; a linear ordering of these triangulations can be defined using the sorted vector of the smallest angle in each triangle; and each swap produced by LOP causes a strict advance through this linear ordering of triangulations.)

The above step must be performed twice when $k = nt - ne - 1$ (i.e. the triangulation of A "closes") since $EM(k)$ and $EM(k+1) = u_0 v_0$ may not be locally optimal in the triangulation of A_{k+1}. First the above step is performed for $e = EM(k)$ and then it is performed for $e = EM(k+1)$. We have chosen u_0 so that $u_0 v_0$ is likely to be a Delaunay edge (see § 5) so $u_0 v_0$ is not likely to be swapped when LOP is applied to it. After the above step is performed for $k = nt - 1$, the internal edges in the triangulation of $A = A_{nt}$ and P are locally optimal and a Delaunay triangulation, $DT(P)$, is obtained by Lemma 4.1 and Theorem 2.1. Figure 5.3 illustrates a Delaunay triangulation $DT(P)$ obtained from $VT(P)$ by making one diagonal edge swap: $u_5 u_7$ replaces $u_6 v_3$ in the quadrilateral $u_5 u_6 u_7 v_3$.

The pseudo-code for converting the triangles of $VT(P)$ in A into Delaunay triangles is given in procedure CONVERT. This procedure can also be used in the case of no interior vertices.

```
Procedure CONVERT (inter, TM, EM, nt, ne);
# Input: if inter = true then TM, EM, and ne are output from procedure MMERGE
#          and nt = nb + nc
#        else TM and EM are output from procedure MERGE, nt = mb + mc − 2,
#          and ne = 0
# Output: updated list of triangles TM such that all triangles are Delaunay
#         and updated list of edges EM such that all edges are Delaunay
  for k := 1 to nt do
    if not inter and k = nt then return;
    if k ≦ nt − ne − 1 then kk := k + 1 else kk := k;
    top := 1;
    stack(top) := k;
    while top ≧ 1 do
      l := stack(top);
      top := top − 1;
      Search TM(kk), TM(kk − 1), ··· , TM(1) sequentially until the two
        triangles TM(m) and TM(n) containing edge EM(l) are found;
      Q := quadrilateral w1w2w3w4 where TM(m) = Δw1w2w3,
        TM(n) = Δw1w3w4, and EM(l) = w1w3;
      if w4 is in the circle through w1, w2, w3 then
        TM(m) := Δw1w2w4;
        TM(n) := Δw2w3w4;
        EM(l) := w2w4;
        w5 := w1;
        for q := 1 to 4 do
          if 1 ≦ index (wqwq+1) ≦ k then
            top := top + 1;
            stack(top) := index (wqwq+1);
  return;
```

In this procedure, the function $index(ww')$ is defined to be 0 if ww' is an edge of the type u_iu_{i+1} or v_jv_{j+1}; otherwise it is the index of edge ww' in the list EM. The function is used to determine whether an edge is an internal edge in the triangulation of A_{k+1}. For each triangle $\Delta w_1w_2w_3$ in the list TM, we store the values $index(w_1w_2)$, $index(w_2w_3)$, and $index(w_3w_1)$ in addition to the vertices of the triangle, so that it is easy to determine $index(w_qw_{q+1})$ in the innermost for loop. When LOP is applied to an internal edge e, the list TM must be searched for the two triangles containing e. The reason for the sequential search starting backwards from $TM(kk)$ will be discussed below.

We now estimate the number of edge swaps required for converting $VT(P)$ into $DT(P)$ since the efficiency of the procedure depends on the number of swaps. In $DT(P)$, strip A may contain edges of the types u_iv_j, u_iv_{j+m}, $m \geqq 2$. In $VT(P)$, A contains mostly edges of type u_iv_j and possibly some edges of type v_jv_{j+m}. If u_iv_j is a Delaunay edge then it is likely to be an edge of $VT(P)$ since u_0v_0 is likely to be a Delaunay edge and in cases (a), (b), and (c) of (5.3) the next edge $u_{i+1}v_j$ or u_iv_{j+1} is chosen to be locally optimal in the quadrilateral $u_iu_{i+1}v_{j+1}v_j$. So edge swaps are needed to obtain Delaunay edges of types u_iu_{i+m} and v_jv_{j+m}. The following theorem indicates when no swaps are required.

THEOREM 7.1. *If u_0v_0 is a Delaunay edge and there are no Delaunay edges of types u_iu_{i+2} or v_jv_{j+2} then $VT(P) = T(P)$ is a Delaunay triangulation.*

Proof. First we show that under the hypothesis of the theorem there are no Delaunay edges of types u_iu_{i+m} or v_jv_{j+m}, $m \geqq 3$. Suppose u_lu_{l+m}, $m \geqq 3$, is a Delaunay edge. Then $u_lu_{l+1} \cdots u_{l+m}u_l$ must form a simple subpolygon of P which contains no vertices in its interior. In a Delaunay triangulation of the mesh vertices, this subpolygon contains $m-1$ triangles involving only the vertices $u_l, u_{l+1}, \cdots, u_{l+m}$. Therefore that

must be a Delaunay edge u_iu_{i+2} where $l \leqq i \leqq l+m-2$. This contradicts the hypothesis, therefore there are no Delaunay edges of type u_iu_{i+m}, $m \geqq 3$. Similarly there are no Delaunay edges of type v_jv_{j+m}, $m \geqq 3$. Therefore the Delaunay edges in A are all of the type u_iv_j.

We show by induction that the edges produced by procedure MERGE (which are the same as those produced by procedure MMERGE) are Delaunay edges. Suppose u_iv_j is a Delaunay edge (initially u_0v_0 is a Delaunay edge). Then $\Delta u_iu_{i+1}v_j$ or $\Delta v_{j+1}v_ju_i$ must be a Delaunay triangle and $u_{i+1}v_j$ or u_iv_{j+1} must be a Delaunay edge, respectively. If $i = nb$, then $\Delta v_{j+1}v_ju_i$ is the only possible triangle, therefore u_iv_{j+1} is a Delaunay edge. Similarly if $j = nc$, then $u_{i+1}v_j$ is a Delaunay edge. Suppose $i < nb$ and $j < nc$. Consider quadrilateral $Q = u_iu_{i+1}v_{j+1}v_j$ in (5.3). In case (a), the circle test chooses the next Delaunay edge. In cases (b), (c), and (d) the chosen triangle is a Delaunay triangle since the other triangle is not a "valid" triangle—it is a CW or overlapping triangle (see Fig. 5.2). Therefore procedure MERGE generates the next Delaunay edge, either $u_{i+1}v_j$ or u_iv_{j+1}. By induction, procedure MERGE produces Delaunay edges in A. The edges generated by procedure INTTRIANG are Delaunay edges by Lemma 4.1. Therefore $VT(P) = T(P)$ is a Delaunay triangulation. □

In general, $DT(P)$ contains edges of types u_iu_{i+m} and v_jv_{j+m}, $m \geqq 2$. The location of the mesh vertices and Lemma 2.1 can be used to determine where these Delaunay edges are likely to occur. We first suppose that the parameters h_i satisfy (1.1). An edge v_jv_{j+m} may be a Delaunay edge if it lies entirely in A and angle $(v_{j+m}v_{j+l}v_j)$ for some l, $0 < l < m$, is smaller than approximately 120°, e.g. the edge joining $v_j = (x_{i+1,0}, y_{i+1})$ and $v_{j+2} = (x_{i+2,0}, y_{i+2})$ in Fig. 4.2. It is unlikely that $m > 2$ and the number of Delaunay edges of type v_jv_{j+2} and the number of swaps to obtain these edges are expected to be small relative to nc.

An edge u_iu_{i+m} may be a Delaunay edge if it is near a vertex of P or $int(P)$ with an interior angle smaller than approximately 90°, e.g. u_5u_7 in Fig. 5.3. If P does not contain "small" interior angles (e.g. ≦20°), then m is likely small relative to nb and the number of Delaunay edges of type u_iu_{i+m}, $m \geqq 2$, and the number of swaps to obtain these edges are expected to be small relative to nb. If P contains a small interior angle (from elementary geometry it can be seen that at most two interior angles of a convex polygon can be less than 60°), then as this angle decreases, the maximal value m of a Delaunay edge u_iu_{i+m} near this angle increases and the number of swaps required to obtain the Delaunay edges in the subpolygon $u_iu_{i+1} \cdots u_{i+m}u_i$ increases. Therefore if the h_i satisfy (1.1) and P contains no small interior angles, then the number of swaps needed to obtain $DT(P)$ from $VT(P)$ is expected to be small relative to $nb + nc$.

Now we suppose that the h_i do not satisfy (1.1) for all i. As the ratios h_i/h increase, more Delaunay edges of type v_jv_{j+m}, $m \geqq 2$, are likely to occur in $DT(P)$ and be produced in $VT(P)$ by procedure MMERGE. As the ratios h_i/h decrease, more Delaunay edges of type u_iu_{i+m}, $m \geqq 2$, are likely to occur in $DT(P)$ and the number of swaps needed to obtain $DT(P)$ from $VT(P)$ can be $O(nb^2)$ if P contains a small interior angle.

When LOP is applied to edge $EM(k)$ in A_{k+1} for $k \leqq nt - ne - 1$, edge swaps, if any, occur for triangles $TM(n)$ near $TM(k+1)$ with indices $n \leqq k+1$. Therefore we store the triangles of A in a linear list and when LOP is applied to an edge e in the triangulation of A_{k+1}, a search is made sequentially backwards in this list starting from $TM(k+1)$ until the two triangles containing edge e are found. The number of triangles searched is expected to be small constant if P does not contain small interior angles. When LOP is applied to edge $EM(nt-ne) = u_0v_0$ in A_{nt-ne}, one of the triangles containing edge u_0v_0 is near the end of list TM and the other is near the front of TM.

By using this information, few triangles are searched when LOP is applied to u_0v_0. If $VT(P)$ contains an edge of type v_jv_{j+m}, $m \geqq 2$, then $O(nt)$ triangles are expected to be searched when LOP is applied to this edge since the two triangles containing this edge may not be close together in *TM*. By Conjecture 6.1, *ne* is expected to be zero if the h_i satisfy (1.1). Therefore if P contains no small interior angles and the h_i satisfy (1.1), then the number of traingles searched in the applications of LOP is expected to be $O(nb+nc)$.

8. Summary and time complexity. The pseudo-code for our triangulation algorithm is summarized in procedure DELTRIANG.

```
Procedure DELTRIANG (P, m, hlist, T, n_t);
# Input: convex polygon P with m vertices and list of
#        triangle size parameters hlist = [h, h_1, h_2, ..., h_m]
# Output: list of triangles T and number of triangles n_t
  SHRINK (P, m, h/√2, int(P), ni);
  if ni > 0 then inter := true else inter := false;
  if inter then
     Rotate coordinate system so that diameter of int(P) is parallel to y-axis;
     INTTRIANG (int(P), h, TI, nt, C, nc);
  else
     Rotate coordinate system so that diameter of P is parallel to y-axis;
     TI := [ ]; C := [ ]; nt := 0; nc := -2;
  for i := 1 to m do
     Generate mesh vertices on edge e_i at an equal spacing of h̄_i
     as computed by (5.1);
  if inter then
     Determine B = [u_0, u_1, ..., u_nb];
     MMERGE (B, nb, C, nc, TM, EM, ne);
  else
     Determine B_L = [u_0, u_1, ..., u_mb], B_R = [u_nb, u_nb-1, ..., u_mb];
     MERGE (inter, B_L, mb, B_R, nb - mb, TM, EM);
     ne := 0;
  CONVERT (inter, TM, EM, nb + nc, ne);
  T := append (TI, TM);
  n_t := nt + nb + nc;
  return;
```

We now discuss the time complexity for procedure DELTRIANG to construct the Delaunay triangulation, $DT(P)$, in P. Let n_t and n_v be the number of triangles and mesh vertices, respectively, generated in P by procedure DELTRIANG. Let *nb* be the number of mesh vertices on ∂P and m be the number of vertices of P. n_t, n_v, *nb*, and m are related by the formulas (Lawson (1977))

$$n_t = 2n_v - nb - 2 \quad \text{and} \quad n_t \geqq nb - 2 \geqq m - 2. \tag{8.1}$$

By Theorem 3.1, $int(P)$ is determined from P in $O(m)$ time by procedure SHRINK. The diameter of $int(P)$ or P is found in $O(m)$ time (Shamos (1975)). In procedure INTTRIANG, the generation of mesh vertices in $int(P)$ requires $O(n_v)$ time and the generation of triangles and closed walk C requires $O(n_t)$ time. The generation of mesh vertices on ∂P requires $O(nb)$ time. The generation of triangles in A by procedure MMERGE requires $O(n_t)$ time if *ne*, the number of edges of type v_jv_{j+k}, $k \geqq 2$, is zero or bounded by a small constant (*ne* is expected to be zero by Conjecture 6.1 if the h_i satisfy (1.1)). We conjecture that the time complexity of procedure MMERGE is $O(n_t)$ even if *ne* is large. In the case of no interior vertices, the generation of triangles in P by procedure MERGE requires $O(n_t)$ time. From the discussion at the end of § 7, the

number of edge swaps and the number of triangle searches in the applications of LOP in procedure CONVERT are expected to be $O(n_t)$ if P contains no small interior angles and the h_i satisfy (1.1). Therefore if P contains no small interior angles and the h_i satisfy (1.1), then the time complexity for constructing $DT(P)$ by procedure DELTRIANG is expected to be $O(n_t) = O(n_v)$ using (8.1). Otherwise the time complexity of procedure DELTRIANG may be nonlinear in n_t since the number of swaps in procedure CONVERT may be $O(nb^2)$.

Procedure DELTRIANG has been implemented in PASCAL and used to generate triangular meshes in over a thousand convex polygons created from the decomposition of various regions by our finite element triangulation method (Joe (1984)). There are few interior angles less than 20° in these polygons and the parameters h_i satisfy (1.1). We made the following observations about the performance of procedure DELTRIANG. In the case of at least two interior vertices, the heuristic for selecting u_0 so that u_0v_0 is a Delaunay edge (see § 5) has not failed, and no edges of type v_jv_{j+k}, $k \geqq 2$, have been generated by procedure MMERGE, i.e. procedures INTTRIANG and MERGE have not produced an invalid triangulation $T(P)$. Also, the average number of edge swaps per polygon and the average number of triangle searches per application of LOP are small constants. Therefore the empirical time complexity of procedure DELTRIANG is $O(n_t)$ for these polygons and triangle size parameters.

Acknowledgment. The author would like to thank Prof. R. B. Simpson for many helpful discussions and his valuable comments on preliminary drafts of this paper.

REFERENCES

R. E. BANK (1982), PLTMG *users' guide*, Dept. Mathematics, Univ. California at San Diego.

A. BYKAT (1976), *Automatic generation of triangular grid*: I—*subdivision of a general polygon into convex subregions*; II—*triangulation of convex polygons*, Int. J. Num. Meth. Engng., 10, pp. 1329-1342.

J. C. CAVENDISH (1974), *Automatic triangulation of arbitrary planar domains for the finite element method*, Int. J. Num. Meth. Engng., 8, pp. 679-696.

B. JOE (1984), *Finite element triangulation of complex regions using computational geometry*, Ph.D. thesis, Dept. Computer Science, Univ. Waterloo, Waterloo, Ontario; also Research Report CS-84-19.

B. JOE AND R. B. SIMPSON (1984), *Triangular meshes for regions for complicated shape*, submitted to Int. J. Num. Meth. Engng.

C. L. LAWSON (1977), *Software for C^1 surface interpolation*, Mathematical Software III, J. R. Rice, ed., Academic Press, New York, pp. 161-194.

D. T. LEE AND R. P. PREPARATA (1979), *An optimal algorithm for finding the kernel of a polygon*, J. Assoc. Comput. Mach., 26, pp. 415-421.

D. T. LEE AND B. J. SCHACHTER (1979), *Two algorithms for constructing a Delaunay triangulation*, Rep. 79ASD007, General Electric Co., Daytona Beach, FL.

M. I. SHAMOS (1975), *Geometric complexity*, Proc. 7th Annual ACM Symposium on Theory of Computing, pp. 224-233.

M. I. SHAMOS AND D. HOEY (1975), *Closest-point problems*, Proc. 16th Annual Symposium on Foundations of Computer Science, pp. 151-162.

R. D. SHAW AND R. G. PITCHEN (1978), *Modifications to the Suhara-Fukuda method of network generation*, Int. J. Num. Meth. Engng., 12, pp. 93-99.

R. SIBSON (1978), *Locally equiangular triangulations*, Computer J., 21, pp. 243-245.

R. B. SIMPSON (1981), *A two-dimensional mesh verification algorithm*, this Journal, 2, pp. 455-473.

SIAM J. SCI. STAT. COMPUT.
Vol. 7, No. 2, April 1986

013

PLTMGC: A MULTI-GRID CONTINUATION PROGRAM FOR PARAMETERIZED NONLINEAR ELLIPTIC SYSTEMS*

RANDOLPH E. BANK† AND TONY F. CHAN‡

Abstract. PLTMGC is a program package for solving nonlinear elliptic systems that have explicit dependence on a scalar parameter. In addition to being able to compute solutions for fixed parameter values, it can be used to solve the linear eigenvalue problem, trace solution branches, locate singular points (simple turning points and bifurcation points) and switch branch at simple bifurcation points. A multi-grid continuation approach is employed in which a continuation procedure is used to follow the solution curve on the coarsest grid and a multi-grid algorithm is used to refine the solution at selected points using an adaptive mesh refinement strategy. Some numerical examples illustrating the performance of the package are given.

Key words. continuation methods, multi-grid algorithms, nonlinear elliptic systems, eigenvalue problems, turning point, bifurcation, adaptive mesh refinement

AMS (MOS) subject classification. 65

1. Introduction. PLTMGC is a program package for solving two-dimensional nonlinear elliptic boundary value problems of the form:

$$\begin{aligned} &-\nabla d\nabla u + f(x, y, u, \nabla u, \lambda) = 0 \quad \text{in } \Omega, \\ &u = g_1(x, y) \quad \text{on } \partial\Omega_1, \\ &(d\nabla u)\cdot n = g_2(x, y, u, \lambda) \quad \text{on } \partial\Omega - \partial\Omega_1. \end{aligned} \tag{1}$$

The package is specifically designed to take into account the explicit dependence of the system (1) on the scalar parameter λ. In this paper, we shall denote the system (1) by

$$G(u, \lambda) = 0, \tag{2}$$

where $u \in B$ (a real Banach space), $\lambda \in R$, and G is a continuously differentiable operator mapping $B \times R$ into B.

There are at least three situations in which it is important to consider the parameter λ explicitly. The first case is when the system (1) is a mathematical model for a physical problem, in which u may represent a physical field variable (e.g. flow field, structural displacement) and λ may be related to a physical parameter (e.g. Reynold's number, load on a structure) and one is interested in the dependence of u on λ. The second class of problems are linear and nonlinear eigenvalue problems [3] which can be cast in the form of (1). The third situation arises in the application of homotopy continuation methods for obtaining good initial guesses to highly nonlinear problems [24]. For example, consider the following nonlinear elliptic system:

$$Lu + N(u) = 0, \tag{3}$$

where L is a linear elliptic operator and N is a highly nonlinear operator. Simple Picard-type iteration or even direct application of Newton's method to (3) may fail unless an extremely good initial guess is available. Assuming that the problem $Lu = 0$

* Received by the editors January 24, 1984, and in revised form January 8, 1985.

† Department of Mathematics, University of California at San Diego, La Jolla, California. The work of this author was supported by the Office of Naval Research under contract N00014-82-K-0197.

‡ Department of Computer Science, Yale University, Yale Station, New Haven, Connecticut 06520. The work of this author was supported in part by the Department of Energy under grant DE-AC02-81ER10996 and by the Army Research Office under grant DAAG-83-0177.

is easier to solve, one may consider introducing an artificial parameter λ in (3) to obtain the following modified system:

$$G(u, \lambda) \equiv Lu + \lambda N(u) = 0$$

and slowly increasing λ from 0 to 1, using an old solution as initial guess for the next.

Thus, solving the system (2) may consist of determining the dependence of the solution $u(\lambda)$ on the parameter λ, i.e. in tracing the solution branches $[u(\lambda), \lambda]$ of (2), in addition to determining the solution u at some given value of λ or $\|u\|$ (target points). (Throughout this paper, $\|u\|$ denotes the usual L_2 norm of u, namely, $\int_\Omega u^2\, dx\, dy$.) When the operator G is nonlinear in u and λ, this can be accomplished numerically by applying an approximate Newton method to (2) for a fixed value of λ, which makes use of the Jacobian matrix $G_u(u, \lambda)$. However, the solution branches often display very interesting and complicated behavior, among which are existence of multiple solutions and singular points (where $G_u(u, \lambda)$ is singular) known as *turning points* (where the solution branch bends back on itself) and *bifurcation points* (where two or more solution branches cross). Straightforward application of Newton's method to (2) encounters difficulties near these singular points. To overcome these difficulties, a path following continuation method [2], [29], [34], [39] is usually employed. These continuation methods are designed to trace past turning points and can be used to switch branches at bifurcation points.

PLTMGC is a derivative of the program package PLTMG [7] which solves nonlinear elliptic systems (with no parameter dependence) of a similar form to (1). It employs a finite element discretization of (1) based on C^0 piecewise linear triangular finite elements and uses a multi-grid approach which obtains the solution on the finest grid by iterating through a hierarchy of coarser grids. The grids are obtained by adaptive mesh refinement techniques which adaptively refine the mesh only in regions where the solution error is large. In order to use PLTMGC, the user has to supply a coarse grid on which the continuation procedures are applied. At selected points, the multi-grid procedures are called on to adaptively refine the grid and obtain the solution on the fine grids.

PLTMGC differs from other continuation program packages [31], [41], [43] in that we treat directly partial differential equations and exploit the inherent structures of the *PDE* in the continuation procedures through the use of multi-grid techniques. We feel that this is crucial for the efficiency of the overall method. In some sense, PLTMGC represents our attempt to cohesively integrate algorithms and ideas from several divergent sources. While many of the algorithms in PLTMGC are taken directly from the existing literature, we have also developed many new ones which are especially suited to PLTMGC and which contribute to the efficiency and robustness of the package. Among these are: a new pseudo-arclength parameterization that facilitates computing target values in λ and $\|u\|$, a new robust predictor, multi-grid deflation techniques for ensuring numerical stability near singular points and a new algorithm for locating simple turning points and bifurcation points. Our goal in this paper is to present only an overview of PLTMGC which summarizes our (current) view as to how these various ideas *should* be synthesized. To remain true to this goal (and to keep the paper of manageable size), we have purposely omitted many technical details which are obviously very important to a successful implementation. Where possible, we provide references to more complete explanations.

The remainder of the manuscript is organized as follows. In § 2, we summarize continuation algorithms, multi-level algorithms and adaptive mesh refinement techniques. In § 3, we describe how these ideas are merged in PLTMGC. Section 4 contains

some numerical examples, illustrating the procedures of § 3. Section 5 contains some concluding remarks and possible extensions.

2. Continuation and multi-level algorithms. In this section, we first review the major components of a basic continuation procedure. Next we summarize those aspects of the multi-level iterative method which are most important to the continuation process. Since much of the analysis of these methods appears elsewhere, (see for example [9], [13], [15], [25], [27], [29], [38]), our exposition is mainly descriptive. This section is divided into four subsections; in § 2.1, we review continuation techniques and in § 2.2, we describe a Newton-multi-level procedure in the context of nested iteration [13], [25]. In § 2.3, we describe the j-level iterative method used to solve the sparse linear systems which arise [8], [11]. Finally, in § 2.4, we describe the use of local adaptive mesh refinement.

2.1. Continuation algorithms. In this section, we review the essential features of path-following continuation methods [20], [24], [29], [40]. The key idea is to parameterize the solutions $[u(\sigma), \lambda(\sigma)]$ in terms of a new parameter σ that approximates the arclength parameter s, instead of parameterizing $u(\lambda)$ in terms of the natural parameter λ. This is usually achieved by augmenting the equation (2) by an auxiliary equation that approximates the arclength condition:

$$\|\dot{u}(s)\|^2 + |\dot{\lambda}(s)|^2 = 1, \tag{4}$$

to give an inflated system with unknowns $u(\sigma)$ *and* $\lambda(\sigma)$:

$$\begin{aligned} G(u(\sigma), \lambda(\sigma)) &= 0, \\ N(u(\sigma), \lambda(\sigma), \sigma) &= 0. \end{aligned} \tag{5}$$

Instead of solving for $u(\lambda)$ for a given value of λ, we solve for $u(\sigma)$ and $\lambda(\sigma)$ for a given value of σ. The auxiliary function N is constructed so that the target solution (u, λ) is a regular solution of (5) [29], even at simple turning points. Thus, a regular nonlinear iterative method can be applied to solve this inflated system.

Most continuation methods are of the predictor-corrector type, usually with a predictor along the tangent:

$$(u_p, \lambda_p) = (u_0 + \alpha_p \dot{u}_0, \lambda_0 + \beta_p \dot{\lambda}_0) \tag{6}$$

with the predictor step (α_p, β_p), and where the unit tangent $(\dot{u}_0, \dot{\lambda}_0)$ at a known solution (u_0, λ_0) is defined by:

$$\begin{aligned} G_u(u_0, \lambda_0)\dot{u}_0 + G_\lambda(u_0, \lambda_0)\dot{\lambda}_0 &= 0, \\ \|\dot{u}_0\|^2 + |\dot{\lambda}_0|^2 &= 1. \end{aligned} \tag{7}$$

The predictor (u_p, λ_p) is then used as an initial guess for a nonlinear iteration used to solve the system (5). The predictor step (α_p, β_p) is usually chosen adaptively to ensure convergence of the corrector. If the predictor fails to achieve convergence in the corrector, then the parameterization N and σ are adjusted (e.g. the continuation step can be reduced) and the process repeated.

In addition, certain points on the solution curve demand special attention. For example, one may want to compute the solution at selected values of λ or $\|u\|$ (target points). Moreover, one may want to be able to detect singular points and locate them accurately. In case a bifurcation is detected, one may also want to switch to the bifurcating branch.

We summarize the essential features in the following general algorithm for one step of the continuation procedure:

CONTINUATION PROCEDURE

1. Compute the unit tangent $[\dot{u}_0, \dot{\lambda}_0]$ at $[u_0, \lambda_0]$ by (7).
2. Determine the local parameterization N and σ.
3. If close to a target point, adjust N and σ so that $[u(\sigma), \lambda(\sigma)]$ will be the target point.
4. Compute a predicted solution $[u_p, \lambda_p]$.
5. Use $[u_p, \lambda_p]$ as initial guess in a nonlinear iteration for solving the system (5) to obtain $[u(\sigma), \lambda(\sigma)]$. If the iteration fails to converge, then adjust N and σ and go to step 4.
6. If a limit point is detected, compute it accurately if desired.
7. If a bifurcation point is detected, compute it accurately and switch branch if desired.

2.2. Nested iteration. For concreteness, we consider the nonlinear elliptic equation:

$$\begin{aligned} &\mathscr{L}(u) \equiv -\nabla \cdot d\nabla u + f(u, \nabla u, \lambda) = 0 \quad \text{in } \Omega, \\ &u = 0 \quad \text{on } \partial\Omega, \end{aligned} \tag{8}$$

where, for simplicity, Ω is a closed, polygonal region in R^2. We assume $d > 0$ in $\bar{\Omega}$ and d and f are smooth functions of their arguments. We will assume for the moment that $\lambda \in R$ is fixed, and that for this value of λ, (8) is well-posed, and has one or more isolated solutions.

A weak form of (8) is: Find $u \in H_0^1(\Omega)$ such that

$$a(u, v) = 0 \quad \forall v \in H_0^1(\Omega), \tag{9}$$

where

$$a(u, v) = \int_\Omega \{d\nabla u \nabla v + f(u, \nabla u, \lambda) v\}\, dx\, dy. \tag{10}$$

$H^1(\Omega)$ will denote the usual Sobolev space equipped with the norm

$$\|u\|_1^2 = \int_\Omega \{|\nabla u|^2 + u^2\}\, dx\, dy$$

and

$$H_0^1(\Omega) = \{v \in H^1(\Omega) | v|_{\partial\Omega} = 0\}.$$

For $u \in H_0^1(\Omega)$, we define the form $b(u; v, w)$, linear in its last two arguments, by

$$b(u; v, w) = \int_\Omega \{d\nabla v \nabla w + e \cdot \nabla v\, w + cvw\}\, dx\, dy \tag{11}$$

where

$$e_i = \frac{\partial f}{\partial u_{x_i}}(u, \nabla u, \lambda),$$

$$c = \frac{\partial f}{\partial u}(u, \nabla u, \lambda).$$

We assume that for u sufficiently close to an isolated solution of (9), the linear elliptic problem:

Find $z \in H_0^1(\Omega)$ such that

$$b(u; z, v) = g(v) \quad \forall v \in H_0^1(\Omega) \tag{12}$$

has a unique solution, where g is a smooth linear functional [9].

Let $\mathcal{T}_1$ be a quasi uniform shape regular triangulation of Ω. Let $M_1 \subset H_0^1$ be the N_1-dimensional space of C^0 piecewise linear polynomials associated with $\mathcal{T}_1$. We inductively define a sequence of triangulations $\{\mathcal{T}_j\}$ with $\mathcal{T}_j$ being a refinement of $\mathcal{T}_{j-1}$ (though not necessarily a uniform refinement), and corresponding N_j-dimensional spaces $M_j \subset H_0^1$. We assume $N_j \geqq \beta N_{j-1}$ for $\beta > 2, j > 1$; $\beta \approx 4$ is typical for elliptic equations in R^2. Because $\mathcal{T}_j$ is a refinement of $\mathcal{T}_{j-1}$, M_{j-1} will be a subspace of $M_j, j > 1$.

The discrete form of (9) is: Find $u_j \in M_j$ such that

$$a(u_j, v) = 0 \quad \forall v \in M_j. \tag{13}$$

We assume that for any isolated solution u of (9) there is a corresponding sequence of isolated solutions $\{u_j\}$ of (13), and that these solutions converge to u at a specific rate, i.e.

$$\|u - u_j\|_1 \leqq c N_j^{-q}, \tag{14}$$

where $c = c(u, \mathcal{T}_1)$ is a constant and $q > 0$.

These discrete problems can be solved by approximate Newton methods [13], [21]. Given $u_j^{(0)} \in M_j$, we compute a sequence $u_j^{(k)} \in M_j$, using

$$\begin{aligned} &b_j(x^{(k)}, v) = -a(u_j^{(k)}, v) \quad \forall v \in M_j, \\ &u_j^{(k+1)} = u_j^{(k)} + t^{(k)} x^{(k)}. \end{aligned} \tag{15}$$

The case $b_j(\cdot,\cdot) = b(u_j^{(k)}; \cdot, \cdot)$, $t^{(k)} = 1$ is Newton's method. Equations (15) describe an approximate Newton method because the bilinear form $b_j(\cdot,\cdot)$ is chosen to approximate the Jacobian (11). For example, b_j might correspond to the approximate solution of the Newton equations using an iterative method as an inner iteration [13]. The damping parameters $t^{(k)} \in (0, 1]$ can be chosen to guarantee convergence [12], [21].

The nested iteration strategy (also called full multi-grid by Brandt [15]) involves sequentially solving the problems (13) for $j = 1, 2, \cdots$ using the approximate solution of the $(j-1)$-st problem as the initial guess for the jth.

ALGORITHM NI:

1. Solve (13) for $j = 1$ using S_1 iterations of (15) and an appropriate initial guess $u_j^{(0)}$.

2. Solve (13) for $j > 1$ using S_j iterations of (15) and initial guess $u_j^{(0)} = u_{j-1}^{(S_{j-1})}$.

Under appropriate hypothesis [13] one can show Algorithm NI to be an extremely effective procedure for solving (13). For each problem after the first, one must reduce the initial error by only a fixed amount $\varepsilon < \beta^{-q}$ independent of j in order for the computed solutions $u_j^{(S_j)}$ to converge to the true solution u of (9) at the same rate as the u_j (see (14)). Because of this modest required error reduction, one can often show $S_j = 1$ for j sufficiently large is adequate [13]. Because the N_j increases geometrically, asymptotically the cost of solving the nonlinear problem at level j is only a small multiple (usually less than 2) of the cost of approximately solving one set of linear equations of the form (15) for the grid $\mathcal{T}_j$. If this set of equations can be solved in $O(N_j)$ operations (as it frequently can using the j-level iterative method), then the overall complexity will be of optimal order $O(N_j)$ [13].

2.3. The j-level iterative method. The j-level iterative method is designed to solve linear systems of the form:

Find $z \in M_j$ such that

$$b(u; z, v) = g(v) \quad \forall v \in M_j. \tag{16}$$

Here $g: M_j \to R$ is a linear functional. In the special case $u = u_j^{(k)}$, $g(v) = -a(u_j^{(k)}, v)$, (16) becomes the Newton equation for (13), and z can be identified as $x^{(k)}$.

For $j = 1$, equations (16) are solved directly, e.g., by sparse Gaussian elimination. For $j > 1$, a single iteration of the j-level scheme takes an initial guess $z^{(0)} \in M_j$ to a final guess $z^{(m+1)} \in M_j$. The procedure is defined recursively as follows:

For $1 \leqq k \leqq m$,

$$z^{(k)} = S(z^{k-1)}), \text{ where } S \text{ is a smoothing operator.} \tag{17}$$

Set $\bar{g}(v) = g(v) - b(u; z^{(m)}, v)$ and solve the coarse grid problem: find $\delta \in M_{j-1}$ such that

$$b(u; \delta, v) = \bar{g}(v) \quad \forall v \in M_{j-1}, \tag{18}$$

using 2 iterations of the $j-1$ level scheme and initial guess zero for $j-1 > 1$ or directly if $j-1 = 1$. In either case we obtain an (approximate) solution $\bar{\delta} \in M_{j-1}$. Finally, we set

$$z^{(m+1)} = z^{(m)} + \bar{\delta}. \tag{19}$$

The operation S corresponds to the application of some iterative method (e.g. symmetric Gauss-Seidel, damped Jacobi) to the set of equations (16). Typically, m is quite small ($\leqq 4$). Equations (18) and (19) are now commonly called a "coarse grid correction"; it is easy to see that $\delta \in M_{j-1}$ is a Galerkin approximation to the error $z - z^{(m)} \in M_j$. From (16) and (18) we have

$$b(u; z - z^{(m)}, v) = \bar{g}(v) \quad \forall v \in M_{j-1}.$$

Under suitable hypothesis on the triangulations $\mathcal{T}_j$, subspaces M_j (and their bases), and the smoothing iteration, one can prove that the work required to reduce the error by any fixed amount is $O(N_j)$, leading to the overall optimality of the j-level iterative method when used in conjunction with nested iteration.

2.4. Adaptive mesh refinement. In the use of nested iteration and the j-level iteration the triangulation $\mathcal{T}_{k+1}$ is not required until the approximate solution $u_k^{(s_k)}$ has been computed. This suggests the possibility of using the computed solution $u_k^{(s_k)}$ in the computation of $\mathcal{T}_{k+1}$ by a local adaptive mesh refinement algorithm [4], [5], [6], [7], [15], [44]. The basic goal is to construct a triangulation in which the error is (approximately) uniformly distributed among the elements by refining those elements in which the local error is largest. Suppose element $\bar{t} \in \mathcal{T}_k$ has the largest error among the elements of $\mathcal{T}_k$. Further, suppose that the errors in elements obtained by the refinement of $\bar{t}$ would have local errors of size ε. Then a reasonable refinement criteria is to refine those elements in $\mathcal{T}_k$ whose error satisfies $\|e\|_t > \varepsilon$. This will hopefully result in a new mesh in which all elements have errors bounded by ε, and will tend to equi-distribute the error among the elements. The threshold value ε can be extrapolated using the calculated elementwise error estimates from the previous two levels. Since only a few elements may have errors larger than ε, the procedure may have to be iterated several times in order to compute a triangulation $\mathcal{T}_{k+1}$ whose corresponding subspace M_k has $N_{k+1} \approx \beta N_k$. The scheme used in PLTMGC is described in detail in [7], [14].

3. Implementation in PLTMGC. In this section, we summarize our strategies for combining the methods of § 2 into an efficient and robust procedure for solving (1). There are several possible interpretations of what constitutes the solution of (1). From our present perspective, we desire a procedure which can yield a qualitatively accurate description of all the solution curves, with approximate locations of limit points and bifurcation points. At a few selected points, perhaps only one, we would like to compute an accurate approximation of the solution u.

Thus our overall strategy involves two main components:

1. The use of the continuation process of § 2.1 on the coarsest mesh $\mathcal{T}_1$ only.
2. The use of the multi-level procedure of §§ 2.2–2.4 to produce accurate solutions at a few selected points.

This view is in contrast to that of some other investigators, e.g. Mansfield [32], who suggest schemes for carrying out the continuation process on the finest grid. When a quantitatively accurate description of the solution curves is desired, such schemes would seem more natural than ours because they make use of previously generated solutions on the finer meshes to generate initial guesses for the solutions on the finer meshes in the multi-level algorithm, as suggested by Hackbusch [27] for example. However, we feel that neither the convergence nor the efficiency of the multi-level algorithm is critically sensitive to the accuracy of these initial guesses, because initial guesses interpolated from coarser grids are accurate enough to insure rapid convergence. Also, because of our use of adaptive mesh refinement, in general our procedure will not generate the same sequence of meshes at all points on the solution curves. Finally, following the solution curves on all the finer meshes is far more costly, perhaps by orders of magnitude if several levels are involved. Thus, following solution curves on the coarsest mesh only is advantageous when the main goal of the computation is to compute a particular solution for a particular value of λ or $\|u\|$ and if only a general qualitative behavior of the solution curve is sufficient. This necessarily implies that the coarsest grid has to be fine enough to represent the qualitative behavior of the solution curves of interest. Brandt [15] has suggested using the frozen-tau technique to obtain fine grid accuracy on the coarse grids.

With respect to coarse grid continuation, there are three main tasks which should be handled by a general solver:

1. Continuation from a starting value to a target value.
2. Location of limit points and bifurcation points.
3. Branch switching at bifurcation points.

With respect to the adaptive refinement at a selected point, two cases must be handled:

1. Refinement away from singular points.
2. Refinement near a singular point.

3.1. Computation of tangent field. A major component of a continuation method is the computation of the unit tangent $[\dot{u}(s), \dot{\lambda}(s)]$ to the solution curve at a point $[u, \lambda]$ on the solution curve, which can be computed relatively inexpensively from its definition by solving only *one* linear system with G_u:

$$\begin{aligned} G_u z &= -G_\lambda. \\ \dot{\lambda}^2 &= 1/(\|z\|^2+1). \\ \dot{u} &= \dot{\lambda} z. \end{aligned} \tag{20}$$

Equation (20) determines $[\dot{u}, \dot{\lambda}]$ up to a directional orientation, which can be fixed by

some convention. For example, the strategy we employ is to require that the new tangent forms an acute angle with the previous tangent to insure that we are "following" the solution curve.

Although the above procedure is not mathematically well defined when G_u is exactly singular, in practice it behaves like inverse iteration and produces the right results.

3.2. Local parameterizations. The most popular choices for the local parameterization N are usually some approximations to the arclength condition (4) to ensure that at least locally the solution curve is "followed". A few typical N's that have been used in the literature are:

1. $N_1(u, \lambda, \sigma) \equiv \dot{u}_0^T(u-u_0)+\dot{\lambda}_0(\lambda-\lambda_0)-\sigma$ [29].
2. $N_2(u, \lambda, \sigma) \equiv e_i^T(y-y_0)-\sigma$, where $y=(u, \lambda)^T$, e_i is the ith unit vector and the index i is chosen so that the Jacobian of (5) is as well-conditioned as possible [1], [31], [39].
3. $N_3(u, \lambda, \sigma) \equiv \|u-u_0\|^2+|\lambda-\lambda_0|^2-\sigma^2$ [29].
4. $N_\theta(u, \lambda, \sigma, \theta) = \theta\dot{u}_0^T(u-u_0)+(2-\theta)\dot{\lambda}_0(\lambda-\lambda_0)-\sigma=0$, for $0 \leqq \theta \leqq 2$ [29].
5. $N_r(u, \lambda, \sigma, \theta) = \theta\dot{r}_0(r-r_0)+(2-\theta)\dot{\lambda}_0(\lambda-\lambda_0)-\sigma$, where $r \equiv \|u\|$.

N_1 is the usual pseudo arclength parameterization and defines a hyperplane perpendicular to the tangent and at a distance σ along the tangent from (u_0, λ_0). N_2 corresponds to parameter switching. N_3 is similar to N_1 except a sphere is used instead of a hyperplane. N_θ is a modification of N_1. Choosing θ not equal to 1 corresponds to rotating the tangent. This extra degree of freedom sometimes allows an increase in the continuation step near points on the solution curve with high curvature. Note that $\theta=0$ can be used to compute target points for given values of λ. N_r seems to be a new parameterization and is the one used in PLTMGC. It corresponds to N_θ in the space $(\|u\|, \lambda)$. The motivation behind this is that for many PDE-type problems, there are really only two parameters that affect the continuation procedure: some measure of u and λ. Note that with $\theta=0$ and $\theta=2$ both target points in λ and $\|u\|$ can be computed. Thus, this parameterization can be viewed as a hybrid between N_1 and N_2 adapted for PDE type problems. Note that since $\dot{r}r=\dot{u}^Tu$, N_r is ill-defined when $\dot{u}_0^Tu_0=\dot{\lambda}_0=0$. In such situations, which can arise at isolated points (e.g. symmetry breaking bifurcation points), we switch (temporarily) to N_θ.

3.3. Target points. In PLTMGC, the user can specify target values for both $\|u\|$ and λ. Depending on these values, θ and σ are determined for N_r. If only a value for λ, say λ_t, is given, then θ is set to 0 and σ is set to $2\dot{\lambda}_0(\lambda_t-\lambda_0)$ so that $N_r=0$ corresponds exactly to specifying $\lambda=\lambda_t$ (assuming $\dot{\lambda}_0 \neq 0$). Similarly, if only $\|u\|=r_t$ is specified, then θ is set to 2 and σ set to $2\dot{r}_0(r_t-r_0)$. If both target values are specified, then PLTMGC determines heuristically which is easier to reach. It does this by using an approximate local model of the solution curve obtained by interpolating the two most current solutions and their tangents. Points on this approximate solution curve which achieve either of the specified target values are computed and the one which is "closest" in arclength to (u_0, λ_0) determines which target value is to be attempted.

3.4. The predictor. Once θ and σ are determined, the predictor (u_p, λ_p) of (6) is computed by finding scalars (α_p, β_p) which satisfy

$$N_r(u_p, \lambda_p, \sigma, \theta)=0, \tag{21}$$

$$R(u_p, \lambda_p)=0 \tag{22}$$

where $R(u, \lambda)=u^TG(u, \lambda)/u^Tu$. Our use of (21) is motivated by the pseudo-arclength

method, in which one chooses a predictor with $\alpha_p = \beta_p$ and (21) is satisfied [29]. Our use of the Rayleigh quotient (22) was motivated by the linear eigenvalue problem, and by the success of a continuation procedure of Mittelmann and Weber [35] for a certain class of nonlinear elliptic equations. The predictor equations are solved by several iterations of an approximate Newton method. It is easy to insure that (21) is always exactly satisfied after each iteration so that even if this iteration fails to converge (which is rare), the resulting predictor still satisfies one of the two corrector equations.

Our view of (21)-(22) again reflects our view that PDE's of the form (1) are (locally) very low dimensional problems. In some respects, this is analogous to the reduced basis approach of [23], [36], [37] and an approach of Jarausch and Mackens [28]. We also remark that it is usually worthwhile, from a computational point of view, to compute a more accurate predictor, which in this case involves solving two scalar equations, rather than use more corrector iterations, which involve solving much larger sets of equations. Such an investment can also allow much larger steps, if the goal of the continuation procedure is merely to reach a target point.

This new predictor algorithm seems to be very robust and allows rather large continuation steps to be taken (see § 4).

3.5. The corrector iteration. The main task for the corrector iteration is to solve for the solution of the coupled nonlinear system (5) starting with an initial guess provided by the predictor. Since the solution we sought is a regular solution of (5), in principle any regular nonlinear iterative method (e.g. Newton's method, quasi-Newton methods, etc.) can be applied. However, for large and sparse problems, such as discretizations of partial differential equations that we treat here, it is important for efficiency reasons to exploit the structures in G. For a survey of methods used in the corrector iteration, the reader is referred to [16].

Most correctors used in continuation methods, especially for large and sparse systems, are based on Newton's method [16]. In PLTMGC, an approximate Newton method similar to the iteration (15) is used. The damping step $t^{(k)}$ is chosen to force the norm of the residual to decrease. At each corrector iteration, a linear system has to be solved for ∂u and $\partial \lambda$ involving some approximations to the Jacobian of (5). This often constitutes the major part of the overall computational cost. All the linear systems that arise in the corrector iterations are of the form:

$$\begin{pmatrix} A & b \\ c^T & d \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} f \\ g \end{pmatrix} \tag{23}$$

where (A, b, c, d) are approximations to $(G_u, G_\lambda, N_u, N_\lambda)$ respectively. For solving (23), we use the following block-elimination algorithm:

ALGORITHM *BE*: [29]
1. Solve $Av = b$,
   $Aw = f$.
2. Compute $y = (g - c^T w)/(d - c^T v)$.
3. Compute $x = w - yv$.

Note that only a solver for A is needed. On the coarsest grid, a direct solver is used. On the finer meshes, one can use the j-level multi-grid method described in § 2.3. However, as we have shown in [16], [19], Algorithm *BE* may be numerically unstable when A is nearly singular, as is the case near singular points of the solution curves. Moreover, the near-singularity of A may also retard the convergence of the multi-level algorithm [8], [20].

In [19], we proposed using implicit deflation techniques developed in [17], [42] to compute numerically stable representations for the solutions v and w. These deflation techniques can be viewed as working in subspaces orthogonal to approximate null vectors ϕ and ψ of A and are implicit in the sense that they only involve solving systems with A. These deflated decompositions [17] of v and w lead to a numerically stable variant of the *BE* algorithm. (We assume a one-dimensional null space, although our remarks could be modified to deal with the more general situation.)

ALGORITHM *DBE*: Deflated block-elimination algorithm [19].

1. Compute an approximate normalized left singular vector ψ of A.
2. Compute $\phi = \delta A^{-1}\psi$, where $\delta = 1/\|A^{-1}\psi\|$.
3. Compute $c_b = (\psi^T b)$ and $c_f = (\psi^T f)$.
4. Solve $Av_d = b - c_b\psi$ for v_d. (v is represented as: $v = v_d + (c_b/\delta)\phi$)
5. Solve $Aw_d = f - c_f\psi$ for w_d. (w is represented as: $w = w_d + (c_f/\delta)\phi$)
6. Compute $h_1 = g - c^T w_d$, $h_2 = d - c^T v_d$, $h_3 = h_1 c_b - h_2 c_f$, $h_4 = (c^T\phi)c_f - \delta h_1$, $D = (c^T\phi)c_b - \delta h_2$.
7. Compute $y = h_4/D$ and $x = w_d + (h_3\phi - h_4 v_d)/D$.

Note that only two solves with A is need, exactly the same as in Algorithm *BE*. The only overhead involved for performing the deflation is the computation of approximate left and right singular vectors of A which can be accomplished with one or two backsolves with A [16], [17]. Algorithm *DBE* can be proven to be numerically stable [19] independent of the singularity of A.

To apply the deflation techniques on the fine grids, we need to compute the deflated solutions v_d and w_d using a multi-level algorithm. To do this, we have developed a multi-level deflation technique. Let $\phi_j(\psi_j)$ be the left (right) singular functions corresponding to the near zero singular value of b_j. When solving for each of the vectors v_d and w_d, all residuals are systematically purged of their ψ_j components. One saves the decomposition coefficients c_b and c_f on the finest grid only. This procedure can be proven to be convergent and numerically stable independent of the singularity of A [10], [26]. This strategy is similar in spirit, though not in detail, to that of Chan and Keller [20]. We note that the singular functions can be computed with low cost by using a multi-level inverse iteration algorithm [9], [20], [26] and can be reused for several continuation steps. Because of their robustness and negligible overhead, these deflation algorithms are used at all continuation steps and on all grids, without necessitating a check on the singularity of A.

3.6. Failure control for the corrector iteration. An essential part of the corrector algorithm is the failure control. Our philosophy is that when the corrector iteration is having convergence difficulty, we abort the iteration and change the parameterization. In our implementation, we declare that the corrector iteration has failed if either it took too many iterations without convergence or if it took too many damping steps in attempting to force descent of the residuals.

The corrector iteration usually fails only near points on the solution curve with high curvature. When this happens, it means that either the continuation step σ is too large or that no solution to (5) exists for the current values of θ and σ. For example, if a λ target value is specified near a turning point, with a corresponding value of $\theta = 0$, a solution may not exist. When this occurs, PLTMGC does two things: it reduces the value of σ (e.g. halves it) and it uses a value for $\theta(\approx 1)$ forcing N_r to follow the solution curve more closely and deferring the attempt to hit the specified target value on the current step. This process is repeated until convergence is finally achieved which is guaranteed for small enough σ.

3.7. Singular points. In many applications, in addition to tracing the solution branches, one is also interested in locating the singular points themselves, because they are often related to the stability of the solution. Due to their special physical significance, many algorithms have been proposed for determining these singular points accurately [18], [29], [33]. In this paper, we shall only deal with the determination of simple turning points and simple bifurcation points which can be characterized as points on the solution curve where

(24) G_u is singular, with a one-dimensional null space,
G_u is boundedly invertible on its range,

(25) and $G_\lambda \notin \text{Range}\,(G_u)$ for a simple turning point,
and $G_\lambda \in \text{Range}\,(G_u)$ for a simple bifurcation point.

There are two basic issues: detection and accurate location. It can be shown that the determinant of G_u changes sign across simple turning points and simple bifurcation points and therefore it can be used as a detection device on the coarsest grid. In addition, across a turning point, $\dot{\lambda}$ also changes sign and this serves to differentiate between the two types of singular points. For accurate location, there are primarily two classes of methods. The first consists of local iterative algorithms based on an inflated system consisting of $G(u, \lambda) = 0$ augmented by a characterization similar to (24), constructed so that the singular point is a unique and isolated solution of the inflated system. The other class of algorithms consists of methods based on a path tracing continuation method by successively using it to compute points on the solution curve that approach the singular point. We use the second approach in PLTMGC, primarily because many parts of the procedure are already present in the continuation program.

For turning points the method that we use is similar to one proposed by Chan [18] and Keller [30]. It is based on applying the scalar secant method (combined with bisection to ensure convergence) to $\dot{\lambda}(\sigma) = 0$ to determine the correct value of σ to use in N_r so that the continuation procedure will hit the turning point.

For bifurcation points, we use a secant method (again combined with bisection to ensure convergence) on the characterization $\delta(\sigma) = 0$, where δ is the approximation to the smallest singular value computed in step (2) of Algorithm *DBE*. Since the singular value δ does not change sign across a singular point, we make a slight modification in order to speed convergence. We scale the normalized singular functions ψ and ϕ so that $\psi^T\phi$ has the same sign on both sides of the singular point. Then we define δ to be $\psi^T A\phi$ which has the same magnitude as the smallest singular value of A but does change sign across the singular point. This method seems to be new and is very robust and efficient.

3.8. Branch switching at bifurcation points. For a survey of branch switching methods, we refer the reader to Keller [29]. The method that we use is exactly Method I in [29]. Basically, the tangent $(\dot{u}, \dot{\lambda})$ corresponding to the bifurcated branch is computed using the approximate singular functions ψ and ϕ at the bifurcation point and finite difference approximations to the second derivatives of G. This new tangent is then used in the basic continuation procedure to march along the bifurcated branch.

3.9. Refinement strategies. The algorithms presented so far allow the solution curves to be followed on the coarsest grid. However, at occasional points on the solution curve, we may want to compute a solution with higher accuracy than that provided by the discretization on the coarsest grid. PLTMGC allows three options for

refinement onto finer grids, corresponding to different choices of θ in the local parameterization N_r. For all cases, the step length parameter σ is set to zero, and the standard multi-level procedure with adaptive local mesh refinement is used. We have found it adequate to use values of $\dot{r}$, $\dot{\lambda}$, u_0 and λ_0 corresponding to the level 1 solution in the parameterization N_r for all refined levels, although one could certainly update these values as the refinement proceeds.

First, one can choose to refine with a fixed target value λ_t by setting $\theta = 0$. Alternatively, one can use $\theta = 2$ to obtain solutions on all levels at a fixed target value for $\|u\|$. Third, one can refine on the perpendicular hyperplane passing through the current coarse grid solution by choosing $\theta = 1$. Typical situations are illustrated in Fig. 3.1.

In general, the appropriate choice of θ depends on the particular applications and requires user input. Away from singular points, all three strategies can usually be used. Near singular points, however, the situation is quite different. For example, although a solution on the coarsest grid may exist for a particular value λ_t, solutions on the refined grids may not. Here an appropriate strategy is to take $\theta = 1$ (or $\theta = 2$). This is illustrated in Fig. 3.2. Another example is the linear eigenvalue problem, where there are vertical solution branches in the $(\|u\|, \lambda)$ projection and one should refine with $\theta = 2$ or $\theta = 1$ but not $\theta = 0$ (see § 4).

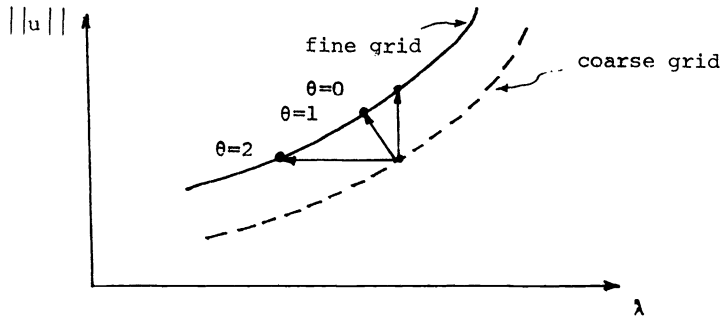


FIG. 3.1. *Refinement away from singular points.*

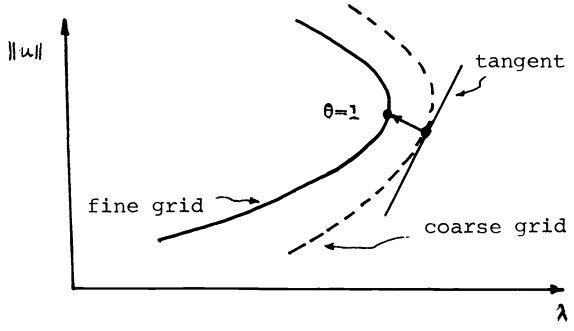


FIG. 3.2. *Refinement near singular points.*

4. Examples of usage. In this section, we shall present a few sample runs with PLTMGC. The goal is to demonstrate the capabilities of PLTMGC rather than to solve complicated problems.

The first two runs are for the standard model problem (Bratu's problem) $\Delta u + \lambda e^u = 0$ with the Dirichlet boundary condition $u = 0$ on a unit square in R^2. This problem

has a simple turning point at $\lambda \approx 6.8$ and $\|u\| \approx 0.7$ (see Fig. 4.1). The level 1 (coarsest) grid for multigrid is shown in Fig. 4.2 and has 25 vertices, only 9 of which are unknowns. The first run is for demonstrating the λ-target and refinement capabilities of PLTMGC.

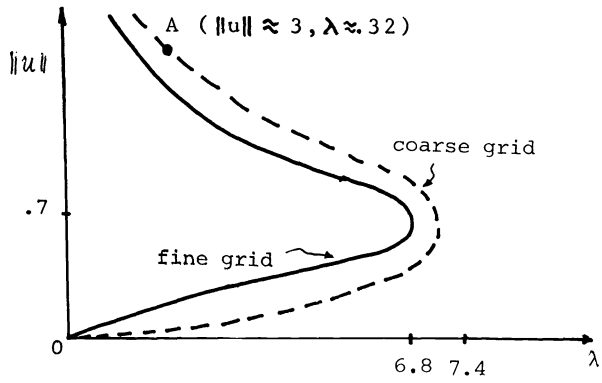


FIG. 4.1. *Solution branches of Bratu's problem*: $\Delta u + \lambda e^u = 0$.

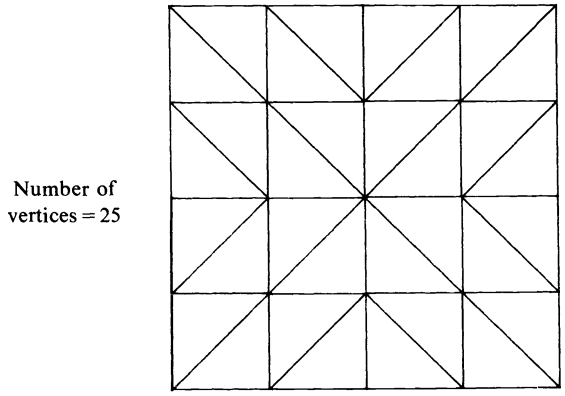


FIG. 4.2. *Coarsest grid for Bratu's problem*: $\Delta u + \lambda e^u = 0$.

Starting from the trivial solution ($u = 0, \lambda = 0$), PLTMGC managed to get to the target point $\lambda = 6$ in one step with 4 corrector iterations (see Table 4.1). Next we ask PLTMGC to refine the solution to 3 multigrid levels with $\theta = 2$ (i.e. constant $\|u\|$ level). The two extra grids are generated adaptively using the mesh refinement strategies discussed in § 2.4 and are shown in Fig. 4.3. As part of the adaptive mesh refinement procedure, the accuracy of the solutions is also generated. To demonstrate the different refinement possibilities, refinements using $\theta = 0$ (i.e. constant λ) and $\theta = 1$ (i.e. perpendicular hyperplane) are also generated from the same coarse grid solution. Notice that the number of corrector iterations on each level is quite small—no more than 3 iterations are needed.

The next run (see Table 4.2) demonstrates the $\|u\|$ target and turning point capabilities. Starting from the $\lambda = 6$ solution generated from the first run, a target value of $\|u\| = 3$ is specified. Even though the target point is beyond the turning point on the upper branch and quite far away from it (point A in Fig. 4.1), convergence is achieved in only one step with 4 corrector iterations. This robustness is primarily due to the extremely good predictor. Moreover, the turning point is detected and located

TABLE 4.1

Bratu's problem from $\lambda = 0$ to $\lambda = 6$ with refinement. Legend. L, level number of grid; NI, number of corrector iterations; DIGITS, number of correct digits in the solution.

L	NI	λ	$\|u\|$	$\dot{\lambda}$	$\dot{r}$	det (G_u)	δ
Continue to target point with $\lambda = 6$:							
1	1	0.000 E+00	0.000 E+00	0.999 E+00	0.000 E+00	0.411 E9	0.117 E+01
1	4	0.600 E+01	0.358 E+00	0.994 E+00	0.111 E+00	0.504 E8	0.630 E+00
Refine to level 3 using $\theta = 2$ (constant $\|u\|$):							
2	3	0.567 E+01	0.358 E+00	0.993 E+00	0.118 E+00	0.595 E8	0.167 E+00
3	3	0.557 E+01	0.358 E+00	0.993 E+00	0.119 E+00	0.626 E8	0.513 E−01

Information about accuracy of the solutions:

NORM	*SOLN*	*ERROR*	*DIGITS*
H1	0.165 E+01	0.176 E+00	0.971
L2	0.357 E+00	0.605 E−02	1.771
MAX	0.682 E+00	0.139 E−01	1.691

L	NI	λ	$\|u\|$	$\dot{\lambda}$	$\dot{r}$	det (G_u)	δ
Refine using $\theta = 0$ (constant λ):							
2	3	0.600 E+01	0.400 E+00	0.990 E+00	0.139 E+00	0.478 E8	0.148 E+00
3	2	0.600 E+01	0.416 E+00	0.988 E+00	0.152 E+00	0.468 E8	0.433 E−01
Refine using $\theta = 1$ (perpendicular hyperplane):							
2	3	0.600 E+01	0.399 E+00	0.990 E+00	0.139 E+00	0.480 E8	0.149 E+00
3	2	0.599 E+01	0.415 E+00	0.989 E+00	0.151 E+00	0.470 E8	0.434 E−01

TABLE 4.2

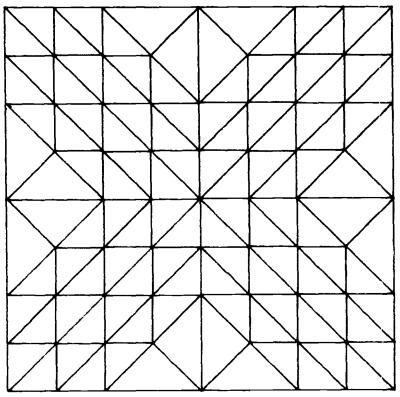
Bratu's problem near turning point.

L	NI	λ	$\|u\|$	$\dot{\lambda}$	$\dot{r}$	det (G_u)	δ
From $\lambda = 6$ to target value $\|u\| = 3$:							
1	4	0.600 E+01	0.358 E+00	0.994 E+00	0.111 E+00	0.504 E8	0.630 E+00
1	4	0.319 E+00	0.300 E+01	−0.550 E+00	0.832 E+00	−0.688 E9	0.246 E+01
Secant method for turning point:							
1	4	0.199 E+01	0.206 E+01	−0.959 E+00	0.281 E+00	−0.136 E9	0.213 E+01
1	4	0.596 E+01	0.121 E+01	−0.978 E+00	0.205 E+00	−0.195 E8	−0.118 E+01
1	3	0.737 E+01	0.783 E+00	−0.744 E+00	0.667 E+00	−0.381 E7	−0.140 E+00
1	3	0.722 E+01	0.570 E+00	0.944 E+00	0.330 E+00	0.115 E8	0.267 E+00
1	3	0.740 E+01	0.689 E+00	0.381 E+00	0.924 E+00	0.148 E7	0.454 E−01
1	3	0.738 E+01	0.770 E+00	−0.675 E+00	0.736 E+00	−0.315 E7	−0.113 E+00
1	3	0.741 E+01	0.718 E+00	−0.947 E−01	0.994 E+00	−0.336 E6	−0.109 E−01
1	3	0.741 E+01	0.710 E+00	0.481 E−01	0.998 E+00	0.171 E6	0.546 E−02
1	3	0.741 E+01	0.713 E+00	−0.412 E−03	0.999 E+00	−0.145 E4	−0.469 E−04
1	3	0.741 E+01	0.713 E+00	0.979 E−06	0.999 E+00	0.796 E1	0.112 E−06

accurately by the secant method on λ as described in § 3.7. Notice that convergence is superlinear as the turning point is approached.

The third run (see Table 4.3) demonstrates the location of simple bifurcation points and branch switching. The problem is the linear eigenvalue problem: $-\Delta u = \lambda u$

LEVEL 2
Number of
vertices = 69



LEVEL 3
Number of
vertices = 205

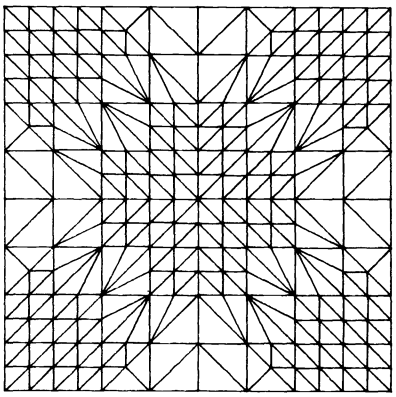


FIG. 4.3. *Fine grids for Bratu's problem*: $\Delta u + \lambda e^u = 0$.

with the Dirichlet boundary condition $u = 0$ on the unit square in R^2. The continuous problem has bifurcation points at the eigenvalues of the Laplacian $-\Delta$, i.e. $(n\pi)^2 + (m\pi)^2$ where n and m run through the natural numbers. Thus the first simple bifurcation is near $\lambda = 2\pi^2$ whereas the next simple bifurcation is near $\lambda = 8\pi^2$. The second eigenvalue of $-\Delta$ near $5\pi^2$ is a multiple one and does not correspond to a simple bifurcation. We use PLTMGC to locate the second simple bifurcation point. The coarsest grid used is illustrated in Fig. 4.4 which is fine enough to resolve this eigenvalue. Starting from the trivial solution ($u = 0$, $\lambda = 0$), the target value of $\lambda = 25$ is reached in one step. Notice that the sign of the determinant has changed because we passed the first bifurcation. Next we continue to the target value of $\lambda = 80$. This is past the multiple eigenvalue near $\lambda = 5\pi^2$ and notice that the determinant does not change sign. At this point the bifurcation detection mechanism is switched on and the target value of $\lambda = 100$ is specified, which is beyond the second simple bifurcation point. Since the determinant changes sign this time, the bifurcation is detected and the secant method for δ described in § 3.7 finds the bifurcation point in 4 steps. Then we switch branches and continue to the target value of $\|u\| = 1$. The solution u here is the normalized eigenfunction on the coarsest grid. To get better accuracy, we refine to 3 multigrid levels using $\theta = 2$ (constant $\|u\|$). The grids generated are given in Fig. 4.5. Notice that the eigenvalue $\lambda = 79.7$ on the finest grid is a rather good approximation to the true value of $\lambda = 8\pi^2$. A plot of the eigenfunction is given in Fig. 4.6.

TABLE 4.3
Linear eigenvalue problem: $-\Delta u = \lambda u$.

L	NI	λ	$\|u\|$	$\dot{\lambda}$	$\dot{r}$	det (G_u)	δ
Continue from $\lambda = 0$ *to* $\lambda = 25$ *then to* $\lambda = 80$:							
1	1	0.000 E+00	0.000 E+00	0.100 E+01	0.000 E+00	0.339 E35	0.304 E+00
1	1	0.250 E+02	0.000 E+00	0.100 E+01	0.000 E+00	−0.206 E32	−0.711 E−01
1	1	0.800 E+02	0.000 E+00	0.100 E+01	0.000 E+00	−0.329 E27	0.468 E+00
Continue to $\lambda = 100$, *check sign of determinant*:							
1	1	0.100 E+03	0.000 E+00	0.100 E+01	0.000 E+00	0.231 E25	0.174 E+00
Determinant changes sign, find bifurcation point by secant method on δ:							
1	1	0.900 E+02	0.000 E+00	0.100 E+01	0.000 E+00	−0.599 E25	0.188 E−01
1	1	0.950 E+02	0.000 E+00	0.100 E+01	0.000 E+00	0.423 E25	−0.453 E−01
1	1	0.915 E+02	0.000 E+00	0.100 E+01	0.000 E+00	−0.119 E21	0.884 E−06
1	1	0.915 E+02	0.000 E+00	0.100 E+01	0.000 E+00	0.124 E22	−0.591 E−05
Switch branch:							
1	0	0.915 E+02	0.000 E+00	0.000 E+00	0.100 E+01	0.124 E22	−0.591 E−05
Continue to target value $\|u\| = 1$:							
1	1	0.915 E+02	0.100 E+01	0.348 E−04	0.100 E+01	−0.126 E21	0.446 E−06
Refine to 3 *multigrid levels with* $\theta = 2$ (*constant* $\|u\|$):							
2	3	0.813 E+02	0.100 E+01	0.196 E−03	0.100 E+01	−0.234 E27	0.402 E−06
3	3	0.797 E+02	0.100 E+01	0.248 E−03	0.100 E+01	−0.354 E27	0.263 E−06

A final example is for the problem:

$$-\Delta u + 10(u - \lambda e^u) = 0$$

on the unit square in R^2 with homogeneous Neumann boundary conditions. This problem has more complicated behaviour than the other examples and illustrates the branch-switching capabilities of PLTMGC. There is a main branch of constant solutions satisfying the scalar equation

$$u = \lambda e^u.$$

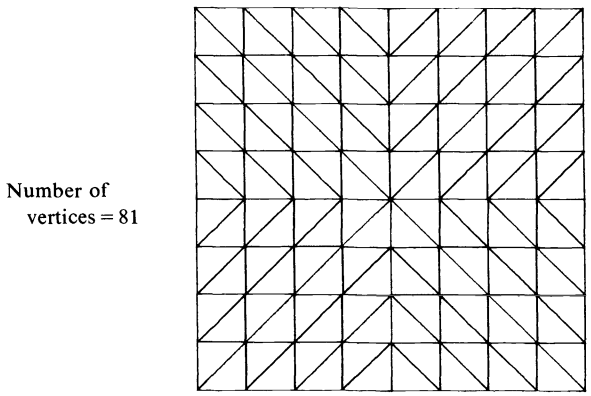


FIG. 4.4. *Coarsest grid for* $-\Delta u = \lambda u$.

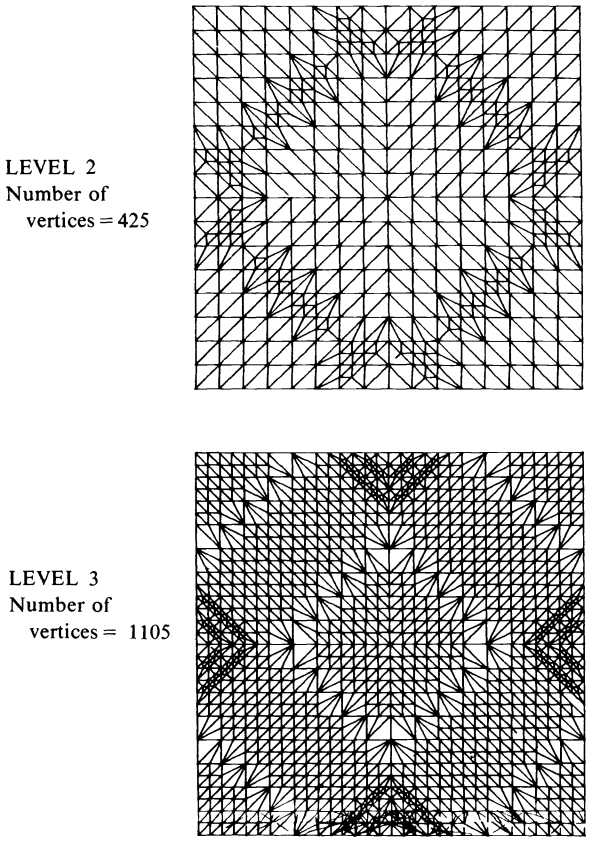


FIG. 4.5. *Find grids for* $-\Delta u = \lambda u$.

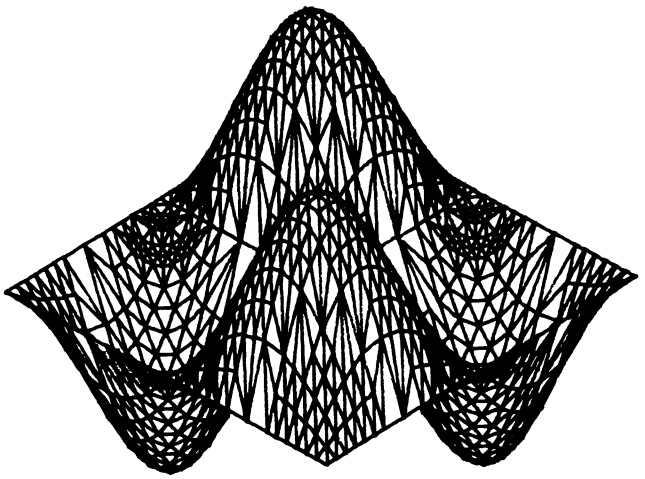


FIG. 4.6. *4th eigenfunction of* $-\Delta u = \lambda u$.

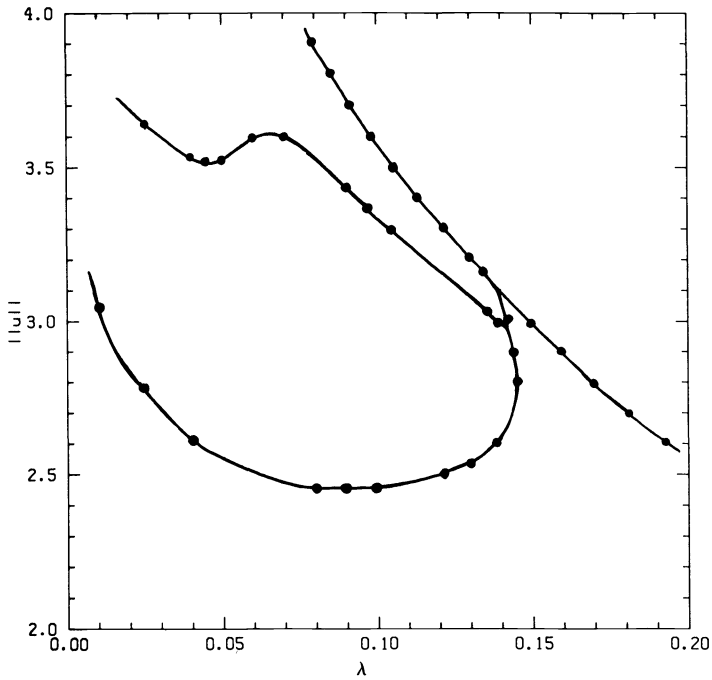


FIG. 4.7. *Bifurcation diagram for* $-\Delta u + 10(u - \lambda e^u) = 0$.

On the upper part of this main branch, there is a symmetry breaking bifurcation point. Moreover, on this secondary branch there is another symmetry breaking bifurcation very close to the first bifurcation point. This part of the bifurcation diagram is shown in Fig. 4.7 for the 5 by 5 grid shown in Fig. 4.2 where the continuation points used by PLTMGC are also shown.

5. Extensions. Although the current version of PLTMGC is quite complete in terms of its capabilities, we feel there are still some extensions that would enhance the usability of the package.

First, it is possible that one may want to locate singular points accurately on the finer meshes. This would appear to require a nontrivial computation on each level, and could involve several strategies, e.g. varying θ in N_r in an attempt to "hit" the limit point, or perhaps the application of the bisection/secant method to $\dot{\lambda}(\sigma) = 0$ or $\delta(\sigma) = 0$ directly on the finer meshes. This extension should be straightforward.

In our experience, the overall efficiency and robustness of the continuation procedure depends critically on the predictor. In fact, it is a little dissatisfying that the current predictor, while surprisingly efficient and robust, is rather ad hoc. Our previous attempts to provide a predictor/step picker with more theoretically justification [22] did not seem to perform as well for PDE type problems.

Lastly, PLTMGC can only handle simple singular points. Extensions to higher order singularities like multiple bifurcation, cusps and following paths of singular points seem to be natural.

REFERENCES

[1] J. P. ABBOTT, *An efficient algorithm for the determination of certain bifurcation points*, J. Comp. Appl. Math., 4 (1978), pp. 19-27.

[2] E. ALLGOWER AND K. GEORG, *Simplicial and continuation methods for approximating fixed points and solutions to systems of equations*, SIAM Rev., 22(1) (1980), pp. 28-85.
[3] H. AMANN, *Fixed point equations and nonlinear eigenvalue problems in ordered Banach spaces*, SIAM Rev., 18 (1976), pp. 620-709.
[4] I. BABUSKA AND W. C. RHEINBOLDT, *A posteriori error estimates for the finite element method*, Int. J. Numer. Meth. Eng., 12 (1975), pp. 1597-1615.
[5] ———, *Error estimates for adaptive finite element computations*, SIAM J. Numer. Anal., 15 (1978), pp. 736-754.
[6] ———, *Reliable error estimation and mesh adaptation for the finite element method*, in Computational Methods in Nonlinear Mechanics, J. T. Oden, ed., North-Holland, New York, 1980, pp. 67-108.
[7] R. E. BANK, PLTMG *User's Guide, June*, 1981 *version*, Technical Report, Dept. of Mathematics, University of Calif. at San Diego, 1982.
[8] ———, *A comparison of two multi-level iterative methods for nonsymmetric and indefinite elliptic finite element equations*, SIAM J. Numer. Anal., 18 (1981), pp. 724-743.
[9] ———, *Analysis of a multilevel inverse iteration procedure for eigenvalue problems*, SIAM J. Numer. Anal., 19 (1982), pp. 886-898.
[10] R. E. BANK AND T. F. CHAN, *Multi-grid deflation*, 1984, in preparation.
[11] R. E. BANK AND T. F. DUPONT, *An optimal order process for solving finite element equations*, Math. Comp., 36 (1981), pp. 35-51.
[12] R. E. BANK AND D. J. ROSE, *Global approximate Newton methods*, Numer. Math., 37 (1981), pp. 279-295.
[13] ———, *Analysis of a multilevel iterative method for nonlinear finite element equations*, Math. Comp., 39 (1982), pp. 453-465.
[14] R. E. BANK AND A. WEISER, *A posteriori error estimates in the finite element method*, 1983, submitted to Math. Comp.
[15] A. BRANDT, *Multi-level adaptive solution to boundary value problems*, Math. Comp., 31 (1977), pp. 333-390.
[16] T. F. CHAN, *Techniques for large sparse systems arising from continuation methods*, in Numerical Methods for Bifurcation Problems, T. Kupper, H. Mittelmann and H. Weber, eds., International Series of Numerical Math., Vol. 70, Birkhauser Verlag, Basel, 1984, pp. 116-128.
[17] ———, *Deflated decomposition of solutions of nearly singular systems*, SIAM J. Numer. Anal., 21 (1984), pp. 738-754.
[18] ———, *Newton-like pseudo-arclength methods for computing simple turning points*, this Journal, 5 (1984), pp. 135-148.
[19] ———, *Deflation techniques and block-elimination algorithms for solving bordered singular systems*, this Journal, 5 (1984), pp. 121-134.
[20] T. F. CHAN AND H. B. KELLER, *Arclength continuation and multi-grid techniques for nonlinear eigenvalue problems*, this Journal, 3 (1982), pp. 173-194.
[21] R. S. DEMBO, S. EISENSTAT AND T. STEIHAUG, *Inexact Newton methods*, SIAM J. Numer. Anal., 18 (1982), pp. 400-408.
[22] C. DEN HEIJER AND W. C. RHEINBOLDT, *On steplength algorithms for a class of continuation methods*, SIAM J. Numer. Anal., 18 (1981), pp. 925-947.
[23] J. P. FINK AND W. C. RHEINBOLDT, *On the error behavior of the reduced basis technique for nonlinear finite element approximations*, Technical Report ICMA-82-37, Institute for Computational Mathematics and Applications, Dept. of Mathematics and Statistics, Univ. Pittsburgh, Pittsburgh, 1982.
[24] C. B. GARCIA AND W. I. ZANGWILL, *Pathways to Solutions, Fixed Points and Equilibria*, Prentice-Hall, Englewood Cliffs, NJ, 1981.
[25] W. HACKBUSCH, *On the fast solution of nonlinear elliptic equations*, Numer. Math., 32 (1979), pp. 83-95.
[26] ———, *On the computation of approximate eigenvalues and eigenfunctions of elliptic operators by means of a multi-grid method*, SIAM J. Numer. Anal., 16 (1979), pp. 201-215.
[27] ———, *Multi-grid solution of continuation problems*, in Iterative Solution of Nonlinear Systems, R. Ansorge, T. Meis and W. Torning, eds., Springer-Verlag, Berlin, 1982.
[28] H. JARAUSCH AND W. MACKENS, *Computing solution branches by use of a condensed Newton-supported Picard iteration scheme*, Technical Report, Institute for Geometrie und Praktische Mathematik, Rheinisch-Westfalische Technische Hochschule, Aachen, 1983.
[29] H. B. KELLER, *Numerical solution of bifurcation and nonlinear eigenvalue problems*, in Applications of Bifurcation Theory, P. Rabinowitz, ed., Academic Press, New York, 1977, pp. 359-384.
[30] ———, *Global homotopies and Newton methods*, in Recent Advances in Numerical Analysis, Carl de Boor and Gene Golub, eds., Academic Press, New York, 1978, pp. 73-94.

[31] M. KUBICEK, *Dependence of solution of nonlinear systems on a parameter*, ACM Trans. Math. Software, 2 (1976), pp. 98-107.

[32] LOIS MANSFIELD, *On the solution of nonlinear finite element systems*, SIAM J. Numer. Anal., 17 (1980), pp. 752-765.

[33] R. G. MELHEM AND W. C. RHEINBOLDT, *A comparison of methods for determining turning points of nonlinear equations*, Computing, 29 (1982), pp. 201-226.

[34] H. D. MITTELMANN AND H. WEBER, *Numerical methods for bifurcation problems—A survey and classification*, in Bifurcation Problems and Their Numerical Solution, H. D. Mittelmann and H. Weber, eds., Birkhauser, Dortmund, 1980, pp. 1-45.

[35] ———, *Multi-grid solution of bifurcation problems*, Technical Report 65, Abteiling Math., Univ. Dortmund, 1983, this Journal, 6 (1985), pp. 49-60.

[36] A. K. NOOR, *Recent advances in reduction methods for nonlinear problems*, Computers and Structures, 13 (1981), pp. 31-44.

[37] A. K. NOOR AND G. M. PETERS, *Reduced basis technique for nonlinear analysis of structures*, AIAA J., 18 (1980), pp. 455-462.

[38] W. C. RHEINBOLDT, *Numerical methods for a class of finite dimensional bifurcation problems*, SIAM J. Numer. Anal., 15(1) (1978), pp. 1-11.

[39] ———, *Solution fields of nonlinear equations and continuation methods*, SIAM J. Numer. Anal., 17 (1980), pp. 221-237.

[40] ———, *Numerical analysis of continuation methods for nonlinear structural problems*, Computers and Structures, 13 (1981), pp. 103-113.

[41] W. C. RHEINBOLDT AND J. V. BURKARDT, *A locally parameterized continuation process*, ACM Trans. Math. Software, 9(2) (June 1983), pp. 215-235.

[42] G. W. STEWART, *On the implicit deflation of nearly singular systems of linear equations*, this Journal, 2 (1981), pp. 136-140.

[43] L. T. WATSON AND D. FENNER, *Chow-Yorke algorithm for fixed points or zeros of C^2 maps*, ACM Trans. Math. Software, 6 (1980), pp. 252-259.

[44] A. WEISER, *Local-mesh, local-order, adaptive finite element methods with a posteriori error estimates for elliptic partial differential equations*, Ph.D. thesis, Yale Univ., New Haven, CT, 1982, Techreport #213.

SIAM J. SCI. STAT. COMPUT.
Vol. 7, No, 2, April 1986

014

NUMERICAL SOLUTIONS FOR FLOW IN A PARTIALLY FILLED, ROTATING CYLINDER*

YIH-YIH LIN†

Abstract. An efficient least-squares method with finite elements is used to simulate the viscous flow with surface tension in a partially liquid-filled, horizontal, rotating cylinder. Mixed interpolations are used for the method: the free boundary is interpolated by periodic cubic splines, the velocity by nine-node isoparametric biquadratic elements, and the pressure by four-node isoparametric bilinear elements. The free boundary is located by minimizing the sum of squares of the boundary residuals. Advantages of the method over the often used collocation method are shown. Numerical solutions are validated to some extent by comparing with results from perturbation methods and physical experiments.

Key words. free boundary, least squares, finite elements, periodic cubic splines

AMS(MOS) subject classifications. 76, 65

1. Introduction. Flow in a partially filled, horizontal, rotating cylinder occurs in various engineering applications, such as cream separators, liquid degassers, and coating operation for pipes and tubes. The knowledge of the critical rotational speed for stable flow, the shape of the interface, and the pressure and the velocity of the liquid is a valuable aid to the designing engineer. In addition to its occurrence in practical applications, this flow provides a simple, but nontrivial example in the study of the complex free-boundary flow phenomena.

Phillips [8], Karweit and Corrsin [5], Greenspan [2], and Whiting [10] made observations and experiments on this flow. Phillips [8], Greenspan [2], Ruschak and Scriven [9], and Gans [1] did theoretical investigations on this flow, using perturbation analyses. It is clear from Whiting's experiments [10] that accurate measurements on this flow are difficult and expensive to obtain. The results of theoretical investigation are valid only for the limiting cases in the perturbation analysis. Therefore, numerical simulations provide a useful alternative to study this flow problem.

Orr and Scriven [7] made a first and successful attempt on the numerical simulation of this flow. In their paper [7], same-order interpolations of cubic polynomials were used for flow fields and the free boundary; the collocation method with residuals computed from the normal-stress boundary condition, rather than the tangential-velocity boundary condition, was used to locate the free boundary; the nodal points were numbered in their natural order, and the arisen circulant finite-element matrix was solved by the ordinary Gaussian elimination; and a mesh of 12 elements angularly and 1 element radially was used with a CDC Cyber 74 main-frame computer. Although the collocation method is convenient and often used for locating free boundaries, it has some drawbacks: (1) The method cannot tell how to choose good sampling points on the free boundary; the choice of Gaussian points as collocation points in Orr and Scriven [7] was based on experience. (2) The method, requiring equal numbers of unknowns and equations, necessarily forces an undertermined pressure constant in the finite-element method to be associated with the constant-volume constraint, as in Orr and Scriven [7]; such an association seems not to have a physical basis.

In this paper we present an efficient least-squares method with finite elements of mixed-order interpolations to simulate the flow. We present computational experiments

* Received by the editors July 24, 1984, and in final form January 8, 1985. This work was partially supported by the National Science Foundation under grant MCS 77-26732.

† Gearhart Industries, Inc., 2525 Wallingwood, Austin, Texas 78746.

with Reynolds numbers as high as 1500 and with meshes as fine as 12 elements angularly and 6 elements radially even though a minicomputer, the VAX 11/780, has been used. We show that the choice of sampling points for locating the free boundary and the determination of the pressure constant in the finite-element method follows naturally from the least-squares method. We present numerical results that indicate the tangential-velocity condition is as good as the normal-stress condition for locating the free boundary. In addition, we compare our numerical solutions with solutions from both physical experiments and a combined method of perturbation analysis and numerical analysis to validate the numerical simulations.

2. Description of the flow. The steady, two-dimensional, and incompressible flow of a Newtonian liquid is considered. An infinitely long, hollow cylinder of radius R_c partially filled with a liquid, of density ρ and viscosity μ, is spun about its axis with constant rotational velocity, as shown in Fig. 1. The liquid is distributed over the solid cylinder as a layer of thickness $R_c - F$, whose free boundary F has a surface tension σ. Inside the layer is a core of air at a uniform pressure P_0. Under the action of gravity, with acceleration constant $\mathbf{G}$, the thickness of the layer varies around the cylinder.

The governing equations for the flow are the Navier-Stokes equations and the continuity equation:

$$\nabla \cdot (\rho \mathbf{V}\mathbf{V} + \mathbf{T}) - \rho \mathbf{G} = 0, \qquad \nabla \cdot \mathbf{V} = 0, \tag{1.1}$$

where $\mathbf{V}$ is the velocity, and $\mathbf{T}$ is the stress tensor defined as

$$\mathbf{T} = P\mathbf{I} - \mu[\nabla \mathbf{V} + (\nabla \mathbf{V})^T],$$

where P is the pressure, and $\mathbf{I}$ is the unit tensor. The flow domain is $\{(R, \theta) \mid F \leqq R \leqq R_c, 0 \leqq \theta \leqq 2\pi\}$. On the solid boundary the no-slip boundary condition is satisfied; on the free boundary the traction condition and the kinematic condition are satisfied. The problem is nondimensionalized by

$$r = R/R_c, \quad \mathbf{v} = \mathbf{V}/\omega R_c, \quad p = (P - P_0)/\rho\omega^2 R_c^2, \quad f = F/R_c.$$

The three dimensionless parameters are the Reynolds number Re, the Weber number s, and the dimensionless gravity g defined as

$$\mathrm{Re} = \rho\omega R_c^2/\mu, \quad s = \sigma/\rho\omega^2 R_c^2, \quad g = G/\omega^2 R_c.$$

The dimensionless stress $\boldsymbol{\tau}$ is $P\mathbf{I} - [\nabla \mathbf{v} + (\nabla \mathbf{v})^T]/\mathrm{Re}$. Let $\mathbf{j}$ be the downward vertical unit vector, and $\mathbf{n}$, $\mathbf{t}$, and κ be the unit normal, the unit tangent, and the curvature of the free boundary, respectively. Also, let $\mathbf{e}_r$ and $\mathbf{e}_\theta$ be the radial and the angular unit vectors in the cylindrical polar coordinate system. The flow problem can then be put in the following form:

Problem 1. Find $f(\theta)$, $\mathbf{v}(r, \theta)$ and $p(r, \theta)$ such that

$$\nabla \cdot (\mathbf{v}\mathbf{v} + \tau) + g\mathbf{j} = 0 \quad \text{in } f(\theta) \leqq r \leqq 1, \tag{1.2}$$

$$\nabla \cdot \mathbf{v} = 0 \quad \text{in } f(\theta) \leqq r \leqq 1, \tag{1.3}$$

$$\mathbf{v} = \mathbf{e}_\theta \quad \text{on } r = 1, \tag{1.4}$$

$$\mathbf{n} \cdot \mathbf{v} = 0 \quad \text{on } r = f(\theta), \tag{1.5}$$

$$\mathbf{t}\mathbf{n} : \tau = 0 \quad \text{on } r = f(\theta), \tag{1.6}$$

$$\mathbf{n}\mathbf{n} : \tau + \kappa s = 0 \quad \text{on } r = f(\theta), \tag{1.7}$$

$$\int_0^{2\pi} \int_{f(\theta)}^{1} r\,dr\,d\theta - \pi(1-f_0^2) = 0, \tag{1.8}$$

where f_0 is the average radius of the free boundary.

Equations (1.2) and (1.3) are the dimensionless governing equations. Condition (1.4) is the no-slip boundary condition on the solid boundary. Condition (1.5) is the kinematic condition, and conditions (1.6) and (1.7) are the traction conditions on the free boundary. Equation (1.8) is the constant-volume constraint arising from the incompressibility of the liquid.

3. The least-squares formulation for locating the free boundary. We will reformulate Problem 1 as a least-squares problem. The residuals used in the least-squares formulation can be computed from any of the three free boundary conditions (1.5), (1.6), and (1.7). In order to simplify the study, we will use boundary conditions (1.5) and (1.7) but not (1.6) to compute the residuals.

Let $R_s(\theta; f, \mathbf{v}, p)$ and $H(f)$ denote the left-hand sides of (1.7) and (1.8) respectively. Problem 1 can be restated as the following problem:

Problem 2. Find f, $\mathbf{v}$, and p such that

$$R_s(\theta; f, \mathbf{v}, p) = 0 \quad \text{for } 0 \leqq \theta \leqq 2\pi,$$
$$H(f) = 0,$$

where $\mathbf{v}$ and p satisfy equations (1.2) and (1.3) and boundary conditions (1.4), (1.5), and (1.6).

Clearly, Problem 2 is mathematically equivalent to the following problem:

Problem 3. Find f, $\mathbf{v}$, and p such that

$$\int_0^{2\pi} R_s^2(\theta; f, \mathbf{v}, p)\,d\theta = 0 \quad \text{for } 0 \leqq \theta \leqq 2\pi,$$
$$H(f) = 0,$$

where $\mathbf{v}$ and p satisfy equations (1.2) and (1.3) and boundary conditions (1.4), (1.5), and (1.6).

By a quadrature rule the integral in Problem 3 can be approximated by $\sum_{i=1}^{M} w_i R_s^2(\theta_i; f, \mathbf{v}, p)$. Let the free boundary be approximated by the n-parameter family of curves $F(\mathbf{y})$, $\mathbf{y} = (y_1, \cdots, y_n)$. Then, Problem 3 can be solved numerically as a weighted least-squares problem with a constraint. Replacing the constraint by a penalty function, we can approximate Problem 3 numerically by the following problem:

Problem 4. Find $\mathbf{y}$ such that the following sum of squares is minimized:

$$I_s = \sum_{i=1}^{M} w_i R_s^2(\theta; \mathbf{y}, \mathbf{v}, p) + kH^2(\mathbf{y}),$$

where k is a positive number, and where $\mathbf{v}$ and p satisfy equations (1.2) and (1.3) and boundary conditions (1.4), (1.5), and (1.6).

Problem 1 has been approximated by Problem 4, which is a least-squares problem. Similarly, let $R_v(\theta; f, \mathbf{v}, p)$ denote the left-hand sides of (1.5). Then, Problem 1 can also be restated as the following problem:

Problem 5. Find $\mathbf{y}$ such that the following sum of squares is minimized:

$$I_v = \sum_{i=1}^{M} w_i R_v^2(\theta; \mathbf{y}, \mathbf{v}, p) + kH^2(\mathbf{y}),$$

where k is a positive number, and where $\mathbf{v}$ and p satisfy equations (1.2) and (1.3) and boundary conditions (1.4), (1.6), and (1.7).

Problem 4 is the normal-stress-residual scheme, and Problem 5 is the tangential-velocity-residual scheme. In the present computation, the 3-point Gaussian quadrature rule was used to approximate the integral; the classical Gauss–Newton method was used to solve the nonlinear least-squares Problems 4 and 5. This least-squares formulation has some advantages over the previous collocation formulation of Orr and Scriven [7]:

(1) The problem of deciding the pressure constant in the finite-element method is readily solved. To solve Problem 4 or 5 numerically, a series of fixed-boundary-value problems for the flow field must be solved. Because the essential boundary condition (1.7) is deleted from the fixed-boundary-value problems in Problem 4, the pressure is determined only up to a constant when these problems are solved. To be consistent with the least-squares formulatation, we propose to determine the pressure constant as the one that minimizes the sum of squares I_s. Since the sum of squares I_s is a quadratic function of p, this pressure constant is easily computed.

(2) The choice of good sampling points, where residuals are computed, is apparent. In the present formulation, the sampling points are naturally determined by the quadrature rule used. Moreover, the better the quadrature approximates the integral, the better the present formulation approximates Problem 1. Here the Gaussian points are obviously a good choice as they lead to a good approximation of the integral.

4. Solutions to fixed-boundary-value problems and representation of the free boundary. To solve Problem 4, one needs to solve a series of fixed-boundary-value problems governed by the full steady-state Navier–Stokes equations. They can be solved by either the finite-element method or the finite-difference method. The finite-element method was used since it was easier to program the finite-element method than to program the finite-difference method for a series of such problems with varying domains.

The Galerkin finite-element method was used with the primitive velocity-pressure formulation. The well-known mixed interpolation elements of Hood and Taylor [3] for hydrodynamic problems was used. These elements are conforming isoparametric quadrilaterals with nine-node biquadratic interpolation for the velocity and conforming isoparametric quadrilaterals with four-node interpolation for the pressure. The weighted residual integrals resulting from the Galerkin formulation were evaluated numerically by the 3 by 3 Gaussian rule. The Newton–Raphson method was used to solve the nonlinear equations arising from the weighted residual integrals. The continuity of such elements is one-order higher than the required minimal C^0 continuity in the weighted residual integrals.

Hood and Taylor [3] have shown that a mixed-order interpolation is at least as good as a same-order interpolation. Orr and Scriven [7] used a same-order interpolation with cubic Hermite functions to interpolate both the velocity and the pressure. By doing so, they probably used more storage than the method used here without gaining more accuracy in solving the fixed boundary problem.

Because of the circular structure of the flow domain, a straightforward method of numbering the finite-element nodes, as in Orr and Scriven [7], generates a circulant finite-element Jacobian matrix. The application of the ordinary Gaussian elimination to invert such a circulant matrix requires full-matrix storage. One way to avoid such full-matrix storage requirement is to use Iron's [4] frontal solution. Instead, we made the Jacobian matrix to be a true band matrix by dividing the domain uniformly in the θ direction and numbering the nodes along the radius up and down as shown in

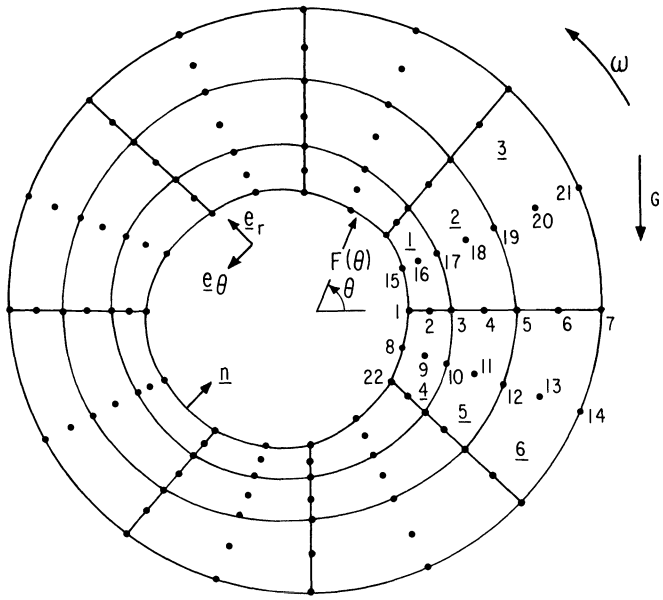


FIG. 1. *Flow in a horizontal rotating cylinder. A finite-element mesh is sketched.*

Fig. 1. This numbering method allowed us to apply an ordinary band solver to efficiently solve the finite-element matrix.

The weighted residual integrals involve both the governing equations and the boundary conditions. Since the boundary condition (1.7) involves the curvature of the free boundary, any valid free boundary representation must be continuous at least up to the first derivative. For consistency with the elements which were used, whose continuity is one-order higher than the minimal requirement, we used the periodic cubic spline functions, whose continuity is one-order higher than the required C^1 continuity, to represent the free boundary. In analogy with Hood and Taylor's [3] reasoning, this consistency of orders of continuity among the elements and the free-boundary representation should be a prerequisite for any efficient numerical scheme.

5. A combined method of perturbation analysis and numerical analysis. When the gravity parameter is small and the Reynolds number is low, a combination of perturbation analysis and numerical analysis can be used to generate numerical solutions for Problem 1. Solutions from such a combined method can then be compared against the solutions from the least-squares method. The combined method is described below.

When g is zero, Problem 1 admits a simple solution, known as the rigid-body rotation solution:

$$f(\theta) = f_0,$$
$$\mathbf{v} = r\mathbf{e}_\theta, \tag{5.1}$$
$$p = (r^2 - f_0^2)/2 - s/f_0.$$

Let the r and the θ components of the velocity $\mathbf{v}$ be u and v. From (5.1) the solution to Problem 1 with small gravity is

$$u = g\bar{u} + O(g^2), \tag{5.2}$$

$$v = r + g\bar{v} + O(g^2), \tag{5.3}$$

$$p=(r^2-f_0^2)/2-s/f_0+g\bar{p}+O(g^2), \tag{5.4}$$

$$f=f_0+g\bar{f}+O(g^2), \tag{5.5}$$

where $\bar{u}$, $\bar{v}$, $\bar{p}$, and $\bar{f}$ are to be determined.

Substituting the solution (5.2)-(5.5) into equations (1.2)-(1.3) and the boundary conditions (1.4)-(1.7), and then setting the coefficients of g term to zero, we obtain that for $f\leqq r\leqq 1$,

$$\bar{u}_\theta+\bar{p}_r-2\bar{v}=\left(\nabla^2\bar{u}-\frac{\bar{u}}{r^2}-\frac{2\bar{v}_\theta}{r^2}\right)\Big/\mathrm{Re}-\sin\theta, \tag{5.6}$$

$$2\bar{u}+\bar{v}_\theta+\bar{p}_\theta/r=\left(\nabla^2\bar{v}-\frac{\bar{v}}{r^2}+2\bar{u}_\theta\right)\Big/\mathrm{Re}-\cos\theta, \tag{5.7}$$

$$(r\bar{u})_r+\bar{v}_\theta=0, \tag{5.8}$$

and on $r=f$,

$$\bar{u}-\bar{f}_\theta=0, \tag{5.9}$$

$$f_0^2(f_0\bar{f}+\bar{p}-2\bar{u}_r/\mathrm{Re})-s(\bar{f}+\bar{f}_{\theta\theta})=0, \tag{5.10}$$

$$f_0\bar{v}+\bar{u}_\theta=0. \tag{5.11}$$

From (1.4) we obtain that on $r=1$,

$$u=0, \qquad v=1. \tag{5.12}$$

It follows from (5.6)-(5.8) that we can represent $\bar{u}$, $\bar{v}$, $\bar{p}$, and $\bar{f}$ as

$$\bar{u}=a(r)\cos\theta+b(r)\sin\theta, \tag{5.13}$$

$$\bar{v}=(rb(r))'\cos\theta-(ra(r))'\sin\theta, \tag{5.14}$$

$$\bar{p}=c(r)\cos\theta+d(r)\sin\theta, \tag{5.15}$$

$$\bar{f}=f_1\cos\theta+f_2\sin\theta, \tag{5.16}$$

where the prime indicates the derivative with respect to r, and where the functions $a(r)$, $b(r)$, $c(r)$, and $d(r)$, and the constants f_1 and f_2 are to be determined.

Substituting (5.13)-(5.16) into equations (5.6)-(5.8), and then setting the coefficients of the $\cos\theta$ and the $\sin\theta$ terms to zero, we obtain a system of ordinary differential equations

$$\begin{aligned}
&a+2ra'+d'=(b''+3b'/r)/\mathrm{Re}-1,\\
&b+2rb'-c'=-(a''+3a'/r)/\mathrm{Re},\\
&a-ra'+d/r=(rb'''+4b'')/\mathrm{Re}-1,\\
&b-rb'-c/r=-(ra'''+4a'')/\mathrm{Re}.
\end{aligned} \tag{5.17}$$

From (5.13)-(5.16) and the boundary conditions (5.9)-(5.12), we obtain the following conditions

$$\begin{aligned}
&a(1)=0, \quad b(1)=0, \quad a'(1)=0, \quad b'(1)=0,\\
&f_0b(f_0)+2a'(f_0)/\mathrm{Re}-c(f_0)=0,\\
&f_0a(f_0)+2b'(f_0)/\mathrm{Re}+d(f_0)=0,\\
&f_0b''(f_0)+b'(f_0)=0,\\
&f_0a''(f_0)+a'(f_0)=0,
\end{aligned} \tag{5.18}$$

and

$$f_2 = a(f_0),$$
$$f_1 = -b(f_0). \tag{5.19}$$

The differential equations (5.17) can be transformed to a system of eight first-order differential equations; this system of first-order differential equations together with the eight boundary conditions of (5.18) can be solved by the standard shooting method.

6. Results. Numerical experiments for the least-square method and the combined method of perturbation analysis and numerical analysis were carried on a VAX 11/780. All computations reported here used single precision arithmetic, which is 32 bits on the VAX 11/780.

Numerical experiments for the convergence behavior of the finite-element method were performed for the case of zero gravity; the results were used to guide our choice of meshes in the least-squares method. The results indicated that a radial refinement of meshes was more effective than a circumferential refinement in reducing error in the finite-element method; and that a mesh with its grid size radially proportional to the r-coordinate was better than a uniform mesh or a mesh with the size proportional to the square of the r-coordinate. Such convergence behavior was not unexpected since for the rigid-body rotation solution $d\mathbf{v}/dr = \mathbf{e}_\theta$ and $d\mathbf{v}/d\theta = \mathbf{0}$. Consequently, meshes with their radial sizes proportional to their r-coordinates and with uniform angular sizes were used in the least-squares method. Meshes of 12 elements angularly and either 4 or 6 elements radially were used.

The stopping criterion for the least-squares method was that the error between two consecutive free surface iterates was less than a number between 10^{-4} and 10^{-2}, depending on the Reynolds number. The stopping criterion for the finite-element method was that the errors between two consecutive velocity iterates and two consecutive pressure iterates were less than 10^{-4} and 10^{-3}, respectively. The routine DTPTB of the IMSL library was used to solve the ordinary differential equations arising from the combined method of numerical analysis and perturbation analysis.

6.1. Normal-stress-residual scheme vs. tangential-velocity-residual scheme. There is some controversy over whether a normal-stress-residual scheme or a tangential-velocity-residual scheme in a collocation method or a least-square method is better. Orr and Scriven [7] thought that a scheme correcting on a less accurately known boundary condition was better than a scheme correcting on a more accurately known boundary condition. Since the normal-stress boundary condition depends on the curvature of the surface, it is less accurately known than the kinematic boundary condition. Accordingly, Orr and Scriven [7] used the normal-stress-residual scheme, which they thought was better than the tangential-velocity-residual scheme. However, we argue for the opposite by pointing out that correcting on a less accurately known quantity necessarily gives greater error than does correcting on a more accurately known quantity. The two schemes perhaps have the same order of accuracy. Numerical experiments for $\mathrm{Re} = 1$, $g = 1$, $f_0 = 0.5$, and $s = 0.1$, 1, and 10 showed that with a mesh of 12 elements angularly and 4 elements radially, and an convergence criterion of 5×10^{-4} on the free-boundary location, solutions from both schemes (Problems 4 and 5 in § 3) agreed to the third digit in the free-boundary location and the velocity. Moreover, they took almost the same computational time. Thus the two methods seem to be equally good. The velocity-tangential-residual scheme (Problem 4 in § 3) was used as the primary computational scheme of the least-squares method.

6.2. The least-squares method vs. the combined method. The combined method of the perturbation analysis and numerical analysis is applicable when the gravity parameter is small. The least-squares method is, in principle, applicable to problems with an arbitrary gravity parameter. Various numerical experiments were done to compare solutions from the least-squares method and the combined method of perturbation analysis and numerical analysis; for a detailed tabulation of these comparisons, see Lin [6]. For a problem with small Reynolds number and gravity not greater than 0.1, solutions from the two methods agreed to the fourth digit; they agreed to the third digit if the Reynolds number was increased to 100. These results indicate that both methods are correct. For Reynolds numbers as high as 1500, solutions from the two methods agreed to the second digit in the free-boundary location; however, the velocity and the pressure did not agree at all, with the greatest deviations occurring near the free and the solid boundaries. This phenomenon can be attributed to the existence of boundary layers there.

6.3. The present results vs. previous experimental results. Whiting [10] made detailed measurements on the flow. After several observations, Whiting [10] made the assumption that the free boundary was approximately an ellipse. He then measured five quantities: the vertical and the horizontal displacements of the center of the air core from the axis of the cylinder; the changes in the length of the two axes from the radius f_0; and the retrograde rotation, which is defined to be $(\omega - \omega_f)/\omega$, where ω_f is the average rotational velocity of particles on the free boundary.

The vertical displacements of the air core in solutions from the least-squares method agreed well with those measured by Whiting [10] as shown in Fig. 2. The directions of the horizontal displacement of the air core agreed although their magnitudes disagreed. The retrograde rotations were found in solutions from the least-squares method with magnitudes much greater than Whiting's.

One possible source of disagreement was that the mesh used in the least-squares method was not fine enough. Flows used by Whiting [10] had high Reynolds numbers and hence had boundary layers near the solid and the free boundaries. A finer mesh for the least-squares method than the mesh used here is possibly needed to resolve the boundary layers.

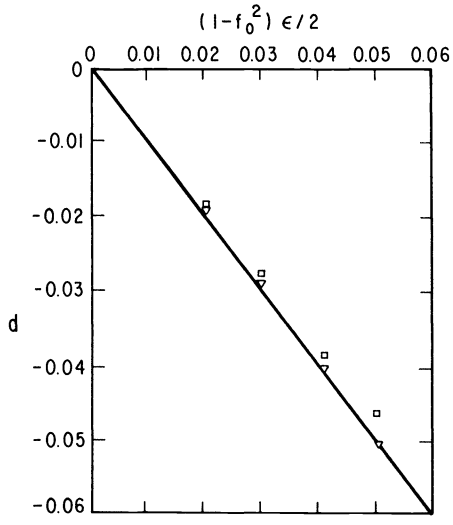


FIG. 2. *Vertical displacement of air core. Solid line*: *Phillips*; ▽: *Whiting*; □: *The least-squares method.*

6.4. Present results vs. previous theoretical results. The vertical displacements of the air core in solutions from the least-squares method agreed well with those predicted by Phillips' [8] perturbation analysis of inviscid liquids as shown in Fig. 2. Results of numerical experiments with the least-squares method were consistent with Phillips' [8] stability criterion, which was a critical rotational velocity below which the flow became unstable and collapsed. For example, Phillips' [8] stability criterion gave a critical value of 28.9 rad/sec for a cylinder of radius 7.88 cm, filled to 0.447 of the total volume with a liquid of viscosity 1.4 cm^2/sec, surface tension 72 dynes/cm and density 0.998 g/cm^3, which was used by Whiting [10]. For this flow the least-squares method diverged when the rotational velocity was 28 rad/sec. Moreover, the stopping criterion had to be increased and the accuracy of the method decreased as the rotational velocity decreased from 45 to 29 rad/sec.

6.5. Present results vs. previous numerical results. For $g=1$, $\mathrm{Re}=1$, $s=1$, and $f_0=0.5$, solutions from the least-squares method and the collocation method of Orr and Scriven [7] agreed to the third digit in the free-boundary condition, and to the second digit in the velocity and the pressure.

6.6. Sample solutions. A series of solutions with the rotational velocity continuously changed was obtained from the least-squares method for the flow described in § 6.4, which was used by Whiting [10]. The solutions were represented vividly in color pictures with red arrows representing the velocity and changing blue colors and intensities representing the pressures; see Lin [6] for these color pictures. Table 1

TABLE 1
Numerical solution at the corner nodes of elements for the flow used by Whiting [10] *with a rotating speed of* 40 *rad/sec.*

θ	r	u	v	p
0.00000	0.44564	−0.04336	0.36947	−0.00134
	0.54543	−0.01341	0.50045	0.04004
	0.66756	−0.01322	0.63548	0.09644
	0.81704	0.00870	0.80690	0.19757
	1.00000	0.00000	1.00000	0.34684
0.52360	0.42217	−0.01731	0.33054	−0.00456
	0.52373	−0.00203	0.47341	0.02273
	0.64974	−0.00905	0.61053	0.07470
	0.80607	0.00469	0.77602	0.16260
	1.00000	0.00000	1.00000	0.31612
1.04720	0.41414	−0.02171	0.32365	−0.00119
	0.51625	−0.01505	0.42302	0.01952
	0.64354	−0.01037	0.58417	0.06425
	0.80221	−0.00435	0.76761	0.14456
	1.00000	0.00000	1.00000	0.291293
1.57080	0.40512	0.00436	0.27186	−0.00355
	0.50779	−0.01188	0.41105	0.01411
	0.63649	−0.00173	0.58294	0.05576
	0.79780	−0.00541	0.76522	0.13413
	1.00000	0.00000	1.00000	0.28077
2.09440	0.41574	0.01021	0.27767	−0.00329
	0.51774	−0.00768	0.42568	0.02124
	0.64478	0.00431	0.58961	0.06285
	0.80298	−0.00493	0.77298	0.14665
	1.00000	0.00000	1.00000	0.29018

TABLE 1—(*cont.*)

θ	r	u	v	p
2.61799	0.43015	0.02358	0.31056	−0.00390
	0.53115	0.00144	0.45557	0.02824
	0.65586	0.00918	0.61074	0.07550
	0.80985	−0.00546	0.78185	0.16818
	1.00000	0.00000	1.00000	0.31368
3.14156	0.45156	0.02563	0.36175	−0.00370
	0.55086	0.00381	0.48922	0.04180
	0.67198	0.00864	0.63366	0.09606
	0.81975	−0.00875	0.79580	0.20109
	1.00000	0.00000	1.00000	0.34642
3.66519	0.46925	0.02269	0.40556	−0.00344
	0.56696	0.00025	0.51777	0.05212
	0.68502	0.00515	0.65863	0.11438
	0.82766	−0.01145	0.81235	0.23077
	1.00000	0.00000	1.00000	0.37570
4.18879	0.48171	0.01267	0.42112	−0.00382
	0.57822	−0.00574	0.53964	0.06183
	0.69406	−0.00066	0.67676	0.12841
	0.83310	−0.00924	0.83007	0.25413
	1.00000	0.00000	1.00000	0.39706
4.71239	0.48322	−0.00063	0.40673	−0.00426
	0.57957	−0.01345	0.54790	0.06424
	0.69514	−0.00232	0.69179	0.13256
	0.83375	−0.00056	0.84228	0.26057
	1.00000	0.00000	1.00000	0.40389
5.23599	0.47308	−0.00952	0.38440	−0.00510
	0.57042	−0.00617	0.56520	0.06008
	0.68780	−0.00335	0.67986	0.12488
	0.82943	0.00671	0.83443	0.25030
	1.00000	0.00000	1.00000	0.39900
5.75959	0.46303	−0.02120	0.41937	−0.00345
	0.56131	−0.00581	0.53606	0.05129
	0.68046	−0.01070	0.65879	0.11449
	0.82490	0.00658	0.81910	0.22835
	1.00000	0.00000	1.00000	0.37747

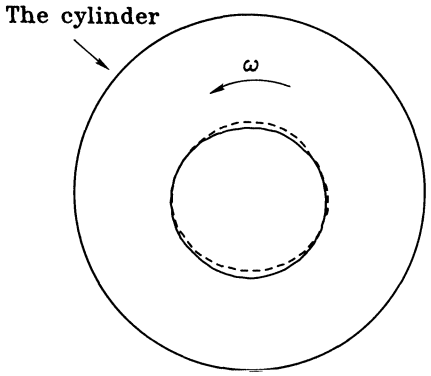


FIG. 3. *Simulated free surface locations of the flow used by Whiting* [10] *with rotating speed of* 30 *rad/sec* (*the solid curve*) *and* 50 *rad/sec* (*the dash curve*), *respectively.*

shows the numerical solution at the corner nodes of elements for this flow with a rotational velocity of 40 rad/sec. Figure 3 shows the computed free boundary locations for this flow with rotational velocity of 30 and 50 rad/sec, respectively; that the air core drops downward as the rotational velocity becomes slow is evident here.

6.7. Timing and storage. The computing time for the least-squares method depended on the closeness of the initial solution. Using a mesh of 12 elements angularly and 4 elements radially and the rigid-body solution as the initial solution, for $\mathrm{Re} = 1$, $g = 0.1$, $f = 0.5$, $s = 1$, the least-squares method required 40 fixed-boundary-value problems to be computed, and each fixed-boundary-value problem required about 300 CPU seconds on the VAX 11/780. For such a problem, 544,000 bytes were required.

7. Conclusion. In this paper we have demonstrated the capabilities of the least-square method with finite elements of mixed-order interpolations for solving a free-boundary problem governed by Navier–Stokes equations with a minicomputer. We have established the correctness of this method to some degree by comparing its solutions with solutions from perturbation methods and physical experiments. The approach presented here can also be applied to other free-boundary problems to obtain useful information about the complex free-boundary flow phenomena.

Acknowledgments. The author thanks Professor John C. Strikwerda, his Ph.D. thesis advisor at the University of Wisconsin–Madison, for his encouragement and the many stimulating discussions during this research. He also thanks Dr. William G. Pritchard for suggesting the interesting problem in this research.

REFERENCES

[1] R. F. GANS, *On steady flow in a partially filled rotational cylinder*, J. Fluid Mech., 82 (1977), pp. 415–427.
[2] H. P. GREENSPAN, *On a rotational flow disturbed by gravity*, J. Fluid Mech., 74 (1976), pp. 335–351.
[3] P. HOOD AND C. TAYLOR, *Navier–Stokes equations using mixed interpolation*, in Finite Element Methods in Flow Problems, J. T. Oden, et al., eds., 1974, pp. 121–132.
[4] B. M. IRONS, *A frontal solution program*, Int. J. Num. Meth. Eng., 2 (1970), pp. 5–32.
[5] J. KARWEIT AND S. CORRSIN, *Observation of cellular patterns in a partially filled, horizontal, rotating cylinder*, Phys. Fluids, 18 (1975), pp. 111–112.
[6] Y. LIN, *Numerical methods for free-boundary problems*, Ph.D. Thesis, Univ. Wisconsin–Madison, 1982.
[7] F. M. ORR AND L. E. SCRIVEN, *Rimming flow: numerical simulation of steady, viscous, free surface flow with surface tension*, J. Fluid Mech., 84 (1978), pp. 145–165.
[8] O. M. PHILLIPS, *Centrifugal waves*, J. Fluid Mech., 7 (1960), pp. 340–352.
[9] K. J. RUSCHAK AND L. E. SCRIVEN, *Rimming flow of liquid in a rotating horizontal cylinder*, J. Fluid Mech., 76 (1976), pp. 113–125.
[10] R. WHITING, *An experimental study of steady flow on a partially filled rotating cylinder*, Ph.D. Thesis, Univ. Rochester, Rochester, NY, 1978.

SIAM J. SCI. STAT. COMPUT.
Vol. 7, No. 2, April 1986

APPLICATION OF THE LIMITING AMPLITUDE PRINCIPLE TO ELASTODYNAMIC SCATTERING PROBLEMS*

C. L. SCANDRETT†§, G. A. KRIEGSMANN†¶ AND J. D. ACHENBACH‡§

Abstract. A technique for numerically solving the two-dimensional time harmonic equations of elastodynamics on exterior domains is presented. The method is based on the numerical realization of the limiting amplitude principle and on the construction of a modified boundary condition at "infinity". The technique is tested on two model problems: the scattering of a plane wave off a circular void and a straight crack. The results agree well with those obtained by other numerical methods.

Key words. elastodynamic waves, finite differences, numerical methods, nonreflecting boundary conditions, limiting amplitude principle

1. Introduction. The scattering of time harmonic elastic waves off compact targets is an important physical phenomenon in several branches of applied science. Examples include geophysical applications in exploratory seismology and defect characterization in nondestructive testing.

Historically, these problems have been solved by three classes of techniques. The first is analytical in character and usually depends on special geometric properties of the target. Modal expansions or transform methods are used to obtain the mathematical solution [1]. The second class of techniques is asymptotic in nature. These include Born or Rayleigh approximations, which are valid when the target is a slight perturbation in a uniform elastic host or when the wavelength of the impinging wave is large compared to the target, respectively. Also contained in this class are high frequency [2], [3] methods which are extensions of geometrical optics. These are valid when the wavelength of the incident radiation is small compared to the target's size.

The third class is numerical in nature. It contains boundary integral equations and their variants as well as the T-matrix method. It also includes those methods which directly discretize the equations. These are the finite difference and finite element methods. All of these methods require matrix inversions and become inefficient in certain limits. The later two techniques require artificial boundary conditions to render a finite numerical domain and a finite matrix problem.

The method for solving two-dimensional time harmonic scattering problems presented here is an extension of a relaxation scheme used to solve the Helmholz equation [4]. The same methodology has also been used to solve scattering problems in acoustics, electromagnetics [24], and hydrodynamics. (See [5] for a bibliography.) The basic technique is a numerical realization of the limiting amplitude principal [6]. This principle asserts that a linear elastic medium which contains a localized target will, upon excitation by a time harmonic force of frequency ω, evolve to a time harmonic state of the same frequency. This is physically evident as long as there are no bound or trapped eigenstates where energy can be stored. This is the case for the problems considered here. The method then is to numerically solve the time dependent equations

* Received by the editors September 20, 1984, and in revised form February 22, 1985.

† Department of Engineering Sciences and Applied Mathematics, the Technological Institute, Northwestern University, Evanston, Illinois 60201.

‡ Department of Civil Engineering and Department of Engineering Sciences and Applied Mathematics, the Technological Institute, Northwestern University, Evanston, Illinois 60201.

§ The work of this author was carried out under Contract DE-AC02-83ER13036.A002 with the Department of Energy, Office of Basic Energy Sciences, Engineering Research Program.

¶ The work of this author was supported by the National Science Foundation under grant MCS-8300578.

of elastodynamics with the proper time harmonic input corresponding to a given incident wave. This is done by an explicit finite difference scheme which marches the solution forward in time until a time harmonic response is obtained.

As mentioned earlier, the use of finite difference schemes on infinite domains necessitates the introduction of an artificial boundary. This makes the system of finite difference equations bounded but requires special handling of the solution at the artificial boundary. In some early work on pulse problems [7]–[12] the schemes halted when spurious reflections from the artificial boundary begin to contaminate the numerical results at points under consideration. This method is not applicable when time harmonic solutions are sought because the transients involved are not pulse like in nature. Other techniques used to handle the artificial boundary for transient problems include the addition of viscous damping at the boundary to reduce spurious reflections [13] and the addition of the solutions of two nonphysical boundary value problems which cancel any reflected waves [14]. The first method eliminates reflected compressional waves well but is not so complete in its elimination of reflected shear waves. The second method is inefficient as it requires the solution of at least six Cauchy problems.

A sequence of artificial boundary conditions were developed by Engquist and Majda to overcome many of these difficulties [15]. These conditions are local in character, as they involve radial, tangential, and temporal derivatives of the displacements, and are applied at a fixed finite boundary ($r = R_b < \infty$). The analysis used to derive these conditions is based on operator splitting which finds its mathematical justification in the theory of pseudo-differential operators. In this paper a new derivation of their second order boundary operator is given. It is based on the far field behaviors of the scattered longitudinal and shear waves and is an extension of the development given in references [4] and [16]. In addition, this approach gives a technique which determines the differential scattering cross sections to $O(R_b^{-2})$. This result is an extension of the one given by Kriegsmann and Morawetz [4] for the Helmholtz equation.

The viability of the numerical method is demonstrated in this paper by considering two fundamental scattering problems. The first is the scattering of an incident plane compressional wave off a circular void. The results are shown to compare quite well with those of a normal mode analysis even for moderately high frequencies. The second problem is the scattering of the same incident wave off a crack. The results are shown to compare favorably with those obtained from a singular integral equation method. These matters are carefully laid out in § 5.

The remainder of the paper will now be outlined. Section 2 contains the statement of the general two-dimensional scattering problem and a discussion of the limiting amplitude principle. It also contains the time dependent problem which is to be solved numerically. The artificial boundary condition and an asymptotic approximation of the compressional and shear displacements at infinity are presented in § 3. The finite difference scheme, the differencing of the artificial boundary conditions, and the discretizations of the traction free surface conditions on the target are defined and discussed in § 4. Relationships between the various physical and numerical parameters, which insure stability, are described.

2. Formulation. The problem considered is that of two-dimensional (plane strain) scattering from a compact target in an isotropic, homogeneous medium. The medium is described by the Lamé constants λ and μ and the density ρ. The target is either a void or a crack. A time harmonic incident wave with frequency ω impinges upon the target and scatters from it. (See Fig. 1.) In the region exterior to the target the scattering

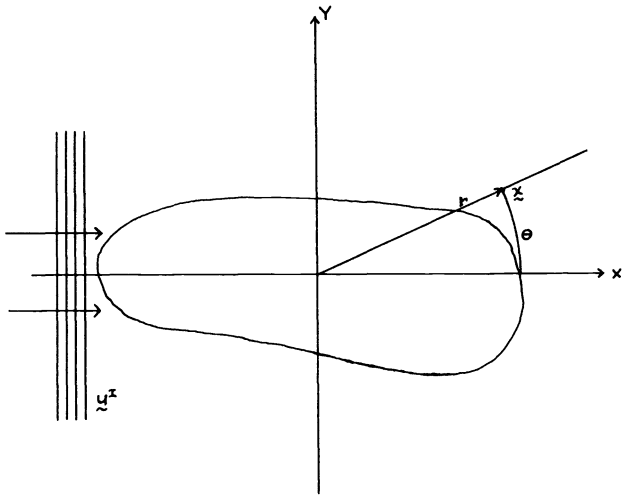


FIG. 1. *Geometry of the scattering problem.*

process is described by the vector wave equation [17]

$$L\mathbf{u}^T \equiv \frac{1}{\beta^2}\nabla^2\mathbf{u}^T + \left(\frac{1}{\alpha^2}+\frac{1}{\beta^2}\right)\nabla\nabla\cdot\mathbf{u}^T + \mathbf{u}^T = \mathbf{0}, \tag{2.1}$$

where

$$\alpha = \omega a/c_L, \quad \beta = \omega a/c_T, \quad c_L = [(\lambda+2\mu)/\rho]^{1/2}, \quad c_T = (\mu/\rho)^{1/2},$$

$\mathbf{u}^T$ is the total dimensionless displacement vector, and a is a characteristic length of the scatterer. Equation (2.1) is dimensionless. The spatial variables and displacement vector are all scaled with respect to a, while the time variable is scaled with respect to ω. Since the target is either a void or a crack, traction free boundary conditions are imposed on its boundary S. That is,

$$\tau_{ij}^T n_j = 0 \tag{2.2}$$

where τ_{ij}^T is the total stress tensor and n_j is the outward pointing normal of S.

The total displacement vector $\mathbf{u}^T$ is split into two parts

$$\mathbf{u}^T = \mathbf{u}^I + \mathbf{u} \tag{2.3}$$

where $\mathbf{u}$ is the scattered displacement and $\mathbf{u}^I$ is the incident wave which is taken as

$$\mathbf{u} \equiv (1, 0, 0)\, e^{i\alpha x}. \tag{2.4}$$

This is a plane compressional wave which travels in the positive x direction. It gives rise to an incident stress tensor with components

$$\tau_{xx}^I \equiv i\alpha(\lambda+2\mu)\, e^{i\alpha x}, \tag{2.5a}$$

$$\tau_{yy}^I \equiv i\alpha\lambda\, e^{i\alpha x}, \tag{2.5b}$$

$$\tau_{xy}^I \equiv 0, \tag{2.5c}$$

which for convenience are written in rectangular coordinates.

Inserting (2.3) and (2.4) into (2.1) it is evident that the scattered displacement vector satisfies (2.1). That is,

$$L\mathbf{u} \equiv \mathbf{0}. \tag{2.6a}$$

Decomposing the total stress tensor into the sum of an incident τ_{ij}^I and scattered component τ_{ij} and using (2.5), it is clear that

$$\tau_{ij}n_j \equiv -\tau_{ij}^I n_j \tag{2.6b}$$

on the surface S. Since the region D, which is exterior to S, is infinite in extent, $\mathbf{u}$ must have the asymptotic representation [17]

$$\mathbf{u} \sim \mathbf{a}_0 \frac{e^{i\alpha r}}{\sqrt{r}} + \mathbf{b}_0 \frac{e^{i\beta r}}{\sqrt{r}} \tag{2.6c}$$

as $r \equiv \sqrt{x^2+y^2} \to \infty$. The scattering problem is solved when the solution to (2.6) is obtained.

The time dependent problem

$$(L-1)\mathbf{U} = \mathbf{U}_{tt}, \tag{2.7a}$$

$$\mathbf{U} = \mathbf{U}_t \equiv 0, \qquad t = 0, \quad (x, y) \in D \tag{2.7b}$$

subject to the boundary conditions

$$T_{ij}n_j = -\tau_{ij}^I n_j e^{-it} \tag{2.7c}$$

and the asymptotic representation

$$\mathbf{U} \sim \mathbf{A}_0 \frac{e^{i(\alpha r - t)}}{\sqrt{r}} + \mathbf{B}_0 \frac{e^{i(\beta r - t)}}{\sqrt{r}}, \qquad r \to \infty \tag{2.7d}$$

will now be considered. The τ_{ij}^I in (2.7c) are defined in (2.5) and the time dependent stress tensor T_{ij} is related to $\mathbf{U}$ in the usual fashion [17]. The limiting amplitude principle applied to (2.7) asserts that in certain situations $\mathbf{U}$ approaches the time periodic state:

$$\mathbf{U} \to \mathbf{q}(x, y)\, e^{-it} \tag{2.8}$$

as t becomes large [4]-[6]. It follows from inserting (2.8) into (2.7) and comparing the resulting equations for (2.6) that $\mathbf{u} \equiv \mathbf{q}$, $\mathbf{A}_0 = \mathbf{a}_0$, and $\mathbf{B}_0 = \mathbf{b}_0$ as long as the solution of the scattering problem is unique.

The applicability of the limiting amplitude principle and the uniqueness of $\mathbf{u}$ are equivalent facts when no trapped modes are present. These states are given by

$$\mathbf{U} = e^{\gamma t}\boldsymbol{\psi}(x, y) \tag{2.9}$$

where γ is positive and $\boldsymbol{\psi}$ is a solution of

$$L\boldsymbol{\psi} = (\gamma^2+1)\boldsymbol{\psi}, \qquad (x, y) \in D \tag{2.10a}$$

$$T_{ij}n_j = 0 \quad \text{on } S \tag{2.10b}$$

and

$$\boldsymbol{\psi} \to 0 \quad \text{(exponentially)}, \quad r \to \infty. \tag{2.10c}$$

In (2.10b) the stress tensor T_{ij} is related to the displacement vector $\boldsymbol{\psi}$ in the usual manner. In § 3 of this paper a differential statement of (2.7d) (and (2.6c)) will be derived. It is similar in nature to the Sommerfeld radiation condition for scalar Helmholtz equations. What is important here is the fact that the solution $e^{\gamma t}\boldsymbol{\psi}$ will automatically satisfy this radiation condition because of (2.10c). Clearly a solution of this type would prove disastrous in any numerical solution of (2.7). Luckily, no such

bound states are possible. This is proved in Appendix A. Thus, the solution of (2.7) approaches the solution of (2.6). This fact is exploited in the numerical method presented in § 4.

3. Development of boundary conditions and scattering cross-sections. Since the scattered displacement vector **U** behaves like the sum of two outgoing cylindrical waves (one shear and the other compressional), polar coordinates (r, θ) will now be used. In addition, it is analytically useful to express **U** in terms of its radial displacement U and its angular displacement V. These displacements may be written in terms of two potential functions as follows [17]:

$$U=\frac{\partial \phi}{\partial r}+\frac{1}{r}\frac{\partial \psi}{\partial \theta}, \qquad V=\frac{1}{r}\frac{\partial \phi}{\partial \theta}-\frac{\partial \psi}{\partial r}, \tag{3.1a, b}$$

where ϕ and ψ satisfy the two wave equations:

$$\nabla^2\phi=\alpha^2\frac{\partial^2\phi}{\partial t^2}, \qquad \nabla^2\psi=\beta^2\frac{\partial^2\psi}{\partial t^2}. \tag{3.2a, b}$$

The time dependent Sommerfeld radiation conditions for ϕ and ψ are:

$$\frac{\partial \phi}{\partial r}+\frac{1}{2r}\phi+\alpha\frac{\partial \phi}{\partial t}=O(r^{-2}) \quad \text{as } r\to\infty, \tag{3.3a}$$

$$\frac{\partial \psi}{\partial r}+\frac{1}{2r}\psi+\beta\frac{\partial \psi}{\partial t}=O(r^{-2}) \quad \text{as } r\to\infty. \tag{3.3b}$$

As ϕ and ψ are smoothly varying functions of r, as required by the wave equations they satisfy, one may differentiate equations (3.3) with respect to r without affecting the order of their accuracy. Then, equations (3.3) become

$$\frac{1}{2r}\frac{\partial \phi}{\partial r}+\frac{\partial^2\phi}{\partial r^2}+\alpha\frac{\partial^2\phi}{\partial r\,\partial t}=O(r^{-2}), \tag{3.4a}$$

$$\frac{1}{2r}\frac{\partial \psi}{\partial r}+\frac{\partial^2\psi}{\partial r^2}+\beta\frac{\partial^2\psi}{\partial r\,\partial t}=O(r^{-2}) \quad \text{as } r\to\infty. \tag{3.4b}$$

The above equations for the potentials can be written in terms of the displacements and combined to produce the following two conditions on the scattered displacements:

$$\frac{1}{2r}U+\frac{\partial U}{\partial r}+\alpha\frac{\partial U}{\partial t}+\frac{\beta-\alpha}{\beta r}\frac{\partial V}{\partial \theta}=O(r^{-2}), \tag{3.5a}$$

$$\frac{1}{2r}V+\frac{\partial V}{\partial r}+\beta\frac{\partial V}{\partial t}+\frac{\beta-\alpha}{\alpha r}+\frac{\partial U}{\partial \theta}=O(r^{-2}). \tag{3.5b}$$

The nonreflecting boundary conditions used in the numerical scheme described in § 4, are obtained from equations (3.5a) and (3.5b) by setting the right-hand side equal to zero at $r=R_b<\infty$. The artificial boundary at $r=R_b$ yields a finite computational domain but causes a truncation error of $O(R_b^{-2})$. Conditions similar to (3.5) are given in rectangular coordinates by Engquist and Majda [15]. An extension of their calculation to cylindrical coordinates gives (3.5) when terms of $O(r^{-2})$ are neglected.

Although the present paper contains the results for two-dimensional problems, the previous analysis is easily extended to three dimensions. For completeness the

results are

$$\frac{\partial U}{\partial r}+\frac{1}{r}U+\alpha\frac{\partial U}{\partial t}+\frac{\beta-\alpha}{\beta r}\cot\theta V+\frac{\beta-\alpha}{\beta r}\frac{\partial V}{\partial\theta}+\frac{\beta-\alpha}{\beta r\sin\theta}\frac{\partial W}{\partial\phi}=O(r^{-2}), \tag{3.6a}$$

$$\frac{\partial V}{\partial r}+\frac{1}{r}V+\beta\frac{\partial V}{\partial t}+\frac{\beta-\alpha}{\alpha r}\frac{\partial U}{\partial\theta}=O(r^{-2}), \tag{3.6b}$$

$$\frac{\partial W}{\partial r}+\frac{1}{r}W+\beta\frac{\partial W}{\partial t}+\frac{\beta-\alpha}{\alpha r\sin\theta}\frac{\partial U}{\partial\phi}=O(r^{-2}), \tag{3.6c}$$

where (U, V, W) are displacements in the r, θ and ϕ directions respectively and r is $\sqrt{x^2+y^2+z^2}$.

The scattering cross section of an obstacle is taken here to mean the far field amplitudes of the scattered shear and compressional displacements as functions of θ. These will be found by using similar reasoning as above. Assuming a time harmonic solution proportional to e^{-it}, the wave equations (3.2) become the reduced equations

$$\nabla^2\phi+\alpha^2\phi=0, \qquad \nabla^2\psi+\beta^2\psi=0. \tag{3.7a, b}$$

Taken individually, for a finite scatterer, the far field solutions to the above equations resemble that of the fundamental solution for the Helmholtz equation. Considering the two-dimensional case and writing the scattered potentials in an asymptotic form we have

$$\phi\sim w^1(r,\theta)\,e^{i\alpha r-it}/\sqrt{r}, \tag{3.8a}$$

$$\psi\sim w^2(r,\theta)\,e^{i\beta r-it}/\sqrt{r}, \tag{3.8b}$$

where

$$w^1=A_0(\theta)+\frac{A_1(\theta)}{r}+\cdots, \tag{3.9a}$$

$$w^2=B_0(\theta)+\frac{B_1(\theta)}{r}+\cdots. \tag{3.9b}$$

Substituting (3.8)-(3.9) into the relationships (3.1) between potentials and displacements, the radial (compressional) displacement scattering cross section is found to be $i\alpha A_0(\theta)$ while the tangential (shear) displacement cross section is $-i\beta B_0(\theta)$. That is,

$$U\sim i\alpha A_0(\theta)\,e^{i\alpha r}/\sqrt{r}, \tag{3.10a}$$

$$V\sim -i\beta B_0(\theta)\,e^{i\beta r}/\sqrt{r} \tag{3.10b}$$

as $r\to\infty$. Therefore it remains only to find expressions for A_0 and B_0.

From equations (3.1) and (3.7) the potentials may be written in terms of the displacements as follows:

$$\phi=-\frac{1}{\alpha^2}\left(\frac{\partial U}{\partial r}+\frac{1}{r}U+\frac{1}{r}\frac{\partial V}{\partial\theta}\right), \tag{3.11a}$$

$$\psi=\frac{1}{\beta^2}\left(\frac{\partial V}{\partial r}+\frac{1}{r}V-\frac{1}{r}\frac{\partial U}{\partial\theta}\right). \tag{3.11b}$$

A derivation of the cross section for an acoustic wave is given in [4]. This result is directly applicable to (3.8)-(3.9) and will not be repeated here. The formulae using

the present notation are

$$\sqrt{r}\left\{1+\frac{i\alpha r}{8(1+\alpha^2r^2)}+\frac{i\alpha r}{2(1+\alpha^2r^2)}\frac{\partial^2}{\partial\theta^2}\right\}\phi = A_0(\theta)+O(\alpha^{-2}r^{-2}), \tag{3.12a}$$

$$\sqrt{r}\left\{1+\frac{i\beta r}{8(1+\beta^2r^2)}+\frac{i\beta r}{2(1+\beta^2r^2)}\frac{\partial^2}{\partial\theta^2}\right\}\psi = B_0(\theta)+O(\beta^{-2}r^{-2}). \tag{3.12b}$$

Eliminating the potentials from equation (3.12) using (3.11) the scattering cross sections are given by

$$i\alpha A_0(\theta) = L_*(\alpha)\left[\frac{\partial U}{\partial r}+\frac{1}{r}U+\frac{1}{r}\frac{\partial V}{\partial\theta}\right]+O(\alpha^{-2}r^{-2}), \tag{3.13a}$$

$$-i\beta B_0(\theta) = L_*(\beta)\left[\frac{\partial V}{\partial r}+\frac{1}{r}V-\frac{1}{r}\frac{\partial U}{\partial\theta}\right]+O(\beta^{-2}r^{-2}), \tag{3.13b}$$

where the operator $L_*(c)$ is defined by

$$L_*(c) = \frac{-i\sqrt{r}}{c}\left\{1+\frac{icr}{8(1+c^2r^2)}+\frac{icr}{2(1+c^2r^2)}\frac{\partial^2}{\partial\theta^2}\right\}. \tag{3.13c}$$

As mentioned in [4], for large values of the input frequency (consequently large values of α and β), problems with convergence arise. This is due to the approximation used for the radiation boundary condition, which is in turn used in the derivation of the above relationship for the scattering cross section.

The method for determining $i\alpha A_0$ and $-i\beta B_0$, used in the numerical scheme described in § 4, is now evident. Equations (3.13a, b) will be evaluated at $r = R_b$ and the O terms will be neglected. This will cause an error of $O(R_b^{-2})$ which is consistent with the error generated by the nonreflecting boundary conditions (3.5a)-(3.5b).

4. The numerical scheme. A finite difference scheme for solving the time dependent scattering problem (2.7) is now presented. Equation (2.7a) is replaced by a system of centered difference equations which are accurate to second order in spatial and temporal step sizes. Prior to each iteration of the scheme, displacements at each grid point of the numerical domain must be provided at two consecutive time steps. (These values are initially zero because of (2.7b).) From these known displacements, the diplacements at the succeeding time step are evaluated explicitly from the centered difference form of (2.7a). The differenced form of (2.7a) in rectangular and polar coordinates may be found in many references, see e.g. [18].

At the boundaries of the numerical grid the equation of motion either does not suffice, or would be incorrectly applied to find displacements at a succeeding time step. Therefore, the boundaries require special treatments.

The method used in this paper for differencing the free surface conditions (2.7c) is similar to that developed by Ilan and Loewenthal [19]. This technique was used in differencing the displacement component normal to the free surface while the "composed" method of Ilan, Ungar, and Alterman [12] was employed for the tangential component. For the problem studied here it was found that using Ilan and Loewenthal's method for both components of displacement produced more rapidly converging solutions than a combination of the two methods.

Since the difference equations used at the free surface S are dependent upon the shape of the target, only two cases will be listed here. These correspond to the bench mark problems discussed in § 5. The first target is a circular void of radius one. If U

and V denote the radial and angular displacements respectively, then the equations are

$$
\begin{aligned}
U_{1,j}^{n+1} = & -U_{1,j}^{n-1} + \frac{(\Delta t)^2}{\alpha^2 \Delta r}\left(1+\frac{2}{\Delta r}\right) U_{2,j}^{n} \\
& + \left\{2 + \frac{(\Delta t)^2}{\alpha^2}\left[-1 + \frac{1}{\Delta r}\left(1 - 4\frac{\alpha^2}{\beta^2}\right) - \frac{2}{(\Delta r)^2}\right] - \frac{2(\Delta t)^2}{\beta^2 (\Delta\theta)^2}\right\} U_{1,j}^{n} \\
& + \frac{(\Delta t)^2}{\beta^2 (\Delta\theta)^2}\left(U_{1,j+1}^{n} + U_{1,j-1}^{n}\right) \\
& + \left\{\frac{(\Delta t)^2}{2\alpha^2 \Delta\theta}\left[-1 + \frac{1}{\Delta r}\left(1 - 4\frac{\alpha^2}{\beta^2}\right)\right] + \frac{(\Delta t)^2}{2\Delta\theta\beta^2}\left(-1 + \frac{1}{\Delta r}\right)\right\}\left(V_{1,j+1}^{n} - V_{1,j-1}^{n}\right) \\
& + \frac{1}{2\Delta r \Delta\theta}\left(\frac{(\Delta t)^2}{\alpha^2} - \frac{(\Delta t)^2}{\beta^2}\right)\left(V_{2,j+1}^{n} - V_{2,j-1}^{n}\right) + \frac{2(\Delta t)^2}{\beta^2 \Delta r}\left(\frac{\tau_{rr}^{I}}{\mu}\bigg|_{r=1}\right) e^{-it},
\end{aligned}
\tag{4.1}
$$

$$
\begin{aligned}
V_{1,j}^{n+1} = & -V_{1,j}^{n-1} + \frac{(\Delta t)^2}{\beta^2 \Delta r}\left(1+\frac{2}{\Delta r}\right) V_{2,j}^{n} \\
& + \left\{2 + \frac{(\Delta t)^2}{\beta^2}\left[-1 - \frac{3}{\Delta r} - \frac{2}{(\Delta r)^2}\right] - \frac{2(\Delta t)^2}{\alpha^2 (\Delta\theta)^2}\right\} V_{1,j}^{n} \\
& + \frac{(\Delta t)^2}{\alpha^2 (\Delta\theta)^2}\left(V_{1,j+1}^{n} + V_{1,j-1}^{n}\right) \\
& + \left\{\frac{(\Delta t)^2}{2\Delta\theta\alpha^2}\left(1 - \frac{1}{\Delta r}\right) + \frac{(\Delta t)^2}{2\beta^2 \Delta\theta}\left(1 + \frac{3}{\Delta r}\right)\right\}\left(U_{1,j+1}^{n} + U_{1,j-1}^{n}\right) \\
& + \frac{1}{2\Delta r \Delta\theta}\left(\frac{(\Delta t)^2}{\alpha^2} - \frac{(\Delta t)^2}{\beta^2}\right)\left(U_{2,j+1}^{n} - U_{2,j-1}^{n}\right) - \frac{2(\Delta t)^2}{\beta^2 \Delta r}\left(\frac{\tau_{r\theta}^{I}}{\mu}\bigg|_{r=1}\right) e^{-it}
\end{aligned}
\tag{4.2}
$$

where the subscripts and superscripts of $U_{i,j}^{n}$ mean that U is evaluated at $r = 1+(i-1)\,\Delta r$, $\theta = j\,\Delta\theta$, $t = n\,\Delta t$. The stress components τ_{rr}^{I} and τ_{θ}^{I} are obtained from (2.4)-(2.5). They are

$$
\frac{1}{\mu}\tau_{rr}^{I}\big|_{r=1} = -i\alpha\left\{\frac{\lambda}{\mu} + 1 + \cos 2\theta\right\} e^{i\alpha\cos\theta}, \tag{4.3a}
$$

$$
\frac{1}{\mu}\tau_{r\theta}^{I}\big|_{r=1} = i\alpha \sin 2\theta\, e^{i\alpha\cos\theta}. \tag{4.3b}
$$

The second target is a crack located on the x-axis between ± 1. If U_x and U_y denote the horizontal vertical displacements, then the difference equations modelling this traction free surface are given by

$$
\begin{aligned}
U_{x,j,L}^{n+1} = & -U_{x,j,L}^{n-1} + \left(2 - \frac{2(\Delta t)^2}{\alpha^2 (\Delta x)^2} - \frac{2(\Delta t)^2}{\beta^2 (\Delta y)^2}\right) U_{x,j,L}^{n} + \frac{2(\Delta t)^2}{\beta^2 (\Delta y)^2} U_{x,j,L+1}^{n} \\
& + \frac{(\Delta t)^2}{\alpha^2 (\Delta x)^2}\left(U_{x,j+1,L}^{n} + U_{x,j-1,L}^{n}\right) \\
& + \left(\frac{3(\Delta t)^2}{2\Delta x\, \Delta y \beta^2} - \frac{(\Delta t)^2}{2\Delta x\, \Delta y \alpha^2}\right)\left(U_{y,j+1,L}^{n} - U_{y,j-1,L}^{n}\right)
\end{aligned}
$$

(4.4a)
$$+\left(\frac{(\Delta t)^2}{2\Delta x\,\Delta y\alpha^2}-\frac{(\Delta t)^2}{2\Delta x\,\Delta y\beta^2}\right)(U^n_{y,j+1,L+1}-U^n_{y,j-1,L+1})$$
$$-\frac{2(\Delta t)^2}{\beta^2\Delta y\mu}\tau^I_{xy}\,e^{-it},$$

(4.4b)
$$U^{n+1}_{y,j,L}=-U^{n-1}_{y,j,L}+\left(2-\frac{2(\Delta t)^2}{\beta^2(\Delta x)^2}-\frac{2(\Delta t)^2}{\alpha^2(\Delta y)^2}\right)U^n_{y,j,L}$$
$$+\frac{2(\Delta t)^2}{\alpha^2(\Delta y)^2}U^n_{y,j,L+1}+\frac{(\Delta t)^2}{\beta^2(\Delta x)^2}(U^n_{y,j+1,L}+U^n_{y,j-1,L})$$
$$+\left(\frac{(\Delta t)^2}{2\alpha^2\Delta y\,\Delta x}-\frac{3(\Delta t)^2}{2\beta^2\Delta x\,\Delta y}\right)(U^n_{x,j+1,L}-U^n_{x,j-1,L})$$
$$+\left(\frac{(\Delta t)^2}{2\alpha^2\Delta x\,\Delta y}-\frac{(\Delta t)^2}{2\beta^2\Delta x\,\Delta y}\right)(U^n_{x,j+1,L+1}-U^n_{x,j-1,L+1})$$
$$-\frac{2(\Delta t)^2}{\alpha^2\Delta y\mu}\tau^I_{yy}\,e^{-it},$$

where τ^I_{xy} and τ^I_{yy} are the resultant rectangular stress components due to the incident wave.

At the edges of the crack (for example at $x=-1$) the value for a displacement interior to the crack is approximated in order to properly center difference the $\partial^2/\partial x^2$ terms in the equations of motion. Its value was found in the following manner. The stress free boundary conditions applied Δx from the crack edge are center differenced on the upper and lower surfaces of the crack. The values of the displacements interior to the crack are averaged and this value is inserted for the unknown value in the equation of motion. The difference equations for the crack edge at $x=-1$ are:

(4.4c)
$$U^{n+1}_{x,k,L}=-U^{n-1}_{x,k,L}+\left(2-\frac{2(\Delta t)^2}{\alpha^2(\Delta x)^2}-\frac{2(\Delta t)^2}{\beta^2(\Delta y)^2}\right)U^n_{x,k,L}$$
$$+\frac{(\Delta t)^2}{\alpha^2(\Delta x)^2}\left(U^n_{x,k-1,L}+\frac{1}{2}\left(U^n_{x,k+1,L+1}+U^n_{x,k+1,L-1}+\frac{\Delta y}{\Delta x}(\Delta U^n_{y,k+2,L})\right)\right)$$
$$+\frac{(\Delta t)^2}{\beta^2(\Delta y)^2}(U^n_{x,k,L+1}+U^n_{x,k,L-1})$$
$$+\left(\frac{(\Delta t)^2}{4\alpha^2\Delta x\,\Delta y}-\frac{(\Delta t)^2}{4\beta^2\Delta x\Delta y}\right)(U^n_{y,k+1,L+1}-U^n_{y,k+1,L-1}+U^n_{y,k-1,L-1}-U^n_{y,k-1,L+1}),$$

(4.4d)
$$U^{n+1}_{y,k,L}=-U^{n+1}_{y,k,L}+\left(2-\frac{2(\Delta t)^2}{\beta^2(\Delta x)^2}-\frac{2(\Delta t)^2}{\alpha^2(\Delta y)^2}\right)U^n_{y,k,L}$$
$$+\frac{(\Delta t)^2}{\beta^2(\Delta x)^2}\left(U^n_{y,k-1,L}+\frac{1}{2}\left(U^n_{y,k+1,L+1}+U^n_{y,k+1,L-1}+\frac{\Delta y}{\Delta x}\left(\frac{\lambda}{\lambda+2\mu}\right)(\Delta U^n_{x,k+2,L})\right)\right)$$
$$+\frac{(\Delta t)^2}{\alpha^2(\Delta y)^2}(U^n_{y,k,L+1}+U^n_{y,k,L-1})$$
$$+\left(\frac{(\Delta t)^2}{4\alpha^2\Delta x\,\Delta y}-\frac{(\Delta t)^2}{4\beta^2\Delta x\,\Delta y}\right)(U^n_{x,k+1,L+1}-U^n_{x,k+1,L-1}+U^n_{x,k-1,L-1}-U^n_{x,k-1,L+1}),$$

where ΔU_x and ΔU_y represent horizontal and vertical crack opening displacements respectively.

Since the radiation conditions (3.5) are given in terms of U and V, the displacements U_x and U_y for the crack problem must be converted into polar displacements. In practice U_x and U_y are computed using the rectangular difference equations corresponding to (2.7a) on relatively few mesh points surrounding the crack. These values are interpolated to give U and V on the circle $r = R_c$. (This will be discussed in § 5.) Then the cylindrical difference equations corresponding to (2.7a) are used to march the solution out to $r = R_b$, $(i = m)$. For either the circular void or crack problem both the outgoing boundary conditions (3.5a), (3.5b) and the equations of motion (2.7a) are used to eliminate the need for additional grid points required in the centered difference calculation of radial derivatives. The expression for the radial displacement at $r = R_b$ and $t = (n+1)\Delta t$ is

$$U_{m,j}^{n+1} = C_1 U_{m,j}^{n-1} + c_2 U_{m-1,j}^{n} + c_3 U_{m,j}^{n} + c_4 (U_{m,j+1}^{n} + U_{m,j-1}^{n}) + c_5 (V_{m,j+1}^{n} - V_{m,j-1}^{n}) + c_6 (V_{m-1,j+1}^{n} - V_{m-1,j-1}^{n}), \tag{4.5}$$

where

$$c_1 \equiv q\left\{\Delta t\left(1+\frac{\Delta r}{2r}\right) - \alpha\,\Delta r\right\}, \tag{4.6a}$$

$$c_2 \equiv 2q(\Delta t)^2/\alpha\,\Delta r, \tag{4.6b}$$

$$c_4 \equiv \alpha(\Delta t)^2 \Delta r q/\beta^2 r^2 (\Delta\theta)^2, \tag{4.6c}$$

$$c_3 \equiv -c_2 - 2c_4 + q\left\{\frac{-(\Delta t)^2}{\alpha r}\left(1+\frac{3\Delta r}{2r}\right) + 2\alpha\,\Delta r\right\}, \tag{4.6d}$$

$$c_5 \equiv \frac{\beta\,\Delta r(\Delta t)^2 q}{2r\,\Delta\theta}\left\{\left(\frac{1}{\Delta r}+\frac{1}{r}\right)\left(\frac{1}{\alpha}-\frac{1}{\beta}\right)^2 + \frac{1}{\alpha\beta r}\left(1+\frac{\beta}{\alpha}\right)\right\}, \tag{4.6e}$$

$$c_6 \equiv \frac{\alpha(\Delta t)^2 q}{2r\,\Delta\theta}\left(\frac{1}{\beta^2}-\frac{1}{\alpha^2}\right), \tag{4.6f}$$

$$q \equiv [\alpha\,\Delta r + \Delta t(1+\Delta r/2r)]^{-1}, \tag{4.6g}$$

$r = R_b$, and the index j runs from 0 to N.

A similar equation for the tangential displacements $V_{m,j}^{n+1}$ is given by (4.5) and (4.6) except the U's and V's are interchanged, α and β are switched, and the constants C_5 and C_6 change sign.

A stability criterion for initial value problems in elastodynamics has been derived by Alterman and Karal [7], and Alterman and Loewenthal [8]. Following the work of the first set of authors a limit on the size of the time increment for the numerical scheme is:

$$\Delta t \leqq \min\{\Delta r, \Delta\theta\} \Big/ \sqrt{\frac{1}{\alpha^2}+\frac{1}{\beta^2}}. \tag{4.7}$$

When free surfaces or artificial boundaries are present, as is the case for the problems considered here, other types of instabilities can occur. As discussed by Trefethen [20], the finite difference equations are dispersive in character and possess nonphysical solutions which travel with group velocities that are significantly different from c_T and c_L. In fact, these parasitic waves may have negative group velocities. If the discretized boundary conditions allow these spurious waves to propagate into the

numerical domain, then an instability will occur. The analysis of this possibility is contained in the stability theory of Gustafsson, Kreiss, and Sundström [21] for initial, boundary value problems. The application of this theory to the present problem is difficult because of its abstract and complex nature. However, it was found experimentally that no such instabilities occurred with the implementation of (4.5) and either (4.1)-(4.2) or (4.4). The results of these numerical experiments are presented in § 5.

As mentioned in § 2, the limiting amplitude principle ensures the existence of a steady state solution to the scattering problem shown schematically by (2.8) with $\mathbf{q} \equiv \mathbf{u}$. The numerical scheme is to be halted when the scattered displacements have reached this steady state. To test for this condition, consider the set of vector displacements

$$W = \{\mathbf{U}_s^n(\theta), \mathbf{V}_s^n(\theta), \mathbf{U}_{FF}^n(\theta), \mathbf{V}_{FF}^n(\theta)\} \tag{4.8}$$

which are respectively the surface radial and tangential displacements (or $U_s^n(x)$, $V_s^n(x)$ representing horizontal and vertical displacements), and the far field radial and tangential displacements evaluated at time level n. If each term of the set satisfies the criterion

$$\left| 1.0 - \frac{\|\mathbf{g}(\theta)\|^{n+1}}{\|\mathbf{g}(\theta)\|^n} \right| \leqq \varepsilon \tag{4.9}$$

where the $\|\cdot\|$ denotes the euclidean norm of a vector and $|\cdot|$ denotes the absolute value of a number, then the time harmonic scattered solution is numerically found, to a certain degree of accuracy, and the computations are stopped.

When the steady-state solution is found, two further techniques are employed to improve the values of the scattering cross section. The first is a local difference operator which is the result of the analyses performed in § 3. The second method utilizes the displacements found at the free surface in an integral operator. An outline for the setup of the integral is given in [2] and the integrals themselves may be found in Appendix B. For the local difference method the equations (3.13) are differenced to yield

$$\begin{aligned} i\alpha A_0 = {} & Q_1[(U_{m,j}^n - U_{m-2,j}^n)/2\Delta r + U_{m-1,j}^n/r + (V_{m-1,j+1}^n - V_{m-1,j-1}^n)/2r\Delta\theta] \\ & + Q_2\{(U_{m,j+1}^n - 2U_{m,j}^n + U_{m,j-1}^n - U_{m-2,j+1}^n + 2U_{m-2,j}^n - U_{m-2,j-1}^n)/2\Delta r(\Delta\theta)^2 \\ & + (U_{m-1,j+1}^n - 2U_{m-1,j}^n + U_{m-1,j-1}^n)/r(\Delta\theta)^2 \\ & + (V_{m-1,j+2}^n - 2V_{m-1,j+1}^n + 2V_{m-1,j-1}^n - V_{m-1,j-2}^n)/2(\Delta\theta)^3 r\}, \end{aligned} \tag{4.10a}$$

$$\begin{aligned} -i\beta B_0 = {} & Q_3\{(V_{m,j}^n - V_{m-2,j}^n)/2\Delta r + V_{m-1,j}^n/r - (U_{m-1,j+1}^n - U_{m-1,j-1}^n)/2r\Delta\theta\} \\ & + Q_4\{(V_{m,j+1}^n - 2V_{m,j}^n + V_{m,j-1}^n - V_{m-2,j+1}^n + 2V_{m-2,j}^n - V_{m-2,j-1}^n)/2\Delta r(\Delta\theta)^2 \\ & + (V_{m-1,j+1}^n - 2V_{m-1,j}^n + V_{m-1,j-1}^n)/r(\Delta\theta)^2 \\ & - (U_{m-1,j+2}^n - 2U_{m-1,j+1}^n + 2U_{m-1,j-1}^n - U_{m-1,j-2}^n)/2r(\Delta\theta)^3\}, \end{aligned} \tag{4.10b}$$

where

$$Q_1 = -i\frac{\sqrt{r}}{\alpha}\left[1 + \frac{i\alpha r}{8(1+\alpha^2 r^2)}\right], \tag{4.10c}$$

$$Q_2 = \frac{r\sqrt{r}}{2(1+\alpha^2 r^2)}, \tag{4.10d}$$

$$Q_3 = -i\frac{\sqrt{r}}{\beta}\left[1 + \frac{i\beta r}{8(1+\beta^2 r^2)}\right], \tag{4.10e}$$

$$Q_4 = \frac{r\sqrt{r}}{2(1+\beta^2 r^2)}, \tag{4.10f}$$

and $i = 0, 1, \cdots, N$, $r = R_b$.

5. Numerical results. The numerical results presented in this section were computed on a Cyber 170-730 running under a SYS/NUCC operating system. The graphical results shown in Figs. 2-10 were made by a Tektronix flat bed plotter. The total running time for the various numerical experiments was approximately one and a half minutes.

The first benchmark problem is that of a circular cavity of radius one. The relevant parameters are $\alpha = 2$, $\beta = 4$, $\Delta r = \Delta t = 0.1$, $\Delta\theta = \pi/30$, $R_b = 2$, and $\nu = \frac{1}{3}$ where ν is Poisson's ratio. The dashed curves shown in Fig. 2 are the magnitudes of the scattered radial and tangential displacements, on the cavity's surface, as given by the present method. The solid curves in Fig. 2 are the results of a normal mode analysis performed by Dr. J. E. Gubernatis at the Los Alamos National Laboratory. The results are in good quantitative agreement considering the coarse mesh and relatively small numerical domain ($R_b = 2$) used in the calculations.

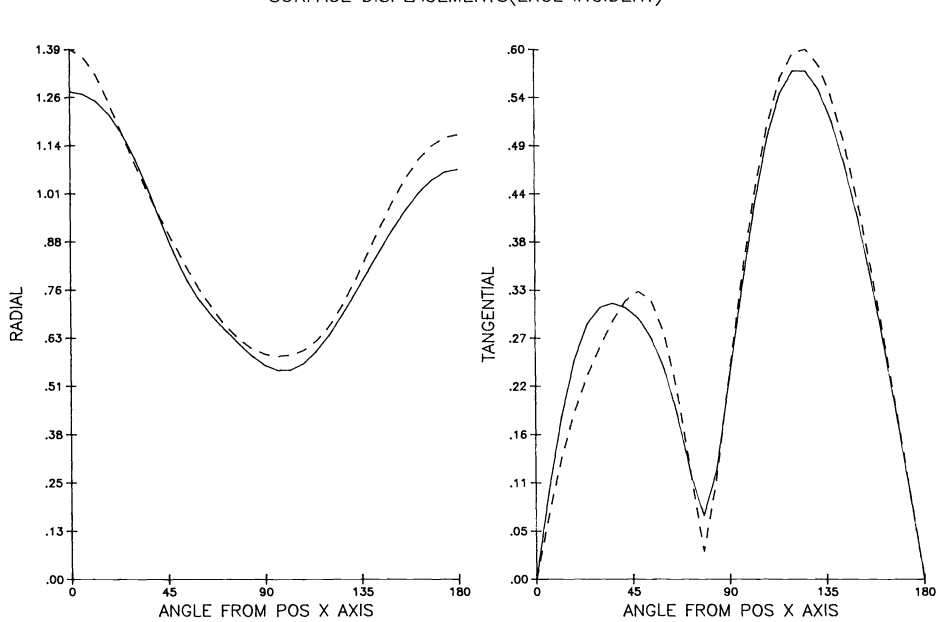


FIG. 2. *Magnitudes of the scattered radial, $\|u(\theta)\|$, and tangential, $\|v(\theta)\|$, surface displacements as functions of θ for the problem of a cylindrical cavity scatterer. Solid line: exact; dashed line: numerical solution, where $\Delta r = \Delta t = .1$, $\Delta\theta = \pi/30$, $\alpha = 2$, $\beta = 4$.*

Figure 3 contains three approximations of the magnitudes of A_0 and B_0. The solid curves are again the result of a normal mode calculation. The dashed curves are the result of applying (4.10) while the dotted curves are obtained from a trapezoidal approximation to the far field integral operators listed in Appendix B. Excellent agreement is found between the first and the last set of curves. As one would expect for large values of R_b (holding all other parameters fixed), the local boundary operator technique becomes increasingly more accurate. This is due to an increase in the accuracy of the numerical values of the displacements at $r = R_b$.

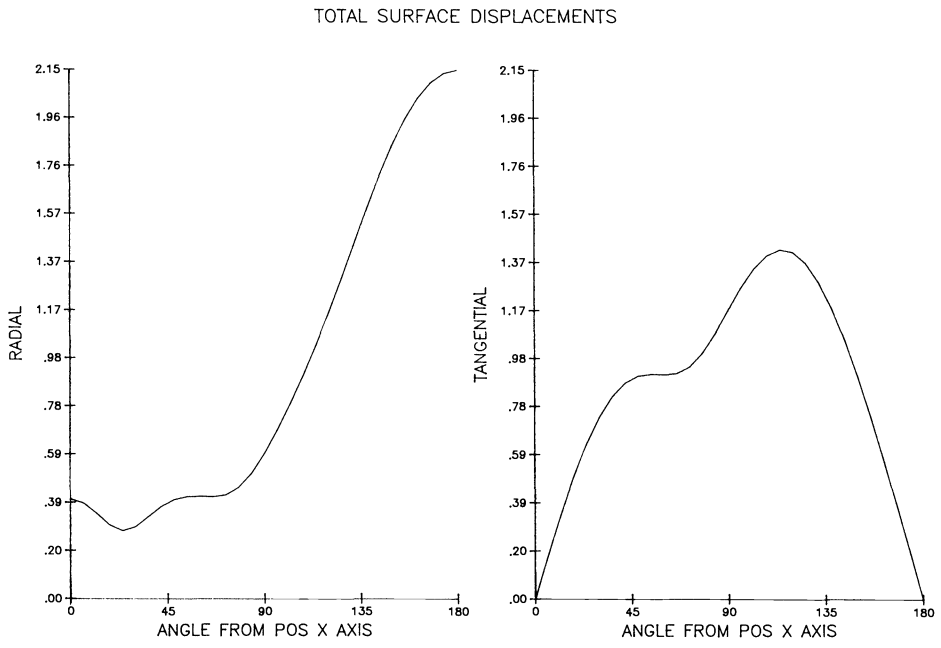


FIG. 3. *Magnitudes of the scattered radial and tangential far field displacements as functions of θ for the cylindrical cavity. Parameters same as in Fig. 2. Solid line: exact; dashed line: local boundary operator applied at R_b; dotted line: integral operator method applied at cylindrical cavity surface.*

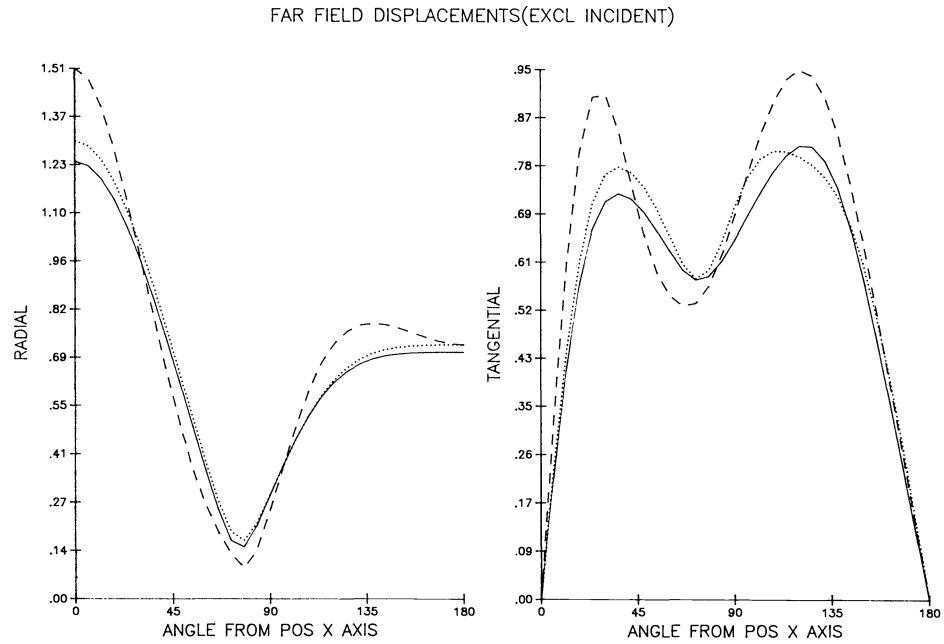


FIG. 4. *Magnitudes of the total radial and tangential surface displacements as functions of θ for the cylindrical cavity. Parameters same as in Fig. 2.*

Figure 4 shows the magnitudes of the total radial and tangential surface displacements as functions of θ. These are the results of the present numerical method. As one would expect, the compressional displacement is largest at the point of impact of the incident wave and drops sharply in the "shadow" region $0 \leqq \theta \leqq 90°$. The tangential component of surface displacement is zero along the line of symmetry for the problem, and has a relative maximum in the "lit" region of the cylindrical surface.

Also considered is an incident wave with a higher frequency giving $\alpha = 10$ and $\beta = 20$. The changes in the relevant parameters are $\Delta r = 0.03$, $\Delta\theta = \pi/100$, $R_b = 3.01$. The choices for Δr and $\Delta\theta$ increments were made with the knowledge of work on grid dispersion for numerical solutions of elastodynamic problems done by Boore [11] and Alford, Kelly, and Boore [22]. The increments were picked to allow approximately 10 nodes per wavelength of the scattered (time harmonic) shear wave.

The results compare very favorably to those tabulated in [2]. The effect of increasing the frequency of the incident displacement is to lower the relative amplitude of the displacements on the shadow side, with the addition of several diffraction lobes as seen in Fig. 5. Here the curves represent the magnitudes of the total radial and tangential surface displacements.

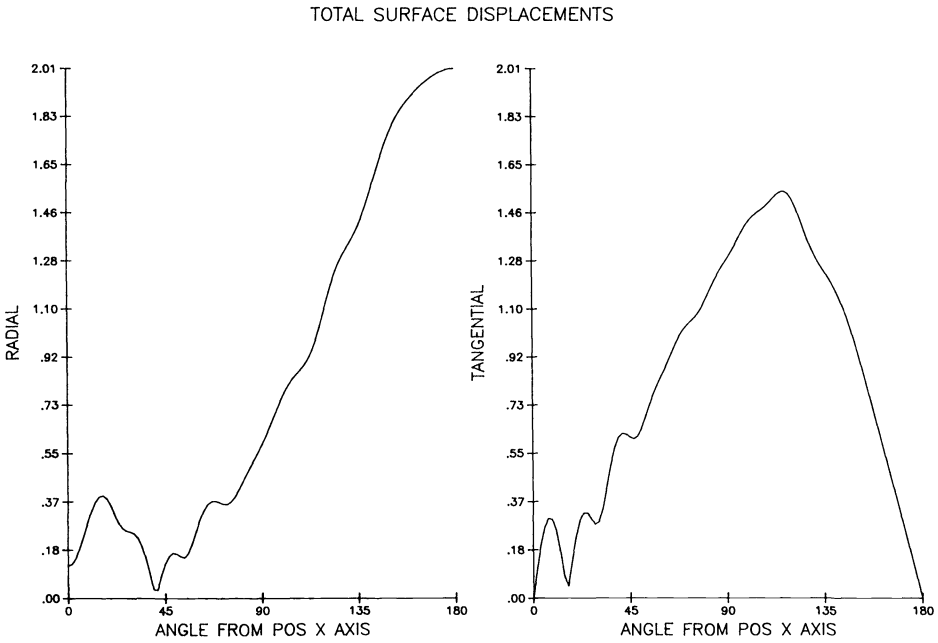


FIG. 5. *Magnitudes of the total radial and tangential surface displacements as functions of θ for the cylindrical cavity. Parameters are taken as*:

$$\Delta r = .03,\ \Delta\theta = \pi/100,\ R_b = 3.01,\ \alpha = 10,\ \beta = 20,\ \Delta t = .1.$$

Figure 6 shows the magnitudes of the scattered radial and tangential displacements in the far field. The dashed cuves are obtained from the numerical data using (4.10) while the solid curves again come from a trapezoidal approximation to the integral operators listed in Appendix B. Once again the integral method appears to be slightly better than the local difference method in computing $|A_0|$ and $|B_0|$. This is based on a qualitative comparison of the exact results [2] and those shown in Fig. 6.

The second problem is that of a crack located on the x axis between $x = \pm 1$. As mentioned in § 4, rectangular difference equations corresponding to (2.7a) are initially

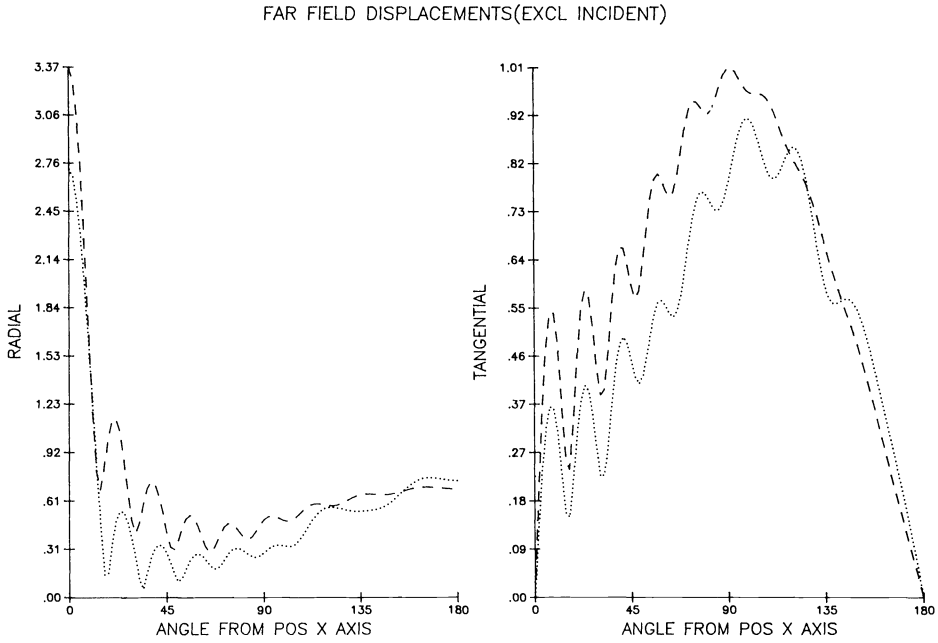


FIG. 6. *Magnitudes of the scattered radial and tangential far field displacements as functions of θ for the cylindrical cavity. Parameters same as in Fig. 5, dashed line: local boundary operator applied at R_b; dotted line: integral operator applied at the free surface of the cylindrical cavity.*

used, in conjunction with the boundary conditions (4.4), to march the numerical solution away from the crack. In practice a square grid of size 2.6×2.6 centered at the origin is filled with rectangular displacement values at t_{n+1}. These values are then used, along with interpolation and geometric transforms, to obtain the radial and tangential displacements on the circle $r = R_c \equiv 1.1$. These displacements are then marched out to the artificial boundary at $r = R_b$.

The relevant parameters used in the calculation are $\Delta r = 0.1$, $\Delta\theta = \pi/30$, $\Delta t = 0.1$, $R_b = 2.5$, $\alpha = 1.858961$, $\beta = 3.477798$, and $\Delta x = \Delta y = 0.1$ for the square grid. Figure 7 shows the magnitudes of the crack opening displacements when the incident compressional wave is given by

$$u^I \equiv (0, 1, 0)\, e^{i\alpha y}. \tag{5.1}$$

There is no shear force acting on the crack so the horizontal crack opening displacement $\Delta u \equiv 0$. There are two curves representing the vertical crack opening displacement Δv. The solid graph is the result of the present method. The dashed curve is the solution of a singular integral equation used by Keer, Lin and Achenbach [23] to study the same problem. The answers compare well. The discrepancy is caused to some extent by the coarseness of the numerical mesh and the modest size of R_b. The probable cause of the deviation is due to the fact that the derivatives of the displacements become singular at the crack tips. The coarseness of the mesh masks this problem.

The radial and tangential cross sections are shown in Fig. 8 for the present calculation. They are obtained by again approximating the integral operators of Appendix B by the trapezoidal method and using the numerically computed surface displacements. The radial cross section, $|A_0|$, has a peak at 90° which corresponds to the geometric shadow cast by the crack. The maximum at 270° is caused by the specularly reflected plane wave.

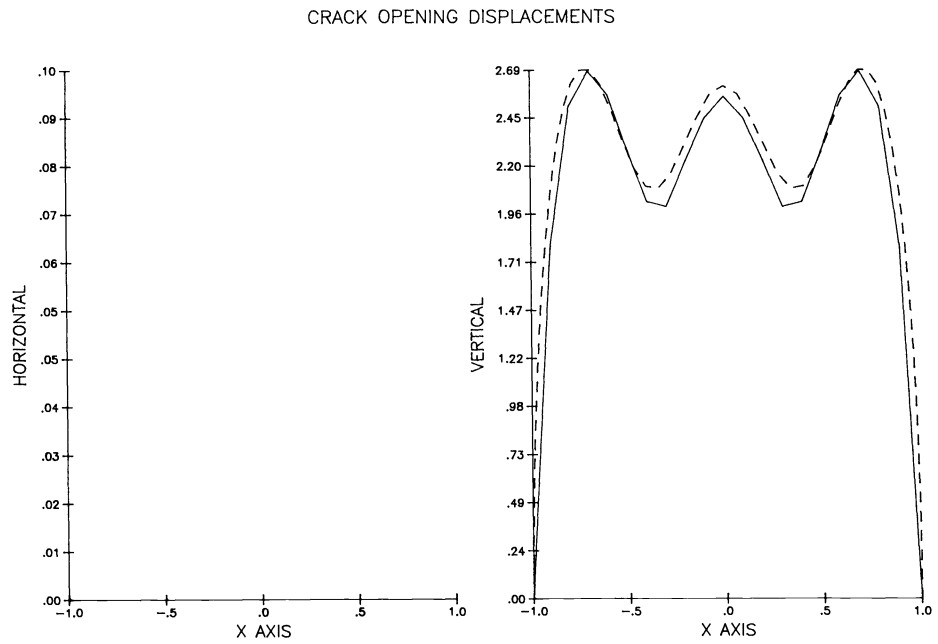


FIG. 7. *Magnitudes of the crack opening displacements* $|\Delta u|$ *and* $|\Delta v|$. *The parameters are*; $\Delta r = 0.1$, $\Delta\theta = \pi/30$, $\Delta t = 0.1$, $R_b = 2.5$, $R_c = 1.1$, $\alpha = 1.858961$, *and* $\beta = 3.47798$. *The solid curves are the results of the present method. The dashed lines correspond to solutions obtained from a singular integral equation formulation. The incident wave impinges normally upon the crack.*

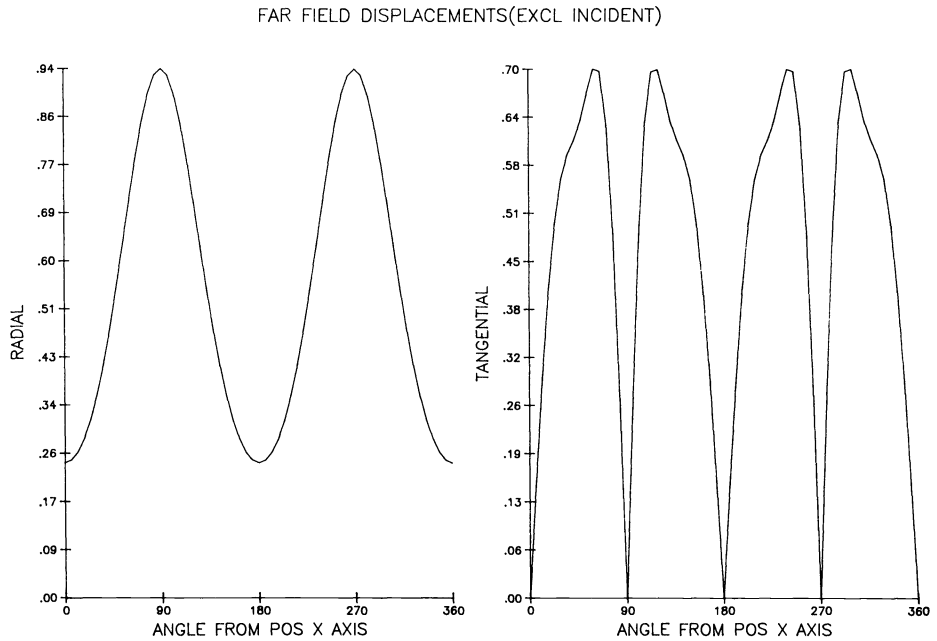


FIG. 8. *Same problem and parameters as in Fig. 7. The curves represent the magnitudes of the far field displacements* $|A_0|$ *and* $|B_0|$.

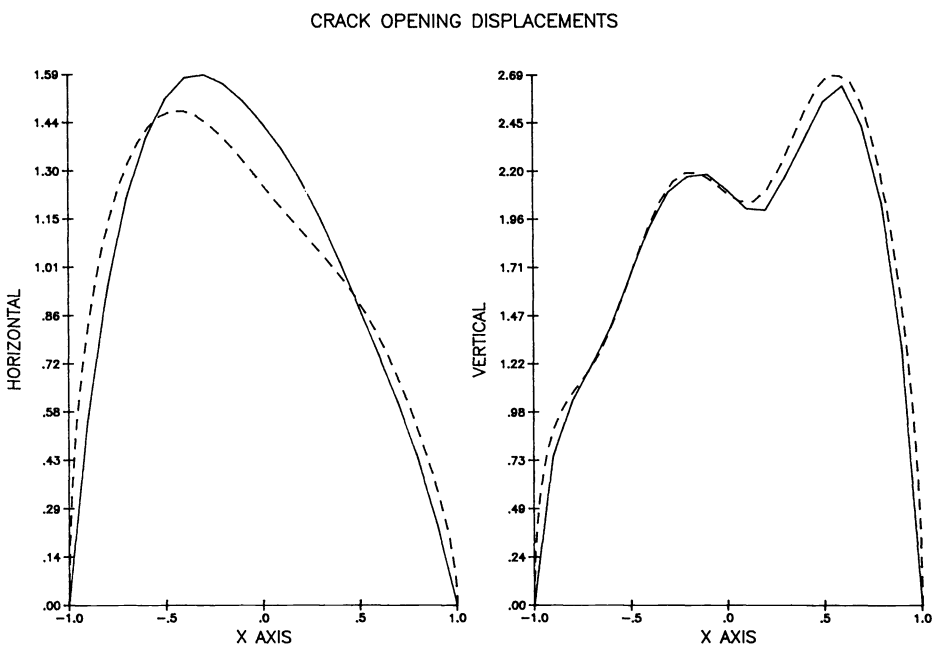


FIG 9. *Same as Fig. 7 except the incident wave makes an angle of* 60° *with the negative x-axis.*

The final problem considered arises when an incident compressional pulse strikes the crack under an angle of 120°. Hence, the incident wave is taken as

$$u^I = \left(-\frac{1}{2}, \frac{\sqrt{3}}{2}, 0\right) e^{(i\alpha/2)(\sqrt{3}y - x)}. \tag{5.2}$$

The relevant parameters are unchanged. The crack opening displacements Δu and Δv for this case are shown in Fig. 9. The solid curves are the results of the present calculations while the dashed are the results of the singular integral equation approach [22]. Except for an apparent phase shift in the Δu curves, the results compare quite well. The corresponding far field displacements $|A_0|$ and $|B_0|$ are shown in Fig. 10.

Appendix A. The nonexistence of bound states. It will now be demonstrated that (2.10) has no solution when $\gamma > 0$. Defining the strain and stress tensors corresponding to ψ by

$$\varepsilon_{ij} = \tfrac{1}{2}[\psi_{i,j} + \psi_{j,i}], \tag{A.1}$$

$$T_{ij} = \lambda \varepsilon_{kk} \delta_{ij} + 2\mu \varepsilon_{ij}, \tag{A.2}$$

respectively, equation (2.10a) can be rewritten as

$$T_{ij,j} = \rho \gamma^2 \psi_i. \tag{A.3}$$

Since the target is a void, the tractions $T_{ij}n_j$ vanish on the surface S. Thus, it follows that

$$\int_S T_{ij} n_j \bar{\psi}_i \, dl = 0 \tag{A.4}$$

where l is the arc length along S and the bar denotes complex conjugation. By applying

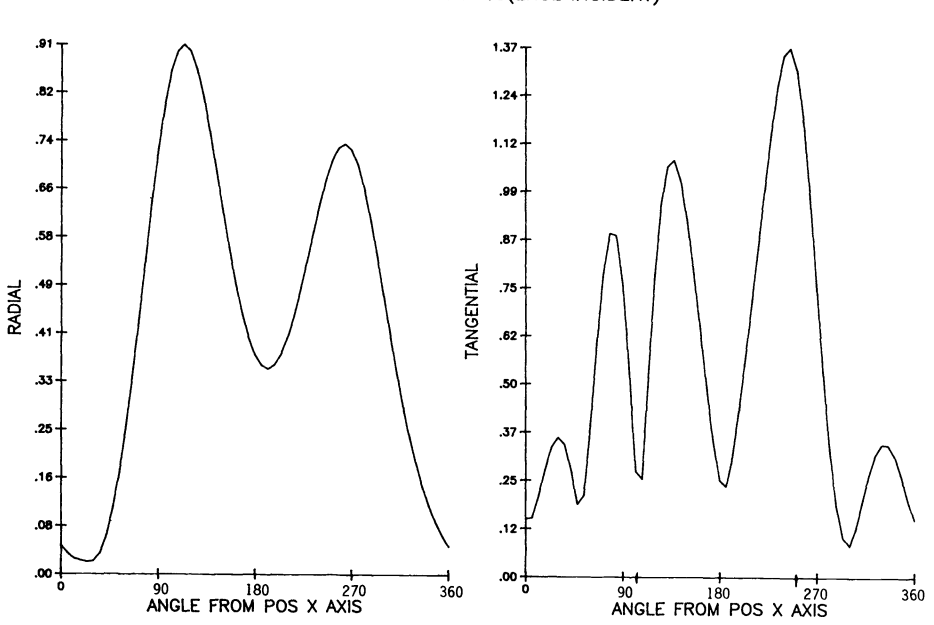


FIG. 10. *Same as Fig.* 8 *except the incident wave makes an angle of* 60° *with the negative x-axis.*

Green's theorem two times, equation (A.4) becomes

$$\iint_D (T_{ij}\bar{\psi}_i)_{,j}\, dx\, dy = 0 \tag{A.5}$$

where D is the infinite region exterior to S. The integral in (A.5) exists because of the decay condition (2.10c). Differentiating the integrand in (A.5) and using (A.3) yields

$$\rho^{\gamma^2} \sum_{i=1}^{2} \iint_D |\psi_i|^2\, dx\, dy = -\iint_D T_{ij}\bar{\psi}_{i,j}\, dx\, dy. \tag{A.6}$$

The integrand on the right-hand side of (A.6) can be rewritten [17] as $T_{ij}\bar{\psi}_{i,j} \equiv T_{ij}\bar{\varepsilon}_{ij}$. Inserting this into the integral and using (A.2), equation (A.6) becomes

$$\rho^2\gamma^2 \sum_{i=1}^{2} \iint_D |\psi_i|^2\, dx\, dy \equiv -\iint_D [\lambda\varepsilon_{kk}\bar{\varepsilon}_{ii} + 2\mu\varepsilon_{ij}\bar{\varepsilon}_{ij}]\, dx\, dy. \tag{A.7}$$

Equation (A.7) leads to a contradiction. The left-hand side is positive while the right is negative. Thus γ is purely imaginary or $\gamma_i \equiv 0$. In either case a bound state is nonexistent.

Appendix B. Far field integral operators. In references [2] and [3] integral operators are obtained which relate the displacements on a target's surface to the far field displacements. For the circular cavity of radius one, they are

$$|A_0(\phi)| = \left(\frac{\alpha}{8\pi}\right)^{1/2} \left| \int_0^{2\pi} [p_1(\theta, \phi)u(1, \theta) + p_2(\theta, \phi)v(1, \theta)]\, e^{-i\alpha\cos(\theta-\phi)}\, d\theta \right|, \tag{B.1}$$

$$|B_0(\phi)| = \left(\frac{\beta}{8\pi}\right)^{1/2} \left| \int_0^{2\pi} [p_3(\theta, \phi)u(1, \theta) - p_4(\theta, \phi)v(1, \theta)]\, e^{-i\beta\cos(\theta-\phi)}\, d\theta \right|, \tag{B.2}$$

where

$$p_1(\theta, \phi) = (1 - \kappa^{-2} + \kappa^{-2} \cos 2\phi) \cos \theta + \kappa^{-2} \sin 2\phi \sin \theta,$$

(B.3)

$$p_2(\theta, \phi) = \kappa^{-2} \sin 2\phi \cos \theta + (1 - \kappa^{-2} - \kappa^{-2} \cos 2\phi) \sin \theta,$$

(B.4) $$p_3(\theta, \phi) \equiv \sin (2\phi - \theta);\ p_4(\theta, \phi) \equiv \cos (2\phi - \theta),$$

and $\kappa = \beta/\alpha$.

When the target is a crack on the x axis with $|x| < 1$, then the expressions are

$$|A_0(\phi)| = \left(\frac{\alpha}{8\pi}\right)^{1/2} \left| \int_{-1}^{1} [\Delta u(x) \kappa^{-2} \sin 2\phi + \Delta v(x)(1 - \kappa^{-2} - \kappa^{-2} \cos 2\phi)]\, e^{-i\alpha x \cos \phi}\, dx \right|, \tag{B.5}$$

$$|B_0(\phi)| = \left(\frac{\beta}{8\pi}\right)^{1/2} \left| \int_{-1}^{1} [\Delta u(x) \cos 2\phi + \Delta v(x) \sin 2\phi]\, e^{-i\beta x \cos \phi}\, dx \right|, \tag{B.6}$$

where Δu and Δv are the jumps in the horizontal and vertical displacements across the crack.

REFERENCES

[1] Y. H. Pao and C. C. Mow, *Diffraction of Elastic Waves and Dynamic Stress Concentration*, Crane Russak, New York, 1973.

[2] R. J. Brind, J. D. Achenbach and J. E. Gubernatis, *High frequency scattering of elastic waves from cylindrical cavities*, Wave Motion, 6 (1984), pp. 41-60.

[3] J. D. Achenbach, A. K. Gautesen and H. McMaken, *Ray Methods for Waves in Elastic Solids*, Pitman, London, 1982.

[4] G. A. Kriegsmann and C. S. Morawetz, *Solving the Helmholtz equation for exterior problems with variable index of refraction* I, this Journal, 1 (1980), pp. 371-385.

[5] G. A. Kriegsmann, *Exploiting the limiting amplitude principle to numerically solve scattering problems*, Wave Motion, 4 (1982), pp. 371-380.

[6] C. S. Morawetz, *The limiting amplitude principle*, Comm. Pure Appl. Math., 15 (1962), pp. 349-361.

[7] Z. Alterman and F. Karal, *Propagation of elastic waves in layered media by finite difference methods*, Bull. Seismol. Soc. Amer., 58 (1968), pp. 367-398.

[8] Z. Alterman and D. Loewenthal, *Seismic waves in a quarter and three quarter plane*, Geophys. J. Roy. Astron. Soc., 20 (2) (1970), pp. 101-126.

[9] Z. Alterman and A. Rotenberg, *Seismic waves in a quarter space*, Bull. Seismol. Soc. Amer., 59 (1) (1969), pp. 347-368.

[10] L. D. Bertholf, *Numerical solution for two-dimensional elastic wave propagation in finite bars*, J. Appl. Mech., 34 (1967), pp. 725-734.

[11] D. Boore, *Finite difference methods for seismic wave propagation in heterogeneous materials*, Meth. Comput. Phys. (Seismology), 115 (1972), pp. 1-37.

[12] A. Ilan, A. Ungar and Z. Alterman, *An improved representation of boundary conditions in finite difference schemes for seismological problems*, Geophys. J. Roy. Astron. Soc., 43 (1975), pp. 727-747.

[13] J. Lysmer and R. L. Kuhlemeyer, *Finite dynamic model for infinite media*, J. Engineering Mech. Div. ASCE EM 4, 95 (1969), pp. 859-877.

[14] W. D. Smith, *A non-reflecting plane boundary for wave propagation problems*, J. Comput. Phys., 15 (1974), pp. 492-503.

[15] B. Engquist and A. Majda, *Radiation boundary conditions for acoustic and elastic wave calculations*, Comm. Pure Appl. Math., 32 (1979), pp. 313-357.

[16] A. Bayliss and E. Turkel, *Radiation boundary conditions for wave-like equations*, Comm. Pure Appl. Math., 33 (1980), pp. 707-725.

[17] J. D. Achenbach, *Wave Propagation in Elastic Solids*, North-Holland, Amsterdam, 1973.

[18] R. J. BLAKE, L. J. BOND AND A. L. DOWNIE, *Advances in numerical studies of elastic wave propagation and scattering*, Review of Progress in Quantitative Nondestructive Evaluation, VI, Plenum, New York, 1982, pp. 157–165.

[19] A. ILAN AND D. LOEWENTHAL, *Instability of finite difference schemes due to boundary conditions in elastic media*, Geophys. Prospecting, 24 (1976), pp. 431–453.

[20] L. N. TREFETHEN, *Group velocity in finite difference schemes*, SIAM Rev., 24 (1982), pp. 114–136.

[21] L. N. TREFETHEN, *Group velocity interpretation of the stability theory of Gustafsson, Kreiss, and Sundström*, J. Comput. Phys., 49 (1983), pp. 199–217.

[22] R. M. ALFORD, K. R. KELLY AND D. BOORE, *Accuracy of finite difference modeling of the acoustic wave equation*, Geophys., 39 (1974), pp. 834–842.

[23] L. M. KEER, W. LIN AND J. D. ACHENBACH, *Resonance effects for a crack near a free surface*, J. Appl. Mech., 51 (1984), pp. 65–70.

[24] G. MUR, *Absorbing boundary conditions for the finite difference approximation of time domain electromagnetic-field equations*, IEEE Trans. Electromagn. Compat. EMC-23 (1981), pp. 377–382.

SIAM J. SCI. STAT. COMPUT.
Vol. 7, No. 2, April 1986

016

CONTROLLING PENETRATION*

J. C. LATTANZIO†, J. J. MONAGHAN†, H. PONGRACIC† AND M. P. SCHWARZ†

Abstract. When particle methods are used to simulate the high Mach number collision of gas clouds particle streaming may occur because the particles can penetrate the interface between the clouds. We show how this penetration can be prevented by using an appropriate artificial viscosity. Numerical experiments in one and three dimensions are described.

Key words. hypersonic collision, particle methods, shock waves

1. Introduction. Elements of fluid do not stream through each other because molecular collisions annihilate the component of the velocity normal to the interface between the elements of fluid. This collision process takes place within a few molecular mean free paths and this length is negligible compared with a characteristic length of the system. If particle methods which take the velocity as a particle attribute, for example GAP (Marder (1975)) and SPH (Gingold and Monaghan (1977)), are used to model the collision between clouds of gas, streaming is generally negligible for low Mach number relative velocities, but it degrades the calculation when the relative velocities have a high Mach number. Hausman (1981) attempted to simulate the high Mach number collision of two gas clouds using Larson's (1978) particle method and found that substantial streaming occurred. For relative velocities with Mach numbers of ~20 he found that two clouds in a head-on collision passed through each other with minor hindrance. We found a similar result with the particle method SPH when a viscosity similar to that discussed by Monaghan and Gingold (1983) was used.

In this paper we show that it is possible to prevent streaming by using an appropriate artificial viscosity. The same viscosity also gives good results for shock tube phenomena.

2. The artificial viscosity. We use the particle method SPH but our results should be applicable to other particle methods. The inviscid equation of motion for particle i takes the form

$$\frac{d\mathbf{v}_i}{dt} = -\sum_j m_j\left(\frac{P_i}{\rho_i^2}+\frac{P_j}{\rho_j^2}\right)\nabla_i W_{ij}, \tag{2.1}$$

where subscripts denote the particle label, P is the pressure, ρ is the density, $W_{ij} \equiv W(|\mathbf{r}_i - \mathbf{r}_j|)$ is the interpolating kernel, ∇_i denotes the gradient taken with respect to the coordinates of particle i and m_j is the mass of particle j.

We begin by adding to (2.1) an artificial viscosity term similar to that used by Gingold and Monaghan (1983). Define $\mu(i,j)$ by

$$\mu(i,j) := \begin{cases} \dfrac{h\mathbf{v}_{ij}\cdot\mathbf{r}_{ij}}{c(r_{ij}^2+\eta^2)} & \text{if } \mathbf{v}_{ij}\cdot\mathbf{r}_{ij} \leqq 0, \\ 0 & \text{otherwise,} \end{cases} \tag{2.2}$$

where $\mathbf{v}_{ij} = \mathbf{v}_i - \mathbf{v}_j$, $\mathbf{r}_{ij} = \mathbf{r}_i - \mathbf{r}_j$, h is the interpolating length used in W_{ij}, c is the maximum speed of sound and $\eta \sim 0.1h^2$. μ is a nondimensional quantity which, in one dimension,

* Received by the editors September 3, 1984, and in revised form February 19, 1985.
† Department of Mathematics, Monash University, Clayton, Victoria 3168, Australia.

is approximately $(h/c)\,\partial v/\partial x$. We now replace (2.1) by

$$\frac{d\mathbf{v}_i}{dt} = -\sum_j m_j\left(\frac{P_i}{\rho_i^2}+\frac{P_j}{\rho_j^2}\right)(1-\alpha\mu(i,j))\nabla_i W_{ij}, \tag{2.3}$$

where α is a nondimensional constant ~ 7.

It is not difficult to show that in one dimension, in the limit as N (the number of particles) $\to\infty$ and $h\to 0$, equation (2.3) becomes

$$\frac{dv}{dt} = -\frac{1}{\rho}\frac{\partial P}{\partial x}+\frac{\alpha h}{c\rho}\frac{\partial}{\partial x}\left(P\frac{\partial v}{\partial x}\right). \tag{2.4}$$

The artificial viscosity is therefore equivalent, in one dimension, to a bulk viscosity. The finite difference equivalent of (2.3) results in an artificial viscosity similar to that devised by Landshoff (see Roache (1975)), and equivalent to the case $N=2$ of the artificial viscosity discussed by Van Leer (1969). The equation of motion (2.3) has several nice features. Linear and angular momentum are conserved exactly, it is invariant to translation and rotation of the coordinate system, and the viscosity is zero when the fluid is rotating uniformly. Furthermore it gives good results for low Mach number shock tube phenomena. However, as we remarked in the introduction, the artificial viscosity fails to stop streaming in high Mach number collisions.

To stop streaming, we need a term which will increase the effective pressure from $\sim\rho c_s^2$ to $\rho(\Delta v)^2$ where c_s is the sound speed and Δv is the normal component of the relative velocity. The artificial viscosity in (2.3) will only increase the effective pressure to $\sim\rho c_s|\Delta v|$. A simple way of achieving this result is to replace (2.3) by

$$\frac{d\mathbf{v}_i}{dt} = -\sum m_j\left(\frac{P_i}{\rho_i^2}+\frac{P_j}{\rho_j^2}\right)[1-\alpha\mu(i,j)+\beta\mu^2(i,j)]\nabla_i W_{ij} \tag{2.5}$$

where β is a nondimensional constant. The invariance properties of (2.5) are identical to those of (2.3). When velocity gradients are small (2.5) reduces to (2.3). When the velocity gradients are large the pressure terms are multiplied by $\sim(v_{ij}/c)^2$ which has the desired result of lifting the pressure to oppose the kinetic pressure $\rho(\Delta v)^2$ of the colliding gas clouds.

In the limit as $N\to\infty$ and $h\to 0$, equation (2.5) becomes in one dimension

$$\frac{dv}{dt} = -\frac{1}{\rho}\frac{\partial P}{\partial x}+\frac{\alpha h}{c\rho}\frac{\partial}{\partial x}\left(P\frac{\partial v}{\partial x}\right)-\frac{\beta}{\rho}\left(\frac{h}{c}\right)^2\frac{\partial}{\partial x}\left(P\left(\frac{\partial v}{\partial x}\right)^2\right), \tag{2.6}$$

which shows that the new viscosity has the effect of adding a von Neumann–Richtmyer viscosity to the equation of motion. It is therefore similar to an artificial viscosity suggested by Landshoff (see Roache (1975)). Because of the extra term compared with (2.3) the constant α, previously ~ 7, can be taken as ~ 1 and $\beta\sim 1$. The optimum values of these parameters depend on the interpolating kernel.

The numerical details involved in implementing (2.5) are the same as those discussed by Monaghan and Lattanzio (1984) except in respect of the time step. When the von Neumann–Richtmyer viscosity is used with standard Lagrangian finite difference schemes the time step is given by (Potter (1973))

$$\Delta t < \min_{\{\text{grid}\}}\frac{\Delta x}{c_s+b|\delta v|}, \tag{2.7}$$

where Δx is the cell width, δv is the velocity difference between two grid points and b is a constant. We also expect a time step criterion of this form but, since velocity

gradients appear via μ_{ij}, $|\delta v|$ will be replaced by a quantity related to μ_{ij}. We have found the following to lead to a stable algorithm

$$\Delta t = 0.3h \min_{\{\text{particles}\}} \frac{1}{\left(c_s + \frac{\zeta c_s^2}{c}\mu_m\right)}, \tag{2.8}$$

where $\zeta \gtrsim 0.6$ and μ_m is the maximum value of $|\mu(i,j)|$ for any particle. When body forces per unit mass **A** exist, we use the minimum of (2.8) and

$$0.3 \min_{\{\text{particles}\}} (h/|\mathbf{A}_i|)^{1/2}.$$

3. Numerical experiments. We have considered three sets of numerical experiments: shock tube, one-dimensional cloud collisions and three-dimensional cloud collisions. The shock tube experiments use the configuration employed by Monaghan and Gingold (1983). The cloud collision experiments use an isothermal equation of state since they are part of a larger experimental program designed to study the collision of interstellar clouds for which the isothermal approximation is acceptable.

3.1. Shock tube experiments. We use the artificial viscosity discussed here with the configuration used by Monaghan and Gingold (1983). This configuration results in a rarefaction wave moving to the left, a contact discontinuity and a shock moving to the right.

In Fig. 1, right frame, we show the density profile obtained using the super-gaussian kernel

$$W(u) = \frac{e^{-u^2/h^2}}{h\sqrt{\pi}}\left(\frac{3}{2} - \frac{u^2}{h^2}\right),$$

with $h = 0.015$, which is twice the initial particle separation (these calculations use particles with different masses, those with $x < 0$ have mass four times those with $x > 0$ so that they can have equal separation while producing a density ratio 4:1), and $\alpha = 0.5$ and $\beta = 0.5$. The results are in excellent agreement with analytical results. The shock front is reasonably narrow ($\sim 3h$) with no sign of post shock oscillations, and the contact discontinuity is well defined (width $\sim 1.5h$). If we take $\alpha = \beta = 1$ the differences are minor. For example the shock width is $\sim 4h$.

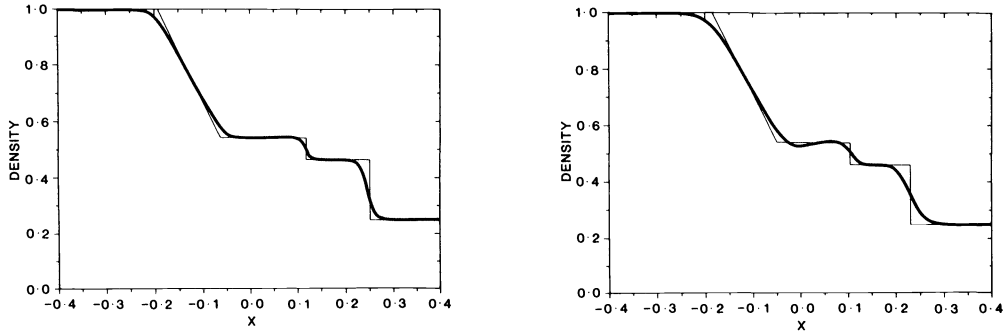


FIG. 1. *Left frame: Density profile for a shock tube problem with initial velocity zero, and initial density ratio 4:1. Initially* $P = A\rho^{1.4}$ *with* $A = 1$ *for* $x < 0$ *and* $A = 1.25$ *for* $x > 0$. *The kernel is gaussian with* $h = 0.015$. *SPH results shown*: ——. *Exact results shown*: ——. *Right frame: As before except that a super-gaussian has been used.*

In Fig. 1, left frame we show the same experiment using the ordinary gaussian kernel

$$W(u)=\frac{e^{-u^2/h^2}}{h\sqrt{\pi}},$$

with $h=0.015$ and $\alpha=\beta=1$. The main features of the experiment are recovered but, as expected, the accuracy is greatly inferior to that obtained using the super-gaussian. If we choose $\alpha=\beta=0.5$ there are post shock oscillations. With $\alpha=1$, $\beta=2.0$ the results are very similar to those with $\alpha=\beta=1$. With $\alpha=2$, $\beta=1$ the shock width is $\sim 6h$.

These results suggest that, for the super-gaussian kernel, a good choice is $\alpha=\beta=0.5$ while for the gaussian kernel a good choice is $\alpha=\beta=1.0$.

3.2. One-dimensional collisions. The configuration we consider consists of two identical gas clouds with equal and opposite velocities. The equation of state is $P=2.1\,\rho$ and the initial $\rho=1.0$. Two surfaces of discontinuity with opposite velocities propagate away from the contact surface leaving a region with increased density and zero velocity. In order to eliminate the consideration of edge effects we assume the clouds are infinite in extent. Let the left moving discontinuity surface have speed D and let the initial speed of the clouds be $\bar{v}$.

The mass flux and momentum flux across the left moving discontinuity are given by

$$\rho_1 v_1=\rho_2 v_2 \tag{3.1}$$

and

$$P_1+\rho_1 v_1^2=P_2+\rho_2 v_2^2, \tag{3.2}$$

where the subscript 1 denotes quantities to the left of the discontinuity and 2 to quantities to the right of it. The velocities are in the rest frame of the discontinuity so that

$$v_1=\bar{v}+D \quad \text{and} \quad v_2=D. \tag{3.3}$$

Denoting the isothermal sound speed by c we find, from (3.1)-(3.3), that

$$D=0.5(-\bar{v}+\sqrt{(\bar{v}^2+4c^2)}),$$

and

$$\rho_2/\rho_1=1+\bar{v}/D, \tag{3.4}$$

when $\bar{v}\gg c$, $D\sim c^2/\bar{v}$ and $\rho_2/\rho_1\sim\bar{v}^2/c^2$. The shock moves more slowly the larger the incoming speed of the gas.

For this case (isothermal one-dimensional motion) it is possible to calculate the viscous shock profile. The momentum flux (from (2.6)) is

$$P+\rho v^2-\frac{\alpha h}{c}P\frac{dv}{dx}+\beta\frac{h^2}{c^2}P\left(\frac{dv}{dx}\right)^2=B, \tag{3.5}$$

and the mass flux is

$$\rho v=j, \tag{3.6}$$

where j and B are constants. Using the isothermal equation of state, and eliminating

ρ from (3.5) and (3.6), we find

$$c^2+v^2-\alpha hc\left(\frac{dv}{dx}\right)+\beta h^2\left(\frac{dv}{dx}\right)^2=Fv, \tag{3.7}$$

where F is a constant. We can rewrite (3.7) in the form

$$\alpha hc\left(\frac{dv}{dx}\right)-\beta h^2\left(\frac{dv}{dx}\right)^2=(v-v_1)(v-v_2), \tag{3.8}$$

where v_1 and v_2 are the asymptotic velocities to the left and right of the shock. Comparison of (3.7) and (3.8) shows that $v_1v_2=c^2$ which, with (3.3), leads back to the expression above for D.

From (3.8) we find

$$h\frac{dv}{dx}=\frac{1}{2}\left(\frac{\alpha}{\beta}\right)c-\left[\frac{c^2\alpha^2}{4\beta^2}+\frac{(v_1-v)(v-v_2)}{\beta}\right]^{1/2}, \tag{3.9}$$

where the root of the quadratic for dv/dx has been chosen by recognizing that the material moving into the shock from the left will slow down and therefore $dv/dx<0$. Although (3.9) can be integrated analytically the resulting expression is cumbersome and not very informative. However, when $\beta\to 0$ it is easy to show that the shock has a thickness $\sim\alpha hc/\bar{v}$. In this case the shock becomes sharper as $\bar{v}$ is increased. When $\alpha\to 0$ the thickness of the shock is $\sim\sqrt{\beta}\,h\pi$ which is independent of $\bar{v}$. This last result is a well-known feature of the Von Neumann-Richtmyer viscosity. When we wish to compare profiles from our SPH simulation with exact viscous results we integrate (3.9) numerically and calculate ρ from (3.6) in the form $\rho=\rho_1v_1/v$.

In Fig. 2, we show the density and velocity profiles for an isothermal collision with $\bar{v}=1.0$. The SPH calculations use a super-gaussian kernel with $h=0.04$ and $\alpha=\beta=0.5$. The SPH results are in very good agreement with the discontinuous solution and with the exact viscous solution. There is no penetration. The shock width is $\sim 4h$. In Fig. 3, we show the profiles obtained using a gaussian kernel with $h=0.04$ and $\alpha=\beta=1.0$. The shock front is broader, as expected, but otherwise there is little difference between the super-gaussian and gaussian kernels for this problem. We have experimented with a wide range of other combinations of α and β for use with the gaussian kernel. With α and β given by $(\alpha,\beta)=(0.2,0.2)$, $(0.2,0.04)$ there were strong post shock oscillations. With $(\alpha,\beta)=(.5,1)$, $(.5,.5)$, $(.5,.25)$ the shock was sharper $(\sim 5h)$ than with $\alpha=\beta=1$ (the best for the shock tube). With $(1,0)$, $(1,1)$, $(1,2)$ the

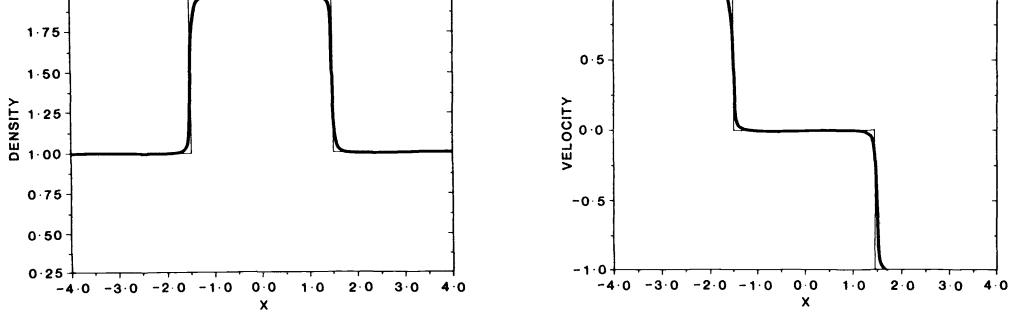


FIG. 2. *Isothermal collision* $P=2.1\rho$, *with initial velocity* $=1.0$, $\alpha=\beta=0.5$, *and* $h=0.04$. *A super-gaussian kernel has been used. Left frame density, right frame velocity. SPH results shown*: ——. *Exact results shown*: ——.

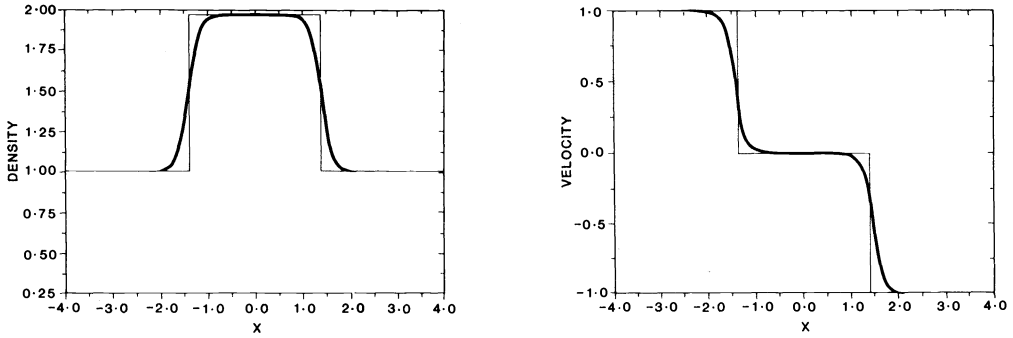


FIG. 3. *Isothermal collision* $P = 2.1\rho$, *with initial velocity* $= 1.0$, $\alpha = \beta = 1.0$, *and* $h = .04$. *A gaussian kernel has been used. Left frame density, right frame velocity.*

results were similar to the (1, 1) result shown in Fig. 2. With (2, 4), (2, 2), (2, 1) the shock was broadened over ~12h. These results are in good agreement with the exact viscous profile from (3.9).

The gaussian kernel gave stable results for $1 < \bar{v} < 10.0$. No tests were run outside this range. In Fig. 4 we show the density and velocity profiles for a one-dimensional, isothermal collision with $\bar{v} = 7.0$ using a super-gaussian kernel. The results are good,

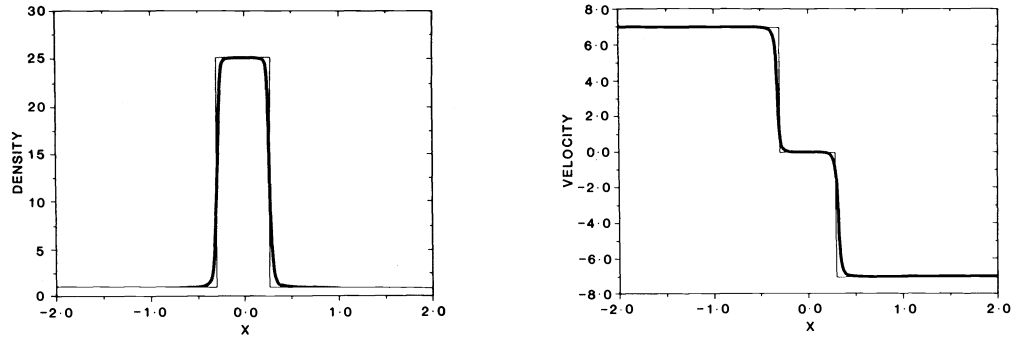


FIG. 4. *As in Fig.* 2 *except that initial velocity* $= 7.0$ *and* $\alpha = \beta = 1$.

but there are oscillations on a scale not discernible in the figure. As $\bar{v}$ increases the oscillations increase and a run with $\bar{v} = 10.0$ was unstable. One is immediately reminded of the problems experienced with nonmonotonicity finite difference algorithms since the super-gaussian kernel does not guarantee monotonicity. For example, in the case of interpolation of a step discontinuity, it produces a dip on the low side and a bump on the high side. We found a simple cure for the problem in our collision experiments. Since the problem occurred at shocks we could clearly predict trouble from the value of μ_{ij}. We therefore defined σ_{ij} by

$$\sigma_{ij} := \frac{|\mu_{ij}|}{1+|\mu_{ij}|},$$

and replaced $\nabla_i W_{ij}$ in (2.5) by

$$(1-\sigma_{ij})\nabla_i W_{ij}^{(1)} + \sigma_{ij}\nabla_i W_{ij}^{(2)},$$

where $W_{ij}^{(1)}$ is the super-guassian kernel and $W_{ij}^{(2)}$ is the gaussian. The result is that when $\sigma_{ij} \to 1$ (i.e. near strong shocks) the gradient is calculated using the gaussian while

elsewhere the super-gaussian is used. We confirmed that this change to the algorithm had negligible effect on the shock tube results for the super-gaussian described in § 3.1, while giving the good collision profiles shown in Fig. 5. We are experimenting with alternative forms for σ_{ij}.

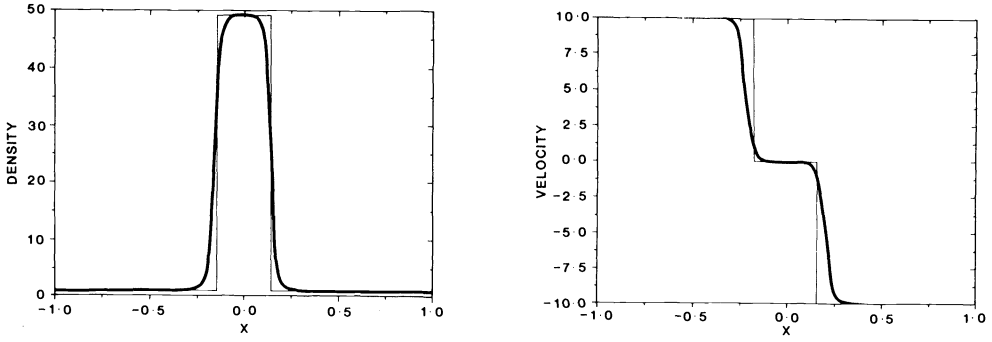


FIG. 5. *Collision profiles for initial velocity* = 10.0, $\alpha = \beta = 1$, $h = .04$ *and the switching prescription discussed at the end of* § 3.2 *has been used.*

3.3. Three-dimensional cloud collisions. We show in Fig. 6 the velocity field for a head on collision between two initially spherical clouds of equal mass and density. The clouds each have mass 361 $M_\odot$, temperature 80 °K, initial speed 6.75 km s^{-1}, Mach number 16.6, and initial radius 5 parsecs. Gravitational forces are included. The particles are arranged initially on a uniform grid, and only those within a specified radius are included in a given cloud. This results in a slightly nonspherical initial configuration. To prevent lines of particles from one cloud running into similar lines from the other cloud (thereby producing a series of largely independent one-dimensional collisions) we rotated one cloud by 45° relative to the other. This preserves left-right symmetry, but exact top-bottom symmetry is then lost. This shows up particularly at the high compression stage. The result of the calculation is that each cloud behaves as if it has run into a virtually impenetrable wall (there is slight penetration over a distance of $\sim h$). A wide variety of other collisions using different masses, and nonzero impact parameters confirm the reliability of the SPH collision calculations with the new viscosity.

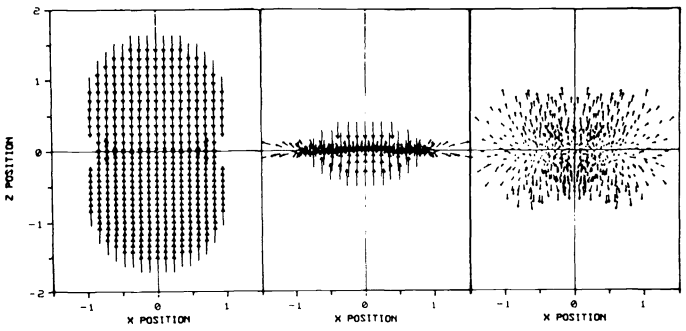


FIG. 6. *The velocity field in the x–z plane of a three-dimensional head-on collision along the z axis of two isothermal clouds. Details are given in the text.* 3544 *particles have been used. Left frame: Velocity field in the initial compression phase. Middle frame: Velocity field near maximum compression. Right frame: The re-expansion phase. Similar results have been obtained with as few as* 66 *particles in each cloud.*

4. Discussion and conclusions. We can briefly summarize our results as follows. The artificial viscosity included in the momentum equation (2.5) gives good results for both shock tube and high Mach number isothermal cloud collisions. In the latter it reduces penetration to $\lesssim$ one resolution length. The viscosity vanishes for rigid rotation, is galilean invariant and when (2.5) is used conserves linear momentum. Whether or not it conserves angular momentum depends on the form of the interpolating kernel W_{ij}. The parameters α and β in (2.5) depend on the interpolating kernel used, but for a given kernel the qualitative features of the solution are not sensitive to their precise values. Once a near optimal choice of α and β has been made (α equal to β is a good rule of thumb) from shock tube phenomena it can also be used for collision phenomena at low Mach numbers ($\lesssim 5$). For higher Mach numbers larger values of α and β are preferable. The reader is reminded that super-gaussian kernels show an instability for high Mach number (~ 10) collisions (see § 3.2).

The particle methods we have described here can be applied effectively to a wide class of problems in astrophysical hydrodynamics. This class includes collision problems in three dimensions which lead to long narrow features which are diffused out by most finite difference schemes or are handled inefficiently because very large grids are needed to resolve these features. Difficult multi-media impact problems of the kind which are normally simulated by PIC can also be simulated (with greater accuracy because there is no grid-particle interpolation) using our particle method. Good finite difference methods (e.g. Harten (1984)) are, however, always preferable to particle methods for one-dimensional problems and many two-dimensional problems.

REFERENCES

R. A. GINGOLD AND J. J. MONAGHAN, *Smoothed particle hydrodynamics: theory and application to non-spherical stars*, Mon. Not. Roy. Astron. Soc., 181 (1977), pp. 375–389.

M. A. HAUSMAN, *Collisional mergers and fragmentation of interstellar clouds*, Astrophys. J., 245 (1981), pp. 72–91.

R. B. LARSON, *A finite particle scheme for three-dimensional gas dynamics*, J. Comput. Phys., 27 (1978), pp. 397–409.

J. C. LATTANZIO, J. J. MONAGHAN, H. PONGRACIC AND M. P. SCHWARZ, *Cloud collisions*, Mon. Not. Roy. Astron. Soc., 1985, in press.

B. M. MARDER, GAP-A PIC-*type fluid code*, Math. Comp., 29 (1975), pp. 434–446.

J. J. MONAGHAN AND R. A. GINGOLD, *Shock simulation by the particle method* SPH, J. Comput. Phys., 52 (1983), pp. 374–389.

J. J. MONAGHAN AND J. C. LATTANZIO, *A refined particle method for astrophysical problems*, Astron. Astrophys., 1985, in press.

D. POTTER, *Computational Physics*, John Wiley, New York, 1980.

P. J. ROACHE, *Computational Fluid Dynamics*, Hermosa Pub., Albuquerque, NM, 1975.

B. VAN LEER, *Stabilization of difference schemes for the equations of inviscid compressible flow by artificial diffusion*, J. Comput. Phys., 3 (1969), pp. 473–485.

SIAM J. SCI. STAT. COMPUT.
Vol. 7, No. 2, April 1986

017

THE PSEUDO-SPECTRAL METHOD AND PATH FOLLOWING IN REACTION-DIFFUSION BIFURCATION STUDIES*

J. C. EILBECK†

Abstract. In an earlier paper, the pseudo-spectral method was advocated as a fast and efficient method for studying the time evolution of solutions of reaction-diffusion problems in certain cases. In this paper we extend the method to follow steady-state solutions as a function of the problem parameters, using path-following techniques. As a specific example the method is applied to a boundary value problem of the form $u_t = u_{xx} + \alpha g(u)$, where $g(u)$ is a cubic polynomial.

Key words. pseudo-spectral method, reaction-diffusion equations, bifurcation diagrams, path-following techniques, boundary value problem

AMS (MOS) subject classifications. 65N35, 35K20

1. Introduction. In this paper we combine pseudo-spectral (collocation) methods (Gottlieb and Orszag (1977), Eilbeck (1983)) with the path following techniques of Keller et al. (cf. Keller (1979), Decker and Keller (1981)) and Kubicek (1976) to provide a simple and efficient method to study bifurcation diagrams for steady-state solutions of reaction-diffusion equations.

Consider a general set of reaction-diffusion equations in a finite region Ω

$$\mathbf{u}_t(\mathbf{x}, t) = d\Delta\mathbf{u}(\mathbf{x}, t) + F(\mathbf{u}, \alpha_1, \alpha_2, \cdots, \alpha_m). \tag{1.1}$$

Here $x \in \Omega \subset \mathbb{R}^n$, $n = 1, 2, 3$; $\mathbf{u} \in \mathbb{R}^k$; d is a $k \times k$ matrix of diffusion coefficients, $\alpha_1, \alpha_2, \cdots, \alpha_m$ are reaction parameters, and F is a nonlinear function $F: \mathbb{R}^k \times \mathbb{R}^m \to \mathbb{R}^k$. The methods we describe below can easily be extended to cover this general case (c.f. Eilbeck (1983)), but for clarity and ease of exposition we shall confine ourselves to the simple case $n = k = m = 1$. More complicated systems in higher space dimensions will be considered in later papers. We also assume a particularly simple dependence on the single parameter $\alpha' = \alpha_1$, so that (1.1) becomes

$$u_t(x, t) = u_{xx}(x, t) + \alpha' F(u), \qquad x \in [-L, L]. \tag{1.2}$$

Typical boundary conditions might be $u = U_0$ for $x = -L, L$ or $\partial u/\partial x = 0$ for $x = -L, L$. We assume the former, and transform the dependent variable to $v = u - U_0$. If $F(U_0) = 0$, then $v(x, t) = 0$ is always one possible solution. It is also convenient to normalize the space variable x so that $[-L, L]$ is transformed into $[0, 1]$. With these scalings (1.2) becomes

$$v_t(x, t) = v_{xx}(x, t) + \alpha g(v), \qquad t \geqq 0, \quad x \in [0, 1], \tag{1.3a}$$

$$v(0, t) = v(1, t) = 0. \tag{1.3b}$$

Here $g(v) = F(v + U_0)$ and if $g(0) = 0$, one solution of (1.3) is $v(x, t) = 0$.

Our aim is to study the number and nature of the solutions of (1.3) as α varies. A major tool for this study is the path following methods which have been developed by Keller and co-workers over the last few years (Keller (1979), Decker and Keller (1981)) and applied to a number of problems in fluid mechanics (Lentini and Keller (1980), Mayer-Spasche and Keller (1980), Schreiber and Keller (1983)). The basic

* Received by the editors February 2, 1984, and in final form December 26, 1984.

† Center for Nonlinear Studies, Los Alamos National Laboratory, Los Alamos, New Mexico 87545. Permanent address, Department of Mathematics, Heriot-Watt University, Riccarton, Edinburgh EH14 4AS, United Kingdom.

idea behind this method, that of using arc-length as a continuation parameter, was also suggested independently by Kubicek (1976).

When applied to infinite-dimensional problems, these path-following techniques require an (unspecified) numerical approximation to convert the original system into a finite-dimensional one. In the fluid problems already studied, both finite difference (Schreiber and Keller (1983)) and spectral methods (Meyer-Spasche and Keller (1980)) have been used. Kernevez et al. (1983) have applied these techniques to reaction-diffusion systems using finite element and finite difference methods.

Spectral methods involve expanding the approximate solutions as a finite sum of eigenfunctions of one of the operators in the problem, then imposing orthogonality conditions (i.e. Galerkin's method) to obtain sets of ordinary differential equations for the coefficients of the spectral functions. When the nonlinearities involved in the problem are relatively simple, this approach works well: however for more complicated nonlinearities it may be more appropriate to use a pseudo-spectral method, i.e. spectral functions plus the collocation method. This is the approach used in this paper, although some comparisons with the pure spectral method are made.

The paper is laid out as follows: in § 2 the pseudo-spectral method is briefly reviewed, with especial emphasis on stationary solutions. The appropriate equations for path following are introduced in § 3, and in § 4 we consider the stability of the resulting solutions. In § 5 we apply the method to a specific example which has already been the subject of much study.

2. The pseudo-spectral method. We follow the same approach as Eilbeck (1983). To derive the spectral or pseudo-spectral method, the solution $v(x, t)$ to (1.3a) is approximated by a function $\tilde{v}(x, t)$ which is a *finite* sum of eigenfunctions of the operator Δ satisfying the boundary conditions (1.3b)

$$v(x, t) \cong \tilde{v}(x, t) = \sum_{j=1}^{N} c_j(t)\phi_j(x), \tag{2.1}$$

where the $\phi_j(x)$ given by

$$\Delta\phi_j(x) = \lambda_j\phi_j(x)$$

are the eigenvalues of the Laplacian subject to the boundary conditions

$$\phi_j(0) = \phi_j(1) = 0.$$

In the one-dimensional case these eigenvalues and eigenfunctions are simply $\lambda_j = -\pi^2 j^2$ and $\phi_j(x) = \sin \pi jx$. The importance of this particular choice of basis functions for the reaction diffusion equations (1.3) is that linear bifurcation theory is reproduced *exactly* in the following sense. If $g(v) = 0$ so that $v(x, t) = 0$ is a solution, and if we consider bifurcations from the zero function, then the position of the first N primary bifurcation points will be given exactly by the numerical approximation. Also any *small* perturbation of the form

$$\delta v(x, 0) = \sum_{j=1}^{N} c_j(0)\phi_j(x)$$

which evolves for small times according to

$$\delta v(x, t) = \sum_{j=1}^{N} c_j(0)\, e^{\lambda_j t}\phi_j(x)$$

will also be treated *exactly* by the numerical method (up to the accuracy of integration

of the resulting ordinary differential equations, and up to such times as nonlinear effects become important).

Since linear theory is reproduced exactly in this sense, the hope is that the nonlinear theory will be approximated well also, even in cases where N is small. This question is discussed further by Eilbeck (1983), and numerical experiments confirm this in many cases. Note that other numerical schemes such as finite difference or finite element methods do *not* reproduce linear theory exactly except in the limit $N \to \infty$ (Beyn (1980), Mittelman and Weber (1980)), so for small N we would not expect them to be so accurate in the nonlinear case. The same comment applies to other choices of spectral functions, such as Chebyshev polynomials.

Once the approximation (2.1) has been made, we insert this into (1.3a) to give

$$\sum_{j=1}^{N} \dot{c}_j(t)\phi_j(x) = -\sum_{j=1}^{N} \pi^2 j^2 c_j(t)\phi_j(x) + \alpha g(\tilde{v}) + r(x, t). \tag{2.2}$$

Here the residual function $r(x, t)$ has been inserted to make (2.1) exact. In the pure spectral method we make $r(x, t)$ small by imposing the condition that $r(x, t)$ be orthogonal to the N basis functions $\phi_1(x), \cdots, \phi_N(x)$

$$\begin{aligned} \int_0^1 r(x, t)\phi_i(x)dx = 0 = \sum_{j=1}^{N} \dot{c}_j(t) \int_0^1 \phi_i(x)\phi_j(x)\, dx \\ + \pi^2 \sum_{j=1}^{N} j^2 c_j(t) \int_0^1 \phi_i(x)\phi_j(x)\, dx \\ - \alpha \int_0^1 g(\tilde{v})\phi_i(x)\, dx. \end{aligned} \tag{2.3}$$

If we define $\tilde{a}_{ij} = \int_0^1 \phi_i(x)\phi_j(x)\, dx$, $\tilde{b}_{ij} = \pi^2 j^2 \tilde{a}_{ij}$, and

$$\tilde{G}_i(\mathbf{c}) = \alpha \int_0^1 g\left(\sum_{j=1}^{N} c_j(t)\phi_j(x)\right)\phi_i(x)\, dx, \qquad i = 1, \cdots, N, \tag{2.4}$$

with $\mathbf{c}(t) = (c_1(t), c_2(t), \cdots, c_N(t))^T$ (the superscript T denoting the transpose), then we have a diagonal set of o.d.e.'s for the vector $\mathbf{c}(t)$.

$$\tilde{A}\dot{c}(t) = \tilde{B}\mathbf{c}(t) + \tilde{G}(\mathbf{c}(t)). \tag{2.5}$$

The problem is that, except in very simple cases, the integrals in (2.3) cannot be evaluated analytically and must be performed numerically. (It should be pointed out that the example considered in § 4 is one such simple case.) For this reason we prefer a more straight-forward method based on the collocation approach: we make $r(x, t)$ small by applying the condition that the residual function vanishes at a set of N *collocation* points $x_1, x_2, \cdots, x_n$.

$$r(x_i, t) = 0 = \sum_{j=1}^{N} \dot{c}_j(t)\phi_j(x_i) + \pi^2 \sum_{j=1}^{N} c_j(t) j^2 \phi_j(x_i) - \alpha g(\tilde{v}(x_i)). \tag{2.6}$$

If we define the matrices $A = \{a_{ij}\} = \{\phi_j(x_i)\}$ and $B = \{b_{ij}\} = \{\pi^2 j^2 a_{ij}\}$, and the vector $\tilde{\mathbf{v}}(t) = (\tilde{v}(x_1, t), \tilde{v}(x_2, t), \cdots, \tilde{v}(x_N, t))^T$ given by

$$\tilde{\mathbf{v}}(t) = A\mathbf{c}(t), \tag{2.7}$$

then (2.6) can be written as a set of implicit o.d.e.'s for the coefficients $c_i(t)$.

$$A\dot{\mathbf{c}}(t) + B\mathbf{c}(t) - \alpha \mathbf{g}(\tilde{\mathbf{v}}) = 0 \tag{2.8}$$

where $g_i(\tilde{\mathbf{v}}) = g(\tilde{v}(x_i, t))$. The fact that the matrix $\tilde{A}$ in the spectral method equations (2.5) is diagonal whereas A in the pseudo-spectral equations (2.8) is nondiagonal is not important in view of the fact that we will be considering only steady-state solutions $\dot{c}(t) = 0$.

The optimal choice of collocation points for the pseudo-spectral method is still an open problem (Eilbeck (1983)). However, in one space dimension, a result in approximation theory (de Boor and Pinkus (1978)) suggests that the choice of equally spaced points should be a good one, i.e. $x_i = i/(N+1)$, $i = 1, 2, \cdots, N$. We have followed this choice in the numerical example in § 5. For time-independent solutions the pseudo-spectral method gives us the N nonlinear simultaneous equations

$$G(\mathbf{c}, \alpha) \equiv B\mathbf{c} - \alpha \mathbf{g}(\tilde{\mathbf{v}}) = 0. \tag{2.9}$$

(Recall that $\tilde{\mathbf{v}}$ is given in terms of $\mathbf{c}$ by (2.7).) Solving (2.9) for a range of values of the parameter α is the subject of the next section. Once a solution $\mathbf{c}(\alpha)$ is known, (2.7) gives the appropriate solution $\tilde{v}(x_i)$ at the collocation points: however we can calculate $\tilde{v}(x)$ for *any* x value using (2.1).

3. Path following methods. In this section we follow closely the work of Keller (1979) and Decker and Keller (1981). The nonlinear equations (2.9) for the steady state solutions using the pseudo-spectral method (or the corresponding equations from the spectral method) are solved for a range of values of α. Assume a solution $\mathbf{c}(\alpha_0)$ is known for some α_0. Away from singular points (where the matrix $\partial G_i/\partial c_j$ is singular), we can solve for $\alpha_0 + \Delta\alpha$ by a combined Euler predictor and a Newton iteration:

$$\begin{aligned}
&\text{(a)} \quad \text{Solve } G_c(\mathbf{c}(\alpha_0), \alpha_0)\mathbf{c}_\alpha(\alpha_0) = -G_\alpha(\mathbf{c}(\alpha_0), \alpha_0) \text{ for } \mathbf{c}_\alpha(\alpha_0),\\
&\qquad \alpha = \alpha_0 + \Delta\alpha,\\
&\qquad \mathbf{c}^0(\alpha) = \mathbf{c}(\alpha_0) + \Delta\alpha\,\mathbf{c}_\alpha(\alpha_0);\\
&\text{(b)} \quad G_c(\mathbf{c}^\nu(\alpha), \alpha)\delta\mathbf{c}^{\nu+1}(\alpha) = -G(\mathbf{c}^\nu, \alpha), \qquad \mathbf{c}^{\nu+1} = \mathbf{c}^\nu + \delta\mathbf{c}^{\nu+1},\\
&\qquad \nu = 0, 1, 2, \cdots \text{ until } \|\delta\mathbf{c}^{\nu+1}(\alpha)\| < \varepsilon.
\end{aligned} \tag{3.1}$$

Here G_c and G_α are the Frechet derivatives of G with respect to $\mathbf{c}$ and α, step (a) is the Euler predictor, and step (b) is the Newton iteration. For the system of equations (2.9), we can easily calculate the elements of G_c and G_α

$$\frac{\partial G_i}{\partial c_j} = b_{ij} - \alpha \left.\frac{\partial g}{\partial v}\right|_{v=v_i} \frac{\partial v_i}{\partial c_j} = b_{ij} - \alpha g'(v_i) a_{ij}, \qquad \frac{\partial G_i}{\partial \alpha} = -g(v_i). \tag{3.2}$$

It is important to note that the matrix $G_c = \partial G_i/\partial c_j$ will be a full matrix for both the spectral and pseudo-spectral methods, whereas for finite difference or finite element methods the corresponding matrices will be sparse (banded). This suggests that for very large N, the spectral methods may not be competitive with such methods. In the spirit of the discussion following (2.1) we therefore advocate the spectral methods as most appropriate in the case where moderate accuracy is required and N is small. As we shall see in the next section, we can get useful results with these methods even in the case $N = 1$. As N increases, there is likely to be a break even point, where it may be more economic to switch to finite difference or finite element methods.

Near singular points (det $G_c = 0$), the continuation method (3.1) breaks down, and it is necessary to adopt a pseudo-arclength procedure. Now α is considered to be a function of a parameter s, and we adjoin to (2.9) an extra equation to determine the

normalization of s.

$$\begin{aligned} &G(\mathbf{c}, \alpha) = 0, \\ &N(\mathbf{c}, \alpha, s) \equiv \dot{\mathbf{c}}^T(s_0)(\mathbf{c}(s) - \mathbf{c}(s_0)) + \dot{\alpha}(s_0)[\alpha(s) - \alpha(s_0)] - [s - s_0] = 0. \end{aligned} \tag{3.3}$$

At $s = s_0$, $\alpha_0 = \alpha(s_0)$ and $\mathbf{c}(s_0) = \mathbf{c}(\alpha(s_0))$ satisfy $G(\mathbf{c}, \alpha) = 0$ and $\dot{\mathbf{c}}(s_0)$ and $\dot{\alpha}(s_0)$ ($\partial \mathbf{c}/\partial s$ and $\partial \alpha/\partial s$) are calculated from

$$G_c(\mathbf{c}(s_0), \alpha_0)\dot{\mathbf{c}}(s_0) + G_\alpha(\mathbf{c}(s_0), \alpha_0)\dot{\alpha}(s_0) = 0, \tag{3.4a}$$

$$\|\dot{\mathbf{c}}(s_0)\|^2 + \dot{\alpha}(s_0)^2 = 1. \tag{3.4b}$$

The ambiguity in sign in solving (3.4b) reflects the choice of direction along the arc length: once this has been fixed initially, the signs of $\dot{\mathbf{c}}(x)$ and $\dot{\alpha}(x)$ further along the path are chosen to satisfy

$$\dot{\mathbf{c}}(s_0)^T\dot{\mathbf{c}}(s) + \dot{\alpha}(s_0)\dot{\alpha}(s) > 0.$$

Once $\dot{\mathbf{c}}(s_0)$ and $\dot{\alpha}(s_0)$ are known, initial (Euler) predictions for $\mathbf{c}^0(s)$ and $\boldsymbol{\alpha}^0(s)$, $(s = s_0 + \Delta s)$, are calculated by

$$\begin{aligned} \mathbf{c}^0(s) &= \mathbf{c}(s_0) + \Delta s\dot{\mathbf{c}}(s_0), \\ \alpha^0(s) &= \alpha(s_0) + \Delta s\dot{\alpha}(s_0), \end{aligned} \tag{3.5}$$

and then the inflated system (3.3) is solved by a Newton iteration analogous to (3.1b). Even at points where G_c is singular, the Jacobian of the inflated system may be nonsingular. This will occur at "normal limit points" where the solution becomes double valued (see § 5 for examples).

At bifurcation points the full Jacobian becomes singular: more detailed tests are necessary to calculate the slopes of the bifurcating arcs at such points and to follow the solution along either branch. However it is often possible to "shoot through" the bifurcation point without further calculations, if it is not necessary to change branches at this point. More details will be found in Decker and Keller (1981). One small comment about this procedure is in order. If we are investigating bifurcations from the zero solution $v = 0$, then $G_\alpha = 0$ for our model problem. In this case the vector ϕ_0 defined in equation (2.9) in Decker and Keller (1981) is zero, and equations (2.13b), (3.23b) in that paper should be replaced by

$$\xi_0^2 + \xi_1^2 = 1 \tag{3.6}$$

with corresponding changes in some other equations in that paper. It is perhaps better to replace (3.6) for our simple system with the equation

$$\xi_0^2(1 + \|\phi_0\|_2^2) + \xi_1^2\|\phi_1\|_2^2 = 1 \tag{3.7}$$

which avoids the need to construct equivalent norms.

In practice, the occurrence of a normal limit point or bifurcation point can be monitored by checking for changes of sign in the determinant of the Jacobian G_c. The stability of observed solutions can also be determined indirectly from the Jacobian, as discussed in the next section.

4. Stability of solutions. The (linear) stability of the approximate solutions $\mathbf{c}(\alpha)$ (and hence $\tilde{\mathbf{v}}(\alpha)$) can be examined as follows. Consider the full evolution equations (2.8) for $\mathbf{c}(t, \alpha)$, linearized about a steady-state solution $\mathbf{c}(0, \alpha)$ found by the procedures outlined in earlier sections

$$A\dot{\mathbf{c}} = -G_c(\mathbf{c}(0, \alpha), \alpha)\mathbf{c}(t, \alpha).$$

Since A is nonsingular, we can write

$$\dot{\mathbf{c}} = -A^{-1}G_c(\mathbf{c}(0, \alpha), \alpha)\mathbf{c}(t, \alpha).$$

Clearly $\mathbf{c}(0, \alpha)$ will be linearly stable if all the eigenvalues of the matrix $-A^{-1}G_c(\mathbf{c}(0, \alpha), \alpha)$ are distinct and have real parts less than zero.

In practice the eigenvalues only need to be calculated once in between each bifurcation point or normal limit point, since (in the absence of Hopf bifurcations) these points are the only ones at which an eigenvalue passes through zero.

5. Numerical example. For our numerical example we consider an equation studied by Smoller and Wasserman (1981) and more recently by Manorajan et al. (1984) and Manoranjan (1984)

$$u_t = u_{xx} + u(1-u)(u-a), \qquad 0 < a < \tfrac{1}{2}, \tag{5.1a}$$

With the boundary condition

$$u(-L, t) = u(L, t) = b. \tag{5.1b}$$

A simple change of variables puts this into the standard form (1.3)

$$v_t = v_{xx} + \alpha g(v), \tag{5.2a}$$

$$g(v) = -v^3 + (1 + a - 3b)v^2 + (2ab - 3b^2 + 2b - a)v + b(1-b)(b-a), \tag{5.2b}$$

$$v(0, t) = v(1, t) = 0, \qquad v = u - b, \tag{5.2c}$$

where $\alpha = 4L^2$ is the bifurcation parameter. We examine first in some detail the case $a = b = 0.25$, since this case has been the most studied (Manoranjan (1984)). As described above we calculate the steady-state solutions ($v_t = 0$). In all cases we plot $v(x)$ for $x \in [0, 1]$ instead of u.

5.1. The case a = b = 0.25. In this case $g(v) = 0$ so $v(x) = 0$ is a solution for all α. Linear stability analysis shows that solutions bifurcate from the zero solution at $\alpha = j^2\pi^2/a(1-a)$ for $j = 1, 2, \cdots$. In the papers by Manoranjan et al., attention was directed at the first bifurcation point ($j = 1$) and the solution curves bifurcating from this point. We have also included the second point ($j = 2$) in our study.

In order to prove an "exact" solution for the nonlinear problem, we solved (2.9) for a large value of N (401) using the methods of § 3 to provide a base-line for our error calculations. (Remember that we do not necessarily advocate the method as particularly efficient for these large values of N, as discussed in § 3.) The calculation was then repeated for $N = 201, 101, 51, 21, 11, 5$ and 1. Odd values of N were chosen to give a collocation point at $x = 0.5$ where v often has an extremum value. The resulting bifurcation curves for $N = 51$ are shown in Fig. 1. In this figure, v^* is calculated as the $\tilde{v}_i$ such that $|\tilde{v}_i| \geqq |\tilde{v}_j|$ for $j = 1, \cdots, N$. Plots of solutions $v(x)$ corresponding to the points (a)–(h) are shown in Fig. 2. In both figures, unstable solution curves or solutions are shown by dashed lines. Point (c) corresponds to $\alpha = \alpha_{\min} = 39.8736$, ($L = 3.1573$) the normal limit point at which $\partial v^*/\partial\alpha = 0$, and v^* at this point has the value 0.2833. Asymptotically the top curve $\to 0.75$ as $\alpha \to \infty$, and both bottom curves $\to -0.25$ as $\alpha \to \infty$. These values correspond to the other zeros of $g(v)$ for $a = 0.25$. The top curve of solutions from point (c) upwards are stable, as is the first lower curve from $\alpha = \alpha_1 = \pi^2/a(1-a) = 52.6779$ downwards. Between these two points the solution is unstable. The second branch, from $\alpha = \alpha_2 = 4\pi^2/a(1-a) = 210.5516$ is unstable for all values of α. This branch is double valued corresponding to the broken symmetry $x \to 1 - x$.

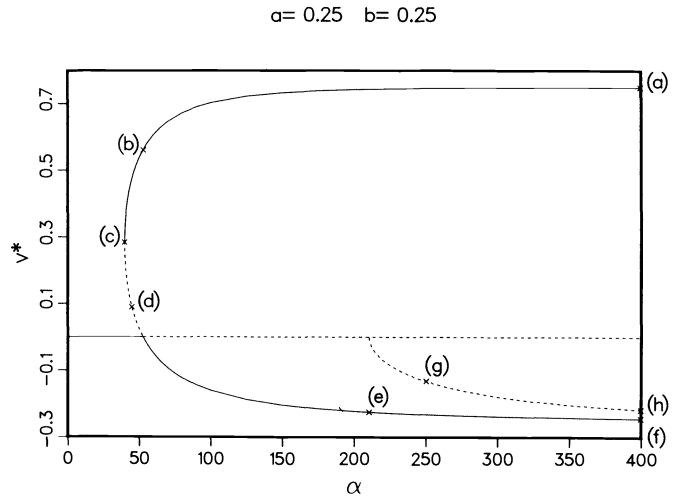


FIG. 1. *Solution curves bifurcating from the first two primary bifurcation points for the model problem* (5.2) *with* $a = b = 0.25$. *Unstable solutions are represented by dashed lines.*

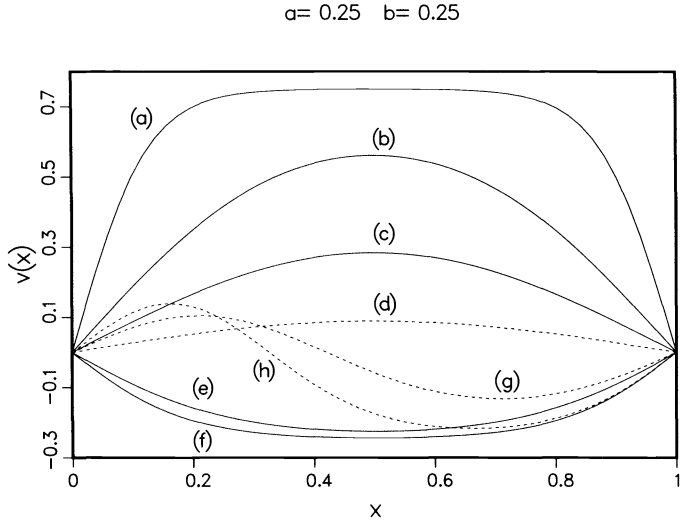


FIG. 2. *Solutions* $v(x)$ *corresponding to the points* (a)-(h) *in Fig.* 1.

In order to compare errors for various values of N, we examine the error in v^* at $\alpha = 400$ and $\alpha = \alpha_1$, and the error in calculating $\alpha_{\min}$, taking the "exact" values to be those given by the numerical approximation at $N = 401$. The results are displayed in Table 1.

We see that even with the moderately small value of $N = 21$ it is possible to obtain almost 6 figure accuracy in the calculated solution.

Some comparisons with other results are of interest, especially for low values of N. The third order perturbation theory in Manoranjan (1984) gives $\alpha_{\min} = 39.7755$ ($L = 3.1534$), with an error of 9.8E−2, better than the pseudo-spectral result with $N = 1$, but not as good as the case for $N = 5$. However the pseudo-spectral result for $N = 1$ gives a simple approximation for the whole of the first curve:

$$-(v^*)^2 + \frac{1}{2}v^* + \frac{3}{16} = \frac{\pi^2}{\alpha}. \tag{5.3}$$

TABLE 1

N	$v^*(400)$	Errors in $v^*(\alpha_1)$	$\alpha_{\min}$
201	1.0E−10	4.0E−10	1.3E−8
101	9.0E−10	6.4E−9	2.1E−7
51	1.3E−8	9.4E−8	3.2E−6
21	9.7E−7	3.0E−6	1.0E−4
11	3.6E−5	3.4E−5	1.1E−3
5	2.0E−3	6.5E−4	1.9E−2
1	2.5E−2	6.1E−2	4.0E−1

This gives $\alpha_{\min}$ to within about 1% relative accuracy, and has the correct asymptotic limits (0.25 ± 0.5) $\alpha \to \infty$. In addition the first bifurcation point α_1 is given exactly, as noted in § 2.

Since $g(v)$ in this case is a polynomial the integrals (2.3) in the spectral method can be calculated exactly. It is straightforward to show that in this case the corresponding approximation to the first curve for $N = 1$ is given by

$$-(v^*)^2 + \frac{16}{9\pi} v^* + \frac{1}{4} = \frac{4\pi^2}{3\alpha}. \tag{5.4}$$

For this case the approximation to $\alpha_{\min}$ is 39.8704 ($L = 3.1572$), with absolute (relative) error 3.2E−3 ($\cong$.008%). Thus for $N = 1$ the spectral method is much more accurate at this point than the pseudo-spectral method. However in the limit $\alpha \to \infty$ the spectral method has the wrong asymptotic limits (.283 ± .574). This reflects the fact that the spectral method is trying to minimize an average error rather than a pointwise error in these cases.

Apart from the anomalous asymptotic result, we would expect in general that for fixed N, the spectral method will always be more accurate than the pseudo-spectral method, in cases where the integrals in (2.3) can be calculated exactly. The pseudo-spectral method is more useful as a general-purpose method since a change in $g(v)$ requires only a small modification of the code.

Finally a calculation using a simple finite difference method with $h = \frac{1}{2}(N = 1)$ gives a much cruder approximation to the first bifurcation curve

$$-(v^*)^2 + \frac{1}{2} v^* + \frac{3}{16} = \frac{8}{\alpha}. \tag{5.5}$$

This gives $\alpha_1 = 42.666$, an absolute error of about 10 for a quantity given *exactly* by the spectral and pseudo-spectral methods. The corresponding result for $\alpha_{\min}$ is 32, with an error of about 8 units. However correct asymptotic limits in the case $\alpha \to \infty$ are obtained again in this case.

Calculations of the *slope* of the first bifurcation curve at $\alpha = \alpha_1$ using the pseudo-spectral method also show good agreement with the theoretical value calculated according to the methods of Decker and Keller (1981).

5.2. Some examples with $b \neq a$. In order to facilitate comparisons with other numerical results, we consider various other values of a and b treated by Manoranjan et al. (1984).

(i) $b = 0$. Solution curves for the case $a = 0.25$, $b = 0$, and $0 \leqq \alpha \leqq 400$ ($0 \leqq L \leqq 10$) are shown in Fig. 3. The zero solution and the upper curve for $\alpha \geqq 79.8543$ ($L \geqq 4.4681$)

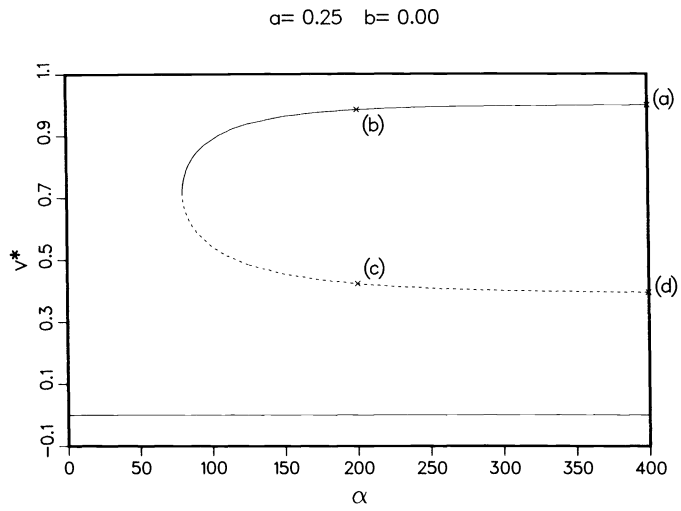


FIG. 3. *The case* $a = 0.25$, $b = 0.0$. *Note that curves of unstable solutions are denoted by dashed lines.*

are stable. No bifurcations from the zero solution or from the upper curves were observed for $\alpha \leqq 400$. Solutions for the points labeled (a)-(d) are shown in Fig. 4.

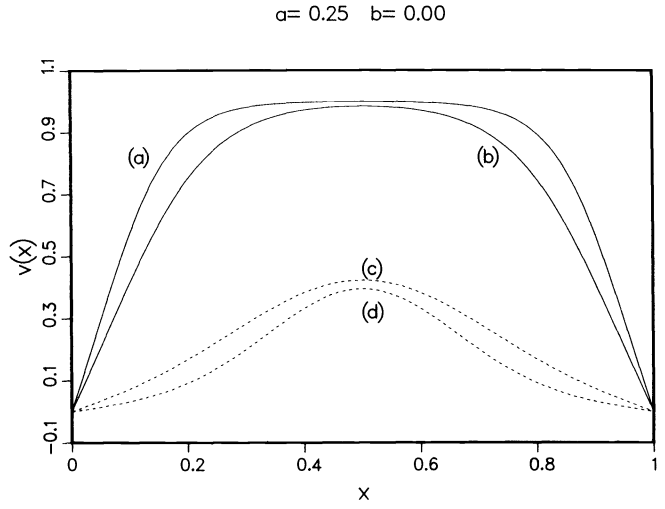


FIG. 4. *Solutions* $v(x)$ *corresponding to the points* (a)-(d) *in Fig.* 3.

(ii) $0 < b < a$. Calculations for the case $a = 0.25$, $b = 0.1$, are summarized in Figs. 5 and 6. The curve $v = 0$ is no longer a solution and has been replaced by a negative solution with a minimum at $x = 0.5$ (see curve (d) in Fig. 6). The upper branches are similar to the previous case, except there is now a secondary bifurcation point at $\alpha \cong 264.4006$ ($L \cong 8.1302$) at which a solution with unsymmetric components branches off. However this double solution curve has only unstable solutions. The top (stable branch starts at $\alpha \cong 63.7075$ ($L \cong 3.9908$).

(iii) $b = a$. The case $b = a = 0.25$ has already been discussed above.

(iv) $a < b < 1$. For $a = 0.25$, $b = 0.5$, we find only one solution curve in the range $0 \leqq \alpha \leqq 400$. This curve is similar to the top curve in Fig. 7. If a is increased to 0.4, a second group of solution curves enters the diagram, as shown in Fig. 7. The bottom

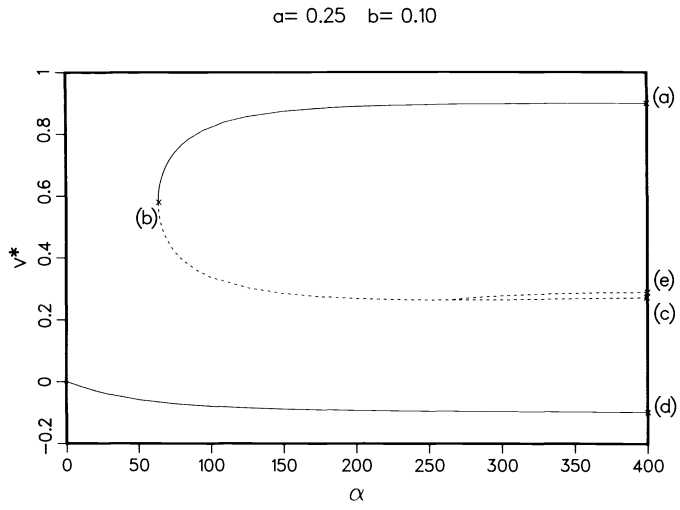


FIG. 5. *The case* $a = 0.25$, $b = 0.1$.

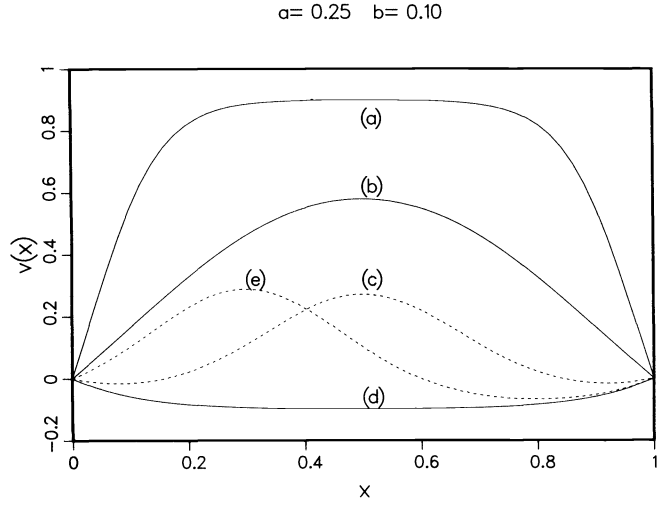


FIG. 6. *Solutions* $v(x)$ *corresponding to the points* (a)-(e) *in Fig.* 5. *Dashed lines show unstable solutions.*

branch of the second group gives a second stable solution for $\alpha \geqq 98.2476$ ($L \geqq 4.9560$). The unstable upper branch of this group has a secondary bifurcation point at $\alpha \cong 171.3209$ ($L \cong 6.5445$). Solutions for $\alpha = 400$ for each of these four curves are shown in Fig. 8.

It is interesting to note that attempts to follow this second group of solutions for decreasing a fails for values of a between 0.31 and 0.32, in good agreement with the conjecture by Manoranjan et al. connected with their inequality (3.6). A more detailed investigation of the variation of simple limit points and bifurcation points as functions of a and b is planned.

(v) $b = 1$. For $a = 0.25$, $b = 1.0$, only the constant solution $v = 0$ ($u = 1$) was found. No bifurcations in the range $0 \leqq \alpha \leqq 400$ were found.

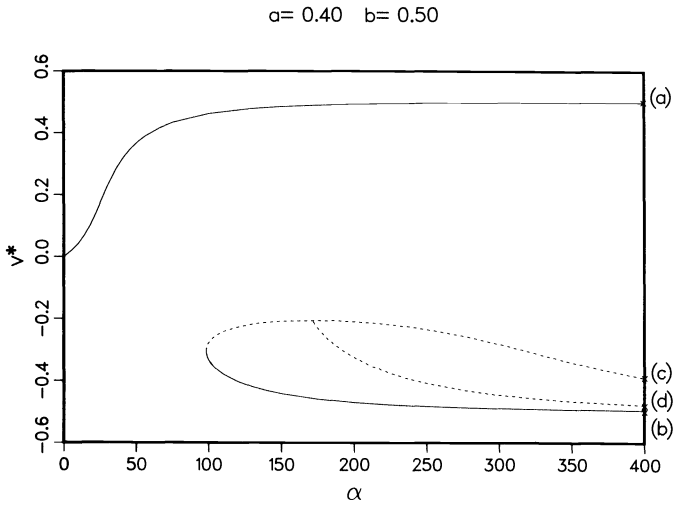


FIG. 7. *The case* $a = 0.4$, $b = 0.5$.

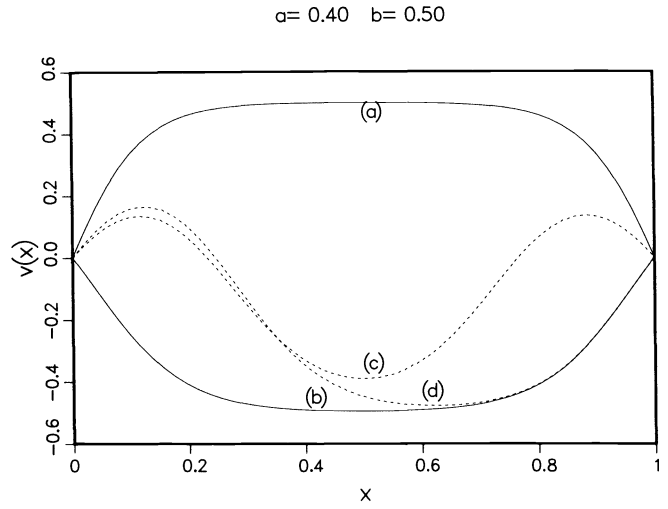


FIG. 8. *Solutions* (a)-(d) *for the points shown in Fig.* 7.

(vi) $b > 1$. Investigations of the case $a = 0.25$, $b = 1.5$, produced only a single solution curve, with the solution having a minimum at $x = 0$, as described by Manoranjan et al.

6. Conclusions. The numerically calculated bifurcation curves presented in the previous section verify and illuminate the numerical calculations of Manoranjan et al. [1984] on the number and behavior of stable steady-state solutions of the full p.d.e. (5.1). In addition a number of interesting additional unstable solutions and solution curves are generated. These unstable solutions could *not* have been found by the time integration of the full equations. Finally, the high accuracy of the pseudo-spectral method for a moderate number of basis functions and collocation points is clearly demonstrated.

It should be emphasised that the test problem we have considered here has a smooth solution, and the pseudo-spectral method with a small number of eigenfunctions

will not give accurate results when the reaction-diffusion equation exhibits internal and/or boundary layers. In some cases the use of Chebyshev basis functions rather than eigenfunctions may give higher accuracy-this question is discussed in a recent note by Eilbeck and Manoranjan (1984).

Acknowledgments. I am grateful to the Royal Society of London for partial financial support during the period whilst this work was being carried out, and to the US/UK Education Commision and the Carnegie Trust (Edinburgh) for further travel grants.

REFERENCES

W.-J. BEYN (1980), *On discretizations of bifurcation problems*, in Bifurcation Problems and Their Numerical Solution, H. D. Mittelman and H. Weber, eds., Birkhauser, Boston, pp. 46-73.

C. DE BOOR AND A. PINKUS (1978), *Proof of the conjectures of Bernstein and Erdös concerning the optimal modes for polynominal interpolation*, J. Approx. Theory, 24, pp. 289-303.

D. W. DECKER AND H. B. KELLER (1981), *Path following near bifurcation*, Comm. Pure Appl. Math., 34, pp. 149-175.

J. C. EILBECK (1983), *A collocation approach to the numerical calculation of simple gradients in reaction-diffusion systems*, J. Math. Biol., 16, pp. 233-249.

J. C. EILBECK AND V. S. MANORANJAN (1984), *A comparison of basis functions for the pseudo-spectral method for a model reaction diffusion problem*, Los Alamos Report LA-UR-84-3273, Los Alamos National laboratory, Los Alamos, NM.

D. GOTTLIEB AND S. A. ORSZAG (1977), *Numerical Analysis of Spectral Methods: Theory and Applications*, Society for Industrial and Applied Mathematics, Philadelphia.

H. B. KELLER (1979), *Global homotopies and Newton methods*, in Recent Advances in Numerical Analysis, C. de Boor and G. H. Golub, eds., Academic Press, New York, pp. 73-94.

J. P. KERNEVEZ, E. DOEDEL, M. C. DUBAN, J. F. HERVAGAULT, G. JOLY AND D. THOMAS (1983), *Spatio-temporal organization in immobilized enzyme systems*, in Rhythms in Biology and other Fields of Application, M. Cosnard, J. Demongeot and A. Le Breton, eds., Lecture Notes in Biomathematics 49, Springer-Verlag, Berlin, pp. 50-75.

M. KUBICEK (1976), *Algorithm* 502: *dependence of solution of nonlinear systems on a parameter*, ACM Trans. Math. Software, 2, pp. 98-107.

M. LENTINI AND H. B. KELLER (1980), *The Von Karman swirling flows*, SIAM J. Appl. Math., 38, pp. 52-64.

V. S. MANORANJAN, A. R. MITCHELL, B. D. SLEEMAN AND KUO PEN YU (1984), *Bifurcation studies in reaction-diffusion*, J. Comp. Appl. Math., 11, pp. 27-37.

V. S. MANORANJAN (1984), *Bifurcation studies in reaction-diffusion* II, J. Comp. Appl. Math., 11, pp. 307-314.

R. MEYER-SPASCHE AND H. B. KELLER (1980), *Computations of the axi-symmetric flow between rotating cylinders*, J. Comp. Phys., 35, pp. 100-109.

H. D. MITTELMAN AND H. WEBER (1980), *Numerical methods for bifurcation problems-a survey and classification*, in Bifurcation Problems and Their Numerical Solution, H. D. Mittelman and H. Weber, eds., Birkhauser, Boston, pp. 1-45.

R. SCHREIBER AND H. B. KELLER (1983), *Driven cavity flows by efficient numerical techniques*, J. Comp. Phys., 49, pp. 310-333.

J. SMOLLER AND A. WASSERMAN (1981), *Global bifurcation of steady-state solutions*, J. Differential Equations, 39, pp. 269-290.

SIAM J. SCI. STAT. COMPUT.
Vol. 7, No. 2, April 1986

018

THE SPECTRAL APPROXIMATION OF BICUBIC SPLINES ON THE SPHERE*

P. DIERCKX†

Abstract. Formulas are given for the spherical harmonic coefficients of bicubic splines. These formulas are useful for determining a spectral approximation to a discrete function which may be defined on a latitude-longitude grid or at arbitrarily scattered points on the surface of the sphere.

Key words. spectral approximation, spherical harmonics, bicubic splines, B-splines, smoothing on the sphere

1. Introduction. In numerical models of meteorological and geophysical processes, spectral approximations of weather data and the earth's topography are often used (see e.g. [2] and [18]).

If $f(\theta, \phi)$ is continuous over the sphere ($\theta =$ latitude, $\phi =$ longitude) and has continuous derivatives up to second order, then it can be expanded in an absolute and uniformly convergent series of surface spherical harmonics, i.e.

$$f(\theta, \phi) = \sum_{n=0}^{\infty} \sum_{m=0}^{n}{}' \bar{P}_n^m(\cos\theta)(a_{n,m}\cos(m\phi) + b_{n,m}\sin(m\phi)) \tag{1.1}$$

$$0 \leqq \theta \leqq \pi, \quad 0 \leqq \phi \leqq 2\pi,$$

where the $\bar{P}_n^m(x)$ denote the normalized associated Legendre functions [1] which can be defined as

$$\bar{P}_n^m(x) = \left(\frac{(2n+1)(n-m)!}{2(n+m)!}\right)^{1/2}(1-x^2)^{m/2}\frac{d^m}{dx^m}P_n(x) \tag{1.2}$$

with $P_n(x)$ the Legendre polynomial of degree n.

The coefficients $a_{n,m}$ and $b_{n,m}$, called the spherical harmonic coefficients of $f(\theta, \phi)$, are given by the formulas

$$a_{n,m} = \frac{1}{\pi}\int_0^{2\pi}\int_0^{\pi} f(\theta, \phi)\bar{P}_n^m(\cos\theta)\cos(m\phi)\sin\theta\, d\theta\, d\phi, \tag{1.3a}$$

$$b_{n,m} = \frac{1}{\pi}\int_0^{2\pi}\int_0^{\pi} f(\theta, \phi)\bar{P}_n^m(\cos\theta)\sin(m\phi)\sin\theta\, d\theta\, d\phi. \tag{1.3b}$$

In [19], Swarztrauber gives a survey of methods for finding an approximation for these coefficients when $f(\theta, \phi)$ is given at a discrete set of data points on a latitude-longitude grid (θ_i, ϕ_j), $i = 0, 1, \cdots, N$; $j = 0, 1, \cdots, M-1$, uniformly spaced in longitude ($\phi_j = 2\pi j/M$). An integer L and corresponding coefficients $\hat{a}_{n,m}$ and $\hat{b}_{n,m}$, $n = 0, 1, \cdots, L$; $m = 0, 1, \cdots, n$ are determined such that the truncated series

$$\sum_{n=0}^{L}\sum_{m=0}^{n}\bar{P}_n^m(\cos\theta_i)(\hat{a}_{n,m}\cos(m\phi_j) + \hat{b}_{n,m}\sin(m\phi_j)) \cong f(\theta_i, \phi_j) \tag{1.4}$$

$$\text{for } i = 0, 1, \cdots, N, \quad j = 0, 1, \cdots, M-1.$$

* Received by editors March 15, 1984, and in revised form February 11, 1985. This research was supported by FKFO under grant 2.0021.75.

† Department of Computer Science, Katholieke Universiteit Leuven, Celestijnenlaan 200A, B-3030 Heverlee, Belgium.

All investigated methods begin by Fourier transform in longitude ϕ, by computing

$$a_m(\theta_i)=\frac{2}{M}\sum_{j=0}^{M-1} f(\theta_i,\phi_j)\cos(m\phi_j), \tag{1.5a}$$

$$b_m(\theta_i)=\frac{2}{M}\sum_{j=0}^{M-1} f(\theta_i,\phi_j)\sin(m\phi_j). \tag{1.5b}$$

They differ in how the coefficients $\hat{a}_{n,m}$ and $\hat{b}_{n,m}$ are computed such that

$$\sum_{n=m}^{L} \hat{a}_{n,m}\bar{P}_n^m(\cos\theta_i)\cong a_m(\theta_i), \tag{1.6a}$$

$$\sum_{n=m}^{L} \hat{b}_{n,m}\bar{P}_n^m(\cos\theta_i)\cong b_m(\theta_i). \tag{1.6b}$$

In this paper we will expose a different approach with the advantage that it can be applied without any restriction on the position of the data points. It is based on spline smoothing. If $s(\theta,\phi)$ is a good spline approximation for $f(\theta,\phi)$, then the following integrals

$$\tilde{a}_{n,m}=\frac{1}{\pi}\int_0^{2\pi}\int_0^{\pi} s(\theta,\phi)\bar{P}_n^m(\cos\theta)\cos(m\phi)\sin\theta\,d\theta\,d\phi, \tag{1.7a}$$

$$\tilde{b}_{n,m}=\frac{1}{\pi}\int_0^{2\pi}\int_0^{\pi} s(\theta,\phi)\bar{P}_n^m(\cos\theta)\sin(m\phi)\sin\theta\,d\theta\,d\phi, \tag{1.7b}$$

will be good approximations for $a_{n,m}$ and $b_{n,m}$ as well.

In § 2 we review a number of methods for finding an appropriate (bicubic) spline approximation on the sphere. In § 3 and § 4 we recall the formulas for calculating the Fourier coefficients of B-splines and derive a simple recurrence scheme for obtaining the trigonometric expansion of $\cos^k\theta\sin^m\theta$. These results are used in § 5 where we show how the integrals (1.7a) and (1.7b) can be analytically calculated in an efficient way. Finally in § 6 some numerical examples are given.

2. Bicubic spline approximations on the sphere. In fitting data on the sphere, a rectangular approximation domain $D=[0,\pi]\times[0,2\pi]$ is concerned.

Consider the strictly increasing sequences of real numbers

$$0=\lambda_0<\lambda_1<\cdots<\lambda_g<\lambda_{g+1}=\pi, \tag{2.1}$$

$$0=\mu_0<\mu_1<\cdots<\mu_h<\mu_{h+1}=2\pi, \tag{2.2}$$

then the function $s(\theta,\phi)$ is called a bicubic spline on D, with knots λ_i, $i=1,2,\cdots,g$ in the θ-direction and μ_j, $j=1,2,\cdots,h$ in the ϕ-direction if the following conditions are satisfied:

(i) On any subrectangle $D_{i,j}=[\lambda_i,\lambda_{i+1}]\times[\mu_j,\mu_{j+1}]$, $i=0,1,\cdots,g$; $j=0,1,\cdots,h$, $s(\theta,\phi)$ is given by a polynomial of degree 3 in θ and ϕ.

(ii) All derivatives $\partial^{i+j}s(\theta,\phi)/\partial\theta^i\,\partial\phi^j$ for $0\leqq i\leqq 2$ and $0\leqq j\leqq 2$ are continuous in D.

If we introduce a number of additional knots satisfying

$$\begin{aligned}&\lambda_{-3}\leqq\lambda_{-2}\leqq\lambda_{-1}\leqq 0,\\ &\pi\leqq\lambda_{g+1}\leqq\lambda_{g+2}\leqq\lambda_{g+3},\end{aligned} \tag{2.3}$$

and

$$\mu_{-3} \leqq \mu_{-2} \leqq \mu_{-1} \leqq 0, \qquad 2\pi \leqq \mu_{h+1} \leqq \mu_{h+2} \leqq \mu_{h+3}, \tag{2.4}$$

but which are further arbitrary (for some practical considerations, see [3] and [7]), every such spline can uniquely be expressed (see for example [12]) as

$$s(\theta, \phi) = \sum_{i=-3}^{g} \sum_{j=-3}^{h} c_{i,j} M_i(\theta) N_j(\phi) \tag{2.5}$$

where $M_i(\theta)$ and $N_j(\phi)$ are normalized cubic B-splines, defined as

$$M_i(\theta) = (\lambda_{i+4} - \lambda_i) \Delta_t^4(\lambda_i, \lambda_{i+1}, \cdots, \lambda_{i+4})(t-\theta)_+^3, \tag{2.6}$$

$$N_j(\phi) = (\mu_{j+4} - \mu_j) \Delta_t^4(\mu_j, \mu_{j+1}, \cdots, \mu_{j+4})(t-\phi)_+^3, \tag{2.7}$$

with

$$x_+^k = (\max(0, x))^k \tag{2.8}$$

and where $\Delta_t^4(x_i, x_{i+1}, \cdots, x_{i+4})H(t)$ stands for the 4th divided difference of the function $H(t)$ on the points $x_i, x_{i+1}, \cdots, x_{i+4}$.

In the literature several algorithms and computer programs are described for fitting splines in the representation (2.5) to data f_q given at a discrete set of data points (θ_q, ϕ_q), $q = 1, 2, \cdots, m$. Hayes and Halliday [12] consider the least-squares problem on the assumption that the knots (2.1)-(2.2) are given. In [5] an algorithm is described for determining a smoothing spline with automatically chosen knots. Instead, the user has then to provide a single parameter by which he can control the tradeoff between closeness of fit and smoothness of fit. If the data are given at the points on a rectangular grid, interpolating splines, least-squares splines and smoothing splines can be determined in a very fast way. This is demonstrated in [4] and [6] for example.

However, all these general surface fitting algorithms may give unsatisfactory results when applied to our specific problem of approximating a function $f(\theta, \phi)$ on the sphere. Indeed, $f(\theta, \phi)$ is not only periodic in longitude but besides, the value of f is independent of ϕ for $\theta = 0$ and $\theta = \pi$. In [7] we have therefore considered the determination of least-squares and smoothing splines which satisfy the following additional constraints

$$\frac{\partial^j s(\theta, 0)}{\partial \phi^j} \equiv \frac{\partial^j (\theta, 2\pi)}{\partial \phi^j}, \qquad 0 \leqq \theta \leqq \pi, \quad j = 0, 1, 2, \tag{2.9}$$

$$s(0, \phi) \equiv s(0, 0), \qquad 0 \leqq \phi \leqq 2\pi, \tag{2.10}$$

$$s(\pi, \phi) \equiv s(\pi, 0), \qquad 0 \leqq \phi \leqq 2\pi, \tag{2.11}$$

$$\frac{\partial s(0, \phi)}{\partial \theta} \equiv \cos\phi \frac{\partial s(0, 0)}{\partial \theta} + \sin\phi \frac{\partial s(0, \pi/2)}{\partial \theta}, \qquad 0 \leqq \phi \leqq 2\pi, \tag{2.12}$$

$$\frac{\partial s(\pi, \phi)}{\partial \theta} \equiv \cos\phi \frac{\partial s(\pi, 0)}{\partial \theta} + \sin\phi \frac{\partial s(\pi, \pi/2)}{\partial \theta}, \qquad 0 \leqq \phi \leqq 2\pi. \tag{2.13}$$

Condition (2.9) simply expresses that the bicubic spline must be periodic in the variable ϕ. The conditions (2.10)-(2.13) guarantee C^1 continuity at the poles. Finally, such constraints can also be implemented in the fast algorithms for approximating data on a latitude-longitude grid (see [9] for example).

Freeden [10] and Wahba [20] have recently described another approximation method in which the notion of periodic splines on the circle and surface splines in Euclidean spaces, is generalized to the sphere. The approximating functions, which are called spherical spline functions, appear to have interesting smoothing properties. A disadvantage however is that the number of coefficients in the representation of these splines, equals (slightly exceeds) the number of data points. Therefore, if m is large, the method seems less appropriate, also due to the lack of sparsity of the system from which the coefficients must be calculated. With tensor product splines, the number of coefficients is normally quite less than the number of data points. Besides, the system to be solved is sparse due to the local support of the B-splines.

3. Integrals and Fourier coefficients of B-splines. Ready formulas for the following integrals will be needed

$$S(i,l)=\int_0^{\pi} M_i(\theta)\sin(l\theta)\,d\theta, \qquad i=-3,\cdots,g,\ l=1,2,\cdots, \tag{3.1}$$

$$C(i,l)=\int_0^{\pi} M_i(\theta)\cos(l\theta)\,d\theta, \qquad i=-3,\cdots,g,\ l=0,1,\cdots, \tag{3.2}$$

$$\tilde{S}(j,m)=\int_0^{2\pi} N_j(\phi)\sin(m\phi)\,d\phi, \qquad j=-3,\cdots,h,\ m=1,2\cdots, \tag{3.3}$$

$$\tilde{C}(j,m)=\int_0^{2\pi} N_j(\phi)\cos(m\phi)\,d\phi, \qquad j=-3,\cdots,h,\ m=0,1,\cdots. \tag{3.4}$$

In the literature we find several papers devoted to the calculation of integrals [11] and Fourier coefficients [8], [13]-[16] of B-splines. As an immediate result of the formulas given in [8] for example, we find more specifically that

$$S(i,l)=6l^{-4}(\lambda_{i+4}-\lambda_i)\,\Delta_t^4(\lambda_i,\cdots,\lambda_{i+4})G(lt_+), \qquad i=-3,-2,-1, \tag{3.5a}$$

$$=6l^{-4}(\lambda_{i+4}-\lambda_i)\,\Delta_t^4(\lambda_i,\cdots,\lambda_{i+4})\sin(lt), \qquad i=0,\cdots,g-3, \tag{3.5b}$$

$$=-\cos(\pi l)6l^{-4}(\lambda_{i+4}-\lambda_i)\,\Delta_t^4(\lambda_i,\cdots,\lambda_{i+4})G(l(\pi-t)_+), \qquad i=g-2,g-1,g, \tag{3.5c}$$

with

$$G(t)=\sin t-t+\frac{t^3}{6} \tag{3.6}$$

and

$$C(i,l)=6l^{-4}(\lambda_{i+4}-\lambda_i)\,\Delta_t^4(\lambda_i,\cdots,\lambda_{i+4})H(lt_+), \qquad i=-3,-2,-1, \tag{3.7a}$$

$$=6l^{-4}(\lambda_{i+4}-\lambda_i)\,\Delta_t^4(\lambda_i,\cdots,\lambda_{i+4})\cos(lt), \qquad i=0,\cdots,g-3, \tag{3.7b}$$

$$=\cos(\pi l)6l^{-4}(\lambda_{i+4}-\lambda_i)\,\Delta_t^4(\lambda_i,\cdots,\lambda_{i+4})H(l(\pi-t)_+), \qquad i=g-2,g-1,g, \tag{3.7c}$$

with

$$H(t)=\cos t-1+\frac{t^2}{2}. \tag{3.8}$$

The formulas for $\tilde{S}(j,m)$ and $\tilde{C}(j,m)$ are immediately obtained from (3.5) and (3.7)

if we simply replace i by j, l by m, g by h, λ by μ and π by 2π. The formula (3.7) cannot be used if $l=0$. By using (2.3), (2.6) and (2.8) and the properties of divided differences we can derive for these integrals of B-splines that

$$C(i,0)=4^{-1}(\lambda_{i+4}-\lambda_i)\,\Delta_t^4(\lambda_i,\cdots,\lambda_{i+4})t_+^4, \qquad i=-3,-2,-1, \tag{3.9a}$$

$$=4^{-1}(\lambda_{i+4}-\lambda_i), \qquad i=0,\cdots,g-3, \tag{3.9b}$$

$$=4^{-1}(\lambda_{i+4}-\lambda_i)\,\Delta_t^4(\lambda_i,\cdots,\lambda_{i+4})(\pi-t)_+^4, \qquad i=g-2,g-1,g. \tag{3.9c}$$

4. The trigonometric expansion of $\cos^k\theta\sin^m\theta$. In this section we derive a simple recursion formula for calculating the coefficients in the trigonometric expansion of $\cos^k\theta\sin^m\theta$. Using the well-known identities

$$2\cos^2\theta\equiv 1+\cos 2\theta, \tag{4.1a}$$

$$2\sin^2\theta\equiv 1-\cos 2\theta, \tag{4.1b}$$

$$2\sin(2l+1)\theta\cos 2\theta\equiv\sin(2l+3)\theta+\sin(2l-1)\theta, \tag{4.1c}$$

we can prove by induction that

$$\cos^{2k}\theta\sin^{2m+1}\theta\equiv 2^{-2k-2m}\sum_{l=0}^{k+m}\alpha_m(k,l)\sin(2l+1)\theta \tag{4.2}$$

$$k=0,1,\cdots,\quad m=0,1,\cdots,$$

if the coefficients $\alpha_m(k,l)$ are recursively defined as follows:

$$\alpha_m(k,l)=2\alpha_m(k-1,l)+\alpha_m(k-1,l-1)+\alpha_m(k-1,l+1), \tag{4.3a}$$

$$k=1,2,\cdots,l=0,1,\cdots,k+m;$$

$$\alpha_m(0,l)=2\alpha_{m-1}(0,l)-\alpha_{m-1}(0,l-1)-\alpha_{m-1}(0,l+1), \tag{4.3b}$$

$$m=1,2,\cdots,\quad l=0,1,\cdots,m;$$

$$\alpha_0(0,0)=1. \tag{4.3c}$$

Herewith it is assumed that

$$\alpha_m(k,-1)=-\alpha_m(k,0), \tag{4.4a}$$

$$\alpha_m(k,k+m+1)=\alpha_m(k,k+m+2)=0. \tag{4.4b}$$

Likewise, we can easily check that

$$\cos^{2k+1}\theta\sin^{2m+1}\theta\equiv 2^{-2k-2m-1}\sum_{l=0}^{k+m}\beta_m(k,l)\sin(2l+2)\theta, \tag{4.5}$$

$$k=0,1,\cdots,\quad m=0,1,\cdots,$$

with the coefficients $\beta_m(k,l)$ satisfying the same relations (4.3) but now on the understanding that

$$\beta_m(k,-1)=\beta_m(k,k+m+1)=\beta_m(k,k+m+2)=0. \tag{4.6}$$

The identities (4.1a) and (4.1b) together with

$$2\cos(2l+1)\theta\cos 2\theta\equiv\cos(2l+3)\theta+\cos(2l-1)\theta \tag{4.7}$$

are used to prove that the coefficients $\gamma_m(k, l)$ in the expansion

$$\cos^{2k+1}\theta \sin^{2m}\theta \equiv 2^{-2k-2m} \sum_{l=0}^{k+m} \gamma_m(k, l) \cos(2l+1)\theta, \tag{4.8}$$

$$k = 0, 1, \cdots, \quad m = 0, 1, \cdots,$$

also fulfill the relations (4.3) on the understanding that

$$\gamma_m(k, -1) = \gamma_m(k, 0), \tag{4.9a}$$

$$\gamma_m(k, k+m+1) = \gamma_m(k, k+m+2) = 0. \tag{4.9b}$$

Finally, we can also prove that

$$\cos^{2k}\theta \sin^{2m}\theta \equiv 2^{-2k-2m+1} \sum_{l=0}^{k+m} \delta_m(k, l) \cos(2l\theta), \tag{4.10}$$

$$k = 0, 1, \cdots, \quad m = 0, 1, \cdots,$$

where

$$\delta_m(k, 1) = 2\delta_m(k-1, 1) + 2\delta_m(k-1, 0) + \delta_m(k-1, 2), \tag{4.11a}$$

$$\delta_m(k, l) = 2\delta_m(k-1, l) + \delta_m(k-1, l-1) + \delta_m(k-1, l+1), \tag{4.11b}$$

$$k = 1, 2, \cdots, \quad l = 0, 2, 3, \cdots,$$

$$\delta_m(0, 1) = 2\delta_{m-1}(0, 1) - 2\delta_{m-1}(0, 0) - \delta_{m-1}(0, 2), \tag{4.11c}$$

$$\delta_m(0, l) = 2\delta_{m-1}(0, l) - \delta_{m-1}(0, l-1) - \delta_{m-1}(0, l+1), \tag{4.11d}$$

$$m = 1, 2, \cdots, \quad l = 0, 2, 3, \cdots,$$

$$\delta_0(0,0) = 1/2, \tag{4.11e}$$

on the understanding that

$$\delta_m(k, -1) = \delta_m(k, k+m+1) = \delta_m(k, k+m+2) = 0. \tag{4.12}$$

5. On calculating the spherical harmonic coefficients of bicubic splines. The formulas given in § 3 and § 4 will now be used to find an analytical expression for the coefficients $\tilde{a}_{n,m}$ and $\tilde{b}_{n,m}$ with $n = 0, 1, \cdots, N$; $m = 0, 1, \cdots, n$. Substituting (2.5) into (1.7) and by using (3.3) and (3.4) we get

$$\tilde{a}_{n,m} = \frac{1}{\pi} \sum_{i=-3}^{g} \sum_{j=-3}^{h} c_{i,j} F^i_{n,m} \tilde{C}(j, m), \tag{5.1a}$$

$$\tilde{b}_{n,m} = \frac{1}{\pi} \sum_{i=-3}^{g} \sum_{j=-3}^{h} c_{i,j} F^i_{n,m} \tilde{S}(j, m), \tag{5.1b}$$

with

$$F^i_{n,m} = \int_0^{\pi} M_i(\theta) \bar{P}^m_n(\cos\theta) \sin\theta \, d\theta. \tag{5.2}$$

From (1.2) and the following explicit expression for the Legendre polynomials (see [1] for example)

$$P_n(x) = \sum_{k=0}^{\lceil n/2 \rceil} (-1)^k \frac{(2n-2k)! x^{n-2k}}{2^n k!(n-k)!(n-2k)!}, \tag{5.3}$$

where $\lceil z \rceil$ stands for the greatest integer less than or equal to z, we find that

$$\bar{P}_{2n}^{2m}(\cos\theta) = \sum_{k=0}^{n-m} d_{n,m}(k) \sin^{2m}\theta \cos^{2k}\theta \tag{5.4}$$

with

$$d_{n,m}(k) = \left(\frac{(4n+1)(2n-2m)!}{2(2n+2m)!}\right)^{1/2} (-1)^{n-m-k} \frac{(2n+2m+2k)!}{2^{2n}(2k)!(n-m-k)!(n+m+k)!}. \tag{5.5}$$

Consequently, substituting (5.4) into (5.2) and by using (3.1) and (4.2), we get

$$F_{2n,2m}^{i} = \sum_{k=0}^{n-m} 2^{-2k-2m} d_{n,m}(k) \sum_{l=0}^{k+m} \alpha_m(k,l) S(i, 2l+1). \tag{5.6}$$

We can easily verify that the coefficients $d_{n,m}(k)$ can recursively be computed as follows

$$d_{n,m}(k) = -\frac{(n-m-k+1)(2n+2m+2k-1)}{k(2k-1)} d_{n,m}(k-1), \tag{5.7a}$$

$$k = 1, 2, \cdots, n-m,$$

$$d_{n,m}(0) = -\left(\frac{(4n+1)(2n-2m-1)(2n+2m-1)}{(4n-3)(2n-2m)(2n+2m)}\right)^{1/2} d_{n-1,m}(0), \tag{5.7b}$$

$$n = m+1, m+2, \cdots,$$

$$d_{m,m}(0) = \left(\frac{(4m+1)(4m-1)}{4m(4m-2)}\right)^{1/2} d_{m-1,m-1}(0), \qquad m = 1, 2, \cdots, \tag{5.7c}$$

$$d_{0,0}(0) = 1/\sqrt{2}. \tag{5.7d}$$

Therefore, we can calculate the requested coefficients $\tilde{a}_{2n,2m}$ and $\tilde{b}_{2n,2m}$ in an efficient way through the following scheme

(5.8a) Set $\tilde{N} = \lceil N/2 \rceil$

(5.8b) Compute $\tilde{C}(j, 2m)$ and $\tilde{S}(j, 2m)$, $j = -3, \cdots, h$; $m = 0, 1, \cdots, \tilde{N}$ from formulas similar to (3.5), (3.7) and (3.9)

(5.8c) Compute $S(i, 2l+1)$, $i = -3, \cdots, g$; $l = 0, 1, \cdots, \tilde{N}$ from (3.5)

(5.8d) Set $\alpha_0(0, 0)$ and $d_{0,0}(0)$ according to (4.3c) and (5.7d)

(5.8e) For $m = 0, 1, \cdots, \tilde{N}$ do:

(5.8e1) Compute $\alpha_m(k, l)$, $k = 0, 1, \cdots, \tilde{N} - m$; $l = 0, 1, \cdots . k+m$ from (4.3b) if $k = 0$ and $m > 0$, and from (4.3a) if $k > 0$

(5.8e2) For $n = m, m+1, \cdots, \tilde{N}$ do:

(5.8e2i) Compute $d_{n,m}(k)$, $k = 0, 1, \cdots, n-m$

from (5.7c) if $n = m$ and $m > 0$,
from (5.7b) if $n > m$ and $k = 0$,
from (5.7a) if $n > m$ and $k > 0$

(5.8e2ii) Compute $\tilde{a}_{2n,2m}$ and $\tilde{b}_{2n,2m}$ by using (5.1) and (5.6).

The remaining coefficients can be calculated through similar schemes requiring the

following formulas

$$F^i_{2n+1,2m} = \sum_{k=0}^{n-m} 2^{-2k-2m-1} e_{n,m}(k) \sum_{l=0}^{k+m} \beta_m(k,l) S(i,2l+2) \tag{5.9}$$

with

$$e_{n,m}(k) = \left(\frac{(4n+3)(2n-2m+1)!}{2(2n+2m+1)!}\right)^{1/2} (-1)^{n-m-k} \cdot \frac{(2n+2m+2k+2)!}{2^{2n+1}(2k+1)!(n-m-k)!(n+m+k)!} \tag{5.10}$$

$$F^i_{2n+1,2m+1} = \sum_{k=0}^{n-m} 2^{-2k-2m-1} f_{n,m}(k) \sum_{l=0}^{k+m+1} \delta_{m+1}(k,l) C(i,2l), \tag{5.11}$$

with

$$f_{n,m}(k) = \left(\frac{(4n+3)(2n-2m)!}{2(2n+2m+2)!}\right)^{1/2} (-1)^{n-m-k} \cdot \frac{(2n+2m+2k+2)!}{2^{2n+1}(2k)!(n-m-k)!(n+m+k+1)!} \tag{5.12}$$

and finally

$$F^i_{2n,2m-1} = \sum_{k=0}^{n-m} 2^{-2k-2m} g_{n,m}(k) \sum_{l=0}^{k+m} \gamma_m(k,l) C(i,2l+1) \tag{5.13}$$

with

$$g_{n,m}(k) = \left(\frac{(4n+1)(2n-2m+1)!}{2(2n+2m-1)!}\right)^{1/2} (-1)^{n-m-k} \cdot \frac{(2n+2m+2k)!}{2^{2n}(2k+1)!(n-m-k)!(n+m+k)!} \tag{5.14}$$

A Fortran program, called SURFCO, in which the spherical harmonic coefficients of a given bicubic spline are calculated in this manner, can be obtained from the author. As can be deduced from the scheme (5.8), this program needs an auxiliary vector for storing the coefficients $d_{n,m}(k)$ (the same vector is used for storing $e_{n,m}(k)$, $f_{n,m}(k)$ and $g_{n,m}(k)$ resp.) and auxiliary arrays for storing the coefficients $S(i,2l+1)$ (or $S(i,2l+2)$, $C(i,2l)$ and $C(i,2l+1)$ resp.), $\tilde{S}(j,2m)$ (or $\tilde{S}(j,2m+2)$), $\tilde{C}(j,2m)$ (or $\tilde{C}(j,2m+1)$) and $\alpha_m(k,l)$ (or β, δ and γ resp.).

Finally, it must be noticed that, although the Fourier coefficients S, $\tilde{S}$, C and $\tilde{C}$, as well as the coefficients α, β, γ, δ and d, e, f, g, can be calculated very accurately with the proposed schemes, there is a large loss in significant figures in evaluating the expressions (5.6), (5.9), (5.11) and (5.13) if $n-m$ is large (subtraction of large numbers to get small ones). Experimentally we have found that the resulting loss in accuracy for the coefficients $\tilde{a}_{n,m}$ and $\tilde{b}_{n,m}$ can roughly be estimated at $2+0.04(n-m)^{1.6}$ decimals. Now, in the context of finding the spherical harmonic coefficients of functions given as empirical data, these roundoff errors will normally be not larger than the approximation errors $|a_{n,m}-\tilde{a}_{n,m}|$ and $|b_{n,m}-\tilde{b}_{n,m}|$. The latter are indeed inevitable, considering that we have only a finite number of data points, which in addition are most likely subject to measurement errors. So, whichever method we use, we cannot expect then to obtain still meaningful results for large values of n.

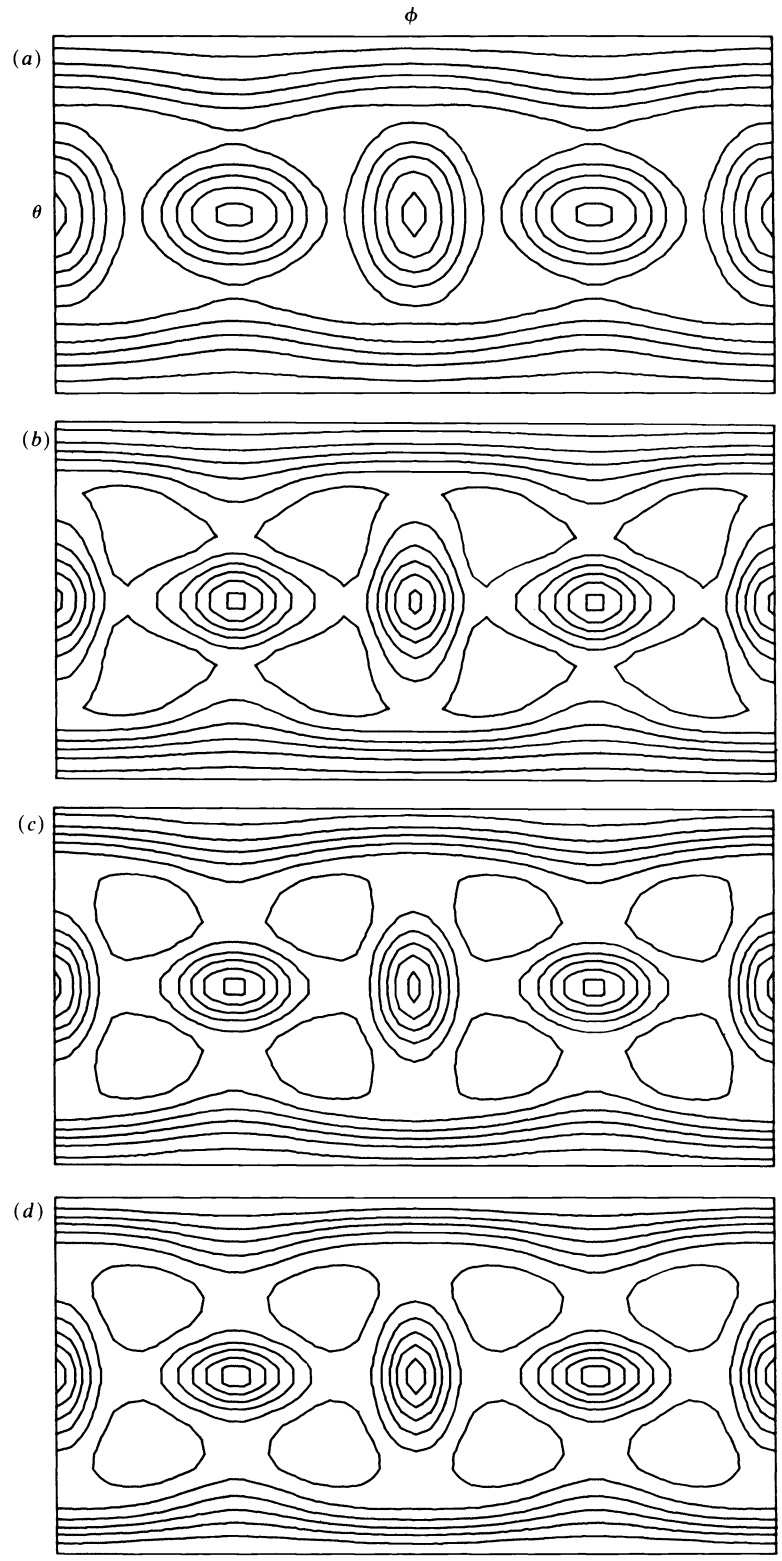


FIG. 1. *Contour plot of spherical harmonic approximations for*

$$f(\theta, \phi) = \sum_{i=1}^{3} \left[\left(\frac{\sin \theta \cos \phi}{A_i} \right)^2 + \left(\frac{\sin \theta \sin \phi}{A_{i+1}} \right)^2 + \left(\frac{\cos \theta}{A_{i+2}} \right)^2 \right]^{-1/2}$$

(*a*) $N = 6$, (*b*) $N = 8$, (*c*) $N = 10$, (*d*) $N = 12$.

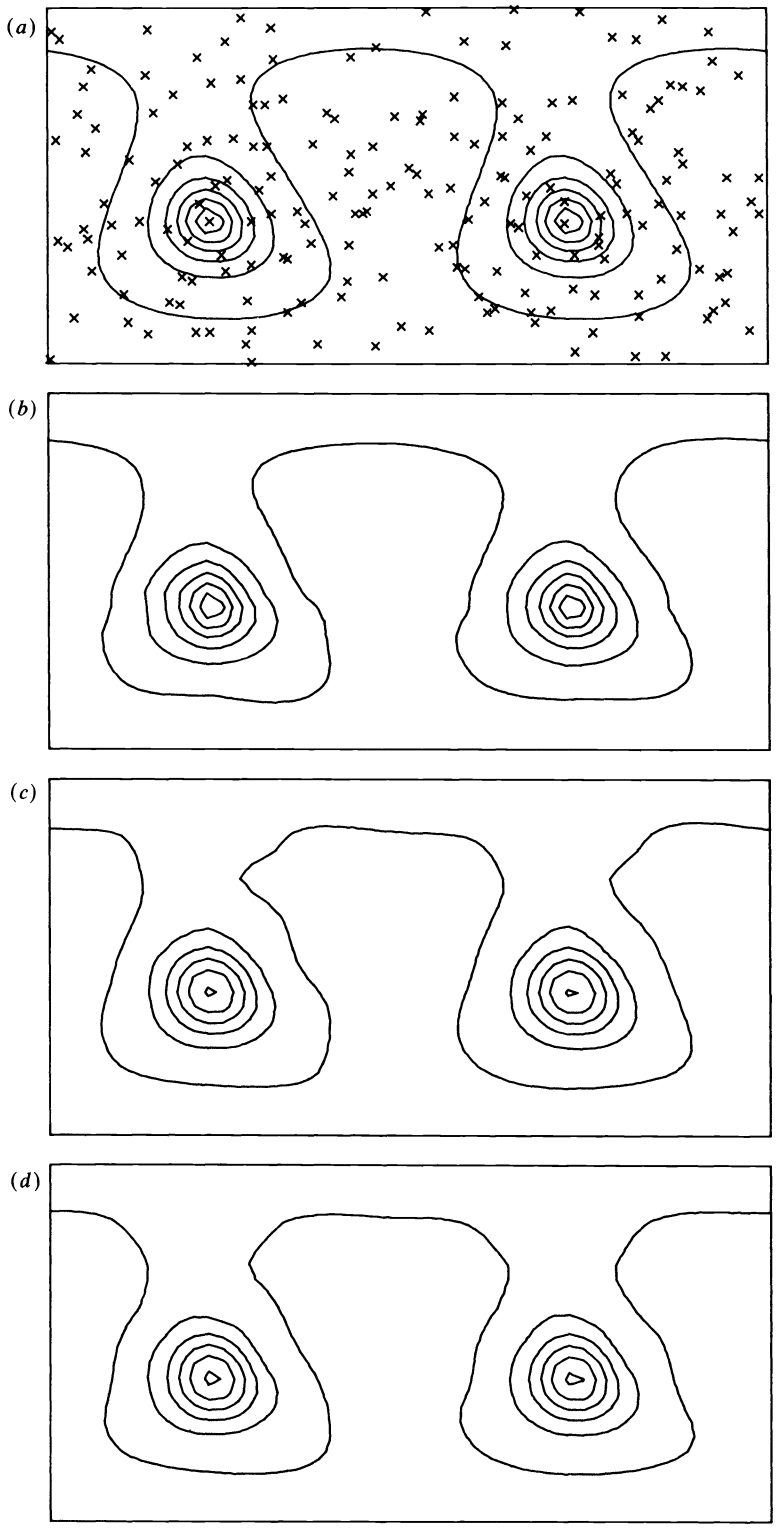


FIG. 2. *Contour plot of* (*a*) $f(\theta, \phi) = 2(4.1 + 3\cos(2\phi + \theta/4)\sin^2\theta + \cos 3\theta)^{-1}$, (*b*) *a smoothing spline approximation*; (*c*), (*d*) *spherical harmonic approximations with* $N = 10$ *and* $N = 11$.

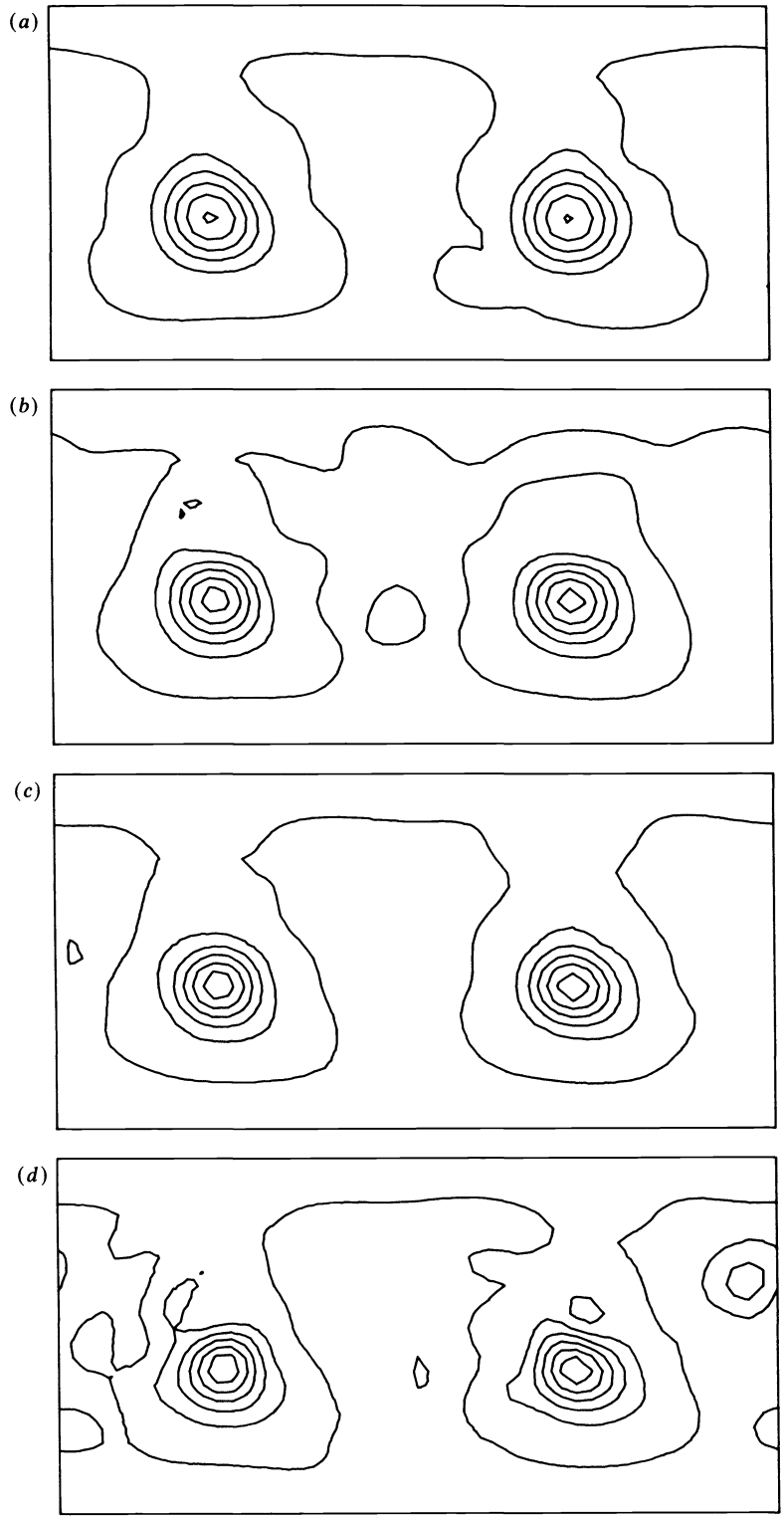


FIG. 3. *Contour plot of least-squares spherical harmonic approximations for* $f(\theta, \phi) = 2(4.1 + 3\cos(2\phi + \theta/4)\sin^2\theta + \cos 3\theta)^{-1}$. (*a*) $N = 8$; (*b*) $N = 9$; (*c*) $N = 10$; (*d*) $N = 11$.

6. Numerical examples.

Example 1. In a first example we tried to find an approximation for the spherical harmonic coefficients of

$$f(\theta,\phi)=\sum_{i=1}^{3}\left[\left(\frac{\sin\theta\cos\phi}{A_i}\right)^2+\left(\frac{\sin\theta\sin\phi}{A_{i+1}}\right)^2+\left(\frac{\cos\theta}{A_{i+2}}\right)^2\right]^{-1/2},$$
$$\{A_i|i=1,\cdots,5\}=\{5,1,2,5,1\}. \tag{6.1}$$

By using subroutine SMOCYL [9], we first determined a bicubic spline $s(\theta,\phi)$, interpolating f at the points on a uniform latitude-longitude grid ($\theta_i=\pi i/40$, $\phi_j=2\pi j/40$), $i=0,1,\cdots,40$; $j=0,1,\cdots,39$. Herewith, we imposed the constraints $\partial s(0,\phi)/\partial\theta\equiv\partial s(\pi,\phi)/\partial\theta\equiv 0$ and periodicity in ϕ. Then, by using SURFCO, we calculated the coefficients $\tilde{a}_{n,m}$ and $\tilde{b}_{n,m}$ of this spline for $n=0,1,\cdots,15$; $m=0,1,\cdots,n$. The total time for approximation and calculation of the coefficients was about 1.4 sec on a IBM 3033. Figure 1 shows a contour map of the truncated series

$$F_N(\theta,\phi)=\sum_{n=0}^{N}{\sum_{m=0}^{n}}'\bar{P}_n^m(\cos\theta)(\tilde{a}_{n,m}\cos(m\phi)+\tilde{b}_{n,m}\sin(m\phi)) \tag{6.2}$$

for different values of N (6, 8, 10, 12) illustrating the convergency. The contour map of Fig. 1d is yet hardly distinguishable from that of $f(\theta,\phi)$.

Example 2. In a second example we tried to find an approximation for the spherical harmonic coefficients of

$$f(\theta,\phi)=2(4.1+3\cos(2\phi+\theta/4)\sin^2\theta+\cos 3\theta)^{-1}. \tag{6.3}$$

Using a random number generator, we generated:

(i) a set of 192 data points (θ_q,ϕ_q) scattered uniformly over D;

(ii) a set of normally distributed stochastic variates ε_q with expected value 0 and standard deviation 0.02;

and considered the data (θ_q,ϕ_q,f_q) with $f_q=f(\theta_q,\phi_q)+\varepsilon_q$. Then, by using subroutine SMOSPH [7], we determined a smoothing spline approximation for $f(\theta,\phi)$, satisfying the constraints (2.9)-(2.13). Figure 2a shows a contour map of $f(\theta,\phi)$ corresponding to function values $0.5,1,1.5,\cdots,3$ and the position of the different data points. In Fig. 2b the contour map of the smoothing spline is given. This spline has $g=6$ knots in the θ-direction and $h=13$ in the ϕ-direction. Figure 2c and Fig. 2d show the contour maps of the corresponding series (6.2) with $N=10$ and $N=11$. Finally in Fig. 3 we have plotted for different values of $N(8,9,10,11)$ the contour map of series of the form (6.2) but now with the coefficients $\tilde{a}_{n,m}$ and $\tilde{b}_{n,m}$ computed directly by means of a least-squares criterion, i.e. such that

$$\sigma_N=\sum_{q=1}^{192}(f_q-F_N(\theta_q,\phi_q))^2 \text{ is minimal.} \tag{6.4}$$

As N gets larger, these approximations become more and more influenced by the errors ε_q, especially in those regions where there is a lack of data. On the contrary with our spline method, as N gets larger, the approximations $F_N(\theta,\phi)$ will converge to the smoothing spline.

Acknowledgment. This research is supported by the FKFO under grant 2.0021.75.

REFERENCES

[1] M. ABRAMOWITZ AND I. STEGUN, *Handbook of Mathematical Functions*, Applied Mathematics Series 55, National Bureau of Standards, Washington, DC, 1964.

[2] G. BALMINO, K. LAMBECK AND W. M. KAULA, *A spherical harmonic analysis of the earth's topography*, J. Geophys. Res., 78 (1973), pp. 478-481.

[3] M. G. COX, *The incorporation of boundary conditions in spline approximation problems*, in Numerical Analysis, G. A. Watson, ed., Lecture Notes in Mathematics 630, Springer-Verlag, Berlin, 1978, pp. 51-63.

[4] P. DIERCKX, *An algorithm for least-squares fitting of cubic spline surfaces to functions on a rectilinear mesh over a rectangle*, J. Comp. Appl. Math., 3 (1977), pp. 113-129.

[5] ———, *An algorithm for surface-fitting with spline functions*, IMA J. Num. Anal., 1 (1981), pp. 267-283.

[6] ———, *A fast algorithm for smoothing data on a rectangular grid while using spline functions*, SIAM J. Num. Anal., 19 (1982), pp. 1286-1304.

[7] ———, *Algorithms for smoothing data on the sphere with tensor product splines*, Computing, 32 (1984), pp. 319-342.

[8] P. DIERCKX AND R. PIESSENS, *Calculation of Fourier coefficients of discrete functions using cubic splines*, J. Comp. Appl. Math., 3 (1977), pp. 207-209.

[9] P. DIERCKX AND P. SUETENS, *A fast algorithm for surface reconstruction from planar contours using tensor product splines*, Rpt. TW 64, Dept. Computer Science, Katholieke Univ. Leuven, Belgium, 1983.

[10] W. FREEDEN, *On spherical spline interpolation and approximation*, Math. Meth. Appl. Sci., 3 (1981), pp. 551-575.

[11] P. W. GAFFNEY, *The calculation of indefinite integrals of B-splines*, J. Inst. Math. Appl., 17 (1976), pp. 37-41.

[12] J. G. HAYES AND J. HALLIDAY, *The least-squares fitting of cubic spline surfaces to general data sets*, J. Inst. Math. Appl., 14 (1974), pp. 89-103.

[13] M. LAX AND G. P. AGRAWAL, *Evaluation of Fourier integrals using B-splines*, Math. Comp., 39 (1982), pp. 535-548.

[14] J. T. MARTI, *An algorithm recursively computing the exact Fourier coefficients of B-splines with non-equidistant knots*, J. Appl. Math. Phys., 29 (1978), pp. 301-305.

[15] E. NEUMAN, *Moments and Fourier transforms of B-splines*, J. Comput. Appl. Math., 7 (1981), pp. 51-62.

[16] ———, *Calculation of complex Fourier coefficients using natural splines*, Computing, 29 (1982), pp. 327-336.

[17] D. L. RAGOZIN, *Uniform convergence of spherical harmonic expansions*, Math. Ann., 195 (1972), pp. 87-94.

[18] I. SIMMONDS, *Spectral representation of horizontal wind in numerical models of the atmosphere*, J. Appl. Meteor., 13 (1974), pp. 221-226.

[19] P. N. SWARZTRAUBER, *On the spectral approximation of discrete scalar and vector functions on the sphere*, SIAM J. Num. Anal., 16 (1979), pp. 934-949.

[20] G. WAHBA, *Spline interpolation and smoothing on the sphere*, this Journal, 2 (1981), pp. 5-16.

SIAM J. SCI. STAT. COMPUT.
Vol. 7, No. 2, April 1986

CONVERSION OF FFT'S TO FAST HARTLEY TRANSFORMS*

OSCAR BUNEMAN†

Abstract. The complex Fourier transform of a real function and its real Hartley transform are expressed in terms of each other, allowing translation of theorems and computer programs between the two versions. Any FFT can thus be converted, by a few indexing changes, into a Fast Hartley Transform which is equally efficient, in terms of floating point operations per real datum transformed. The FHT can therefore transform one real array of length N in half the time that it takes the FFT to process a complex array of length N. Several tricks to speed up both FHT and FFT are presented and a Fortran version of the FHT is supplied which delivers the result in $.75N \log_2 N$ multiplications and $1.75N \log_2 N$ additions.

Key words. real Fourier transforms, Hartley transform theorems, decimation, bit reversal, transform economies, tabulation of trigonometric functions

1. Introduction. Hartley's purely real version of the Fourier transform has the potential of overtaking in popularity the presently more familiar complex version (Hartley [1] uses cos + sin rather than cos + i sin as the kernel.) For the Hartley transform to become competitive, its discrete form (DHT, see Bracewell [2]) must be made as computationally economical as the well-established FFT. In this paper a Fast Hartley Transform (FHT) is presented which fulfills this requirement: it transforms one row of N real data using just half the number of operations and half the array space that it takes to transform N complex data by an FFT.

The two types of transform will be displayed side by side in a manner that highlights their similarity. Essentially, the role played by the imaginary part in the complex Fourier transform is taken on by the real Hartley transform recorded backwards. Such a representation of the two transforms allows one to convert an FFT program into an FHT program with only a few indexing changes. It also allows one to swap any "tricks" between the two that accelerate the execution. Several such tricks will be presented, and a Fortran/Basic version of an FHT is appended to this paper.

2. Fourier transforming real data. The incentive for converting to the Hartley transform is provided by the many applications in which only real data arrays have to be transformed. Such arrays have to be complemented by physically meaningless zero imaginary parts to make up the complex array required by a conventional complex Fourier transform. One knows that the resulting transformed array must be hermitian and one feels that half the computing cycles are wasted on just somehow confirming this feature. A purely real transform should get there in half the time.

This should not be taken to mean that a complex Fourier transform has no place in our computer-oriented society. There are physical variables which are naturally complex. The combination $\mathbf{E}+i\mathbf{B}$ is a classical example: omitting constants such as ε_0, μ_0 or 4π, and using the Fourier kernel $e^{-i\mathbf{k}\cdot\mathbf{r}}$ for transforming from $\mathbf{r}$-space to $\mathbf{k}$-space, Maxwell's equations condense to:

$$\left(\frac{d}{dt}-\mathbf{k}\times\right)(\mathbf{E}+i\mathbf{B})=-\mathbf{j} \quad \text{with } \mathbf{k}.(\mathbf{E}+i\mathbf{B})=-i\rho \text{ initially.}$$

Obviously, recording and updating of fields as complex data in $\mathbf{k}$-space is called for, and transforming back to $\mathbf{r}$-space by means of complex Fourier transforms [3]. But

* Received by the editors August 15, 1983, and in final form January 17, 1985.
† Electrical Engineering Department, Stanford University, Stanford, California 94305.

only in a world which has magnetic monopoles will there be no waste over transforming the source distributions **j** and ρ by a complex FFT!

Hitherto, this author has used the ugly and tedious dodge of line-pairing in multidimensional transforming, i.e. making up pseudo-complex data out of adjacent lines, transforming and then unscrambling. This is a principle used in many packaged FFT's for N real data, such as Singleton's [7] and those summarised in P. Swarztrauber's survey article [9]. The even- and odd-indexed data are paired into quasi-complex numbers, transformed by a simple complex FFT over $N/2$ points, unscrambled and subjected to a separately programmed final stage.

A more elegant, less make-shift method for getting complex transforms of real data without computational wastage is based on Bergland's [10] careful analysis of where in a complex FFT the operations occur which become redundant for real data input. He has devised an ingenious indexing scheme which allows one to by-pass these redundant operations. The bit-representation of indices becomes important in this scheme (or representation in bases other than 2 when N is not a power of 2). While interesting in itself, this method requires considerable additional programming effort: the basic simplicity of the complex FFT is lost.

In this paper an attempt is made to show that if one uses the Hartley Transform on real data, one is led to programs quite similar to, and no more complicated than, complex FFT's, and with no wastage.

3. Correspondences between the Hartley and the complex Fourier transforms. The one-dimensional Hartley- and complex Fourier transforms of some real function $h(x)$ will be defined as follows:

$$H(s) = \int [\cos 2\pi sx + \sin 2\pi sx]h(x)\,dx, \qquad F(s) = \int e^{-2i\pi sx}h(x)\,dx.$$

To establish the connection between them, the identity

$$e^{-2i\pi sx} = \frac{(1+i)\cos 2\pi sx + (1-i)\sin 2\pi sx}{1+i} \tag{1}$$

is useful: one finds

$$F(s) = \frac{H(s) + iH(-s)}{1+i} \tag{2}$$

and

$$H(s) = \operatorname{Re}(1+i)F(s) \quad \text{or} \quad H(-s) = \operatorname{Im}(1+i)F(s). \tag{3}$$

Since $F(s)$ is hermitian, these last two equations for H express the same relationship: $H = \operatorname{Re} F - \operatorname{Im} F$. After applying a packaged economical FFT to the real h-data, only one addition per element is needed to generate H, but we are aiming here at a more direct fast method for getting from h to H.

We observe from (3) that the imaginary part is associated with a sign reversal of the argument in H. Also, the odd and even parts of H are the imaginary and real parts of F, respectively, or the familiar sine and cosine transforms. Note, incidentally, that Hartley transforming preserves the parity of a function.

Amplitudes are obtained from Fourier transforms by squaring real and imaginary parts, and adding. From Hartley transforms one gets amplitudes by squaring $H(s)$ and $H(-s)$ and adding. Phases are obtained by taking arctangents of $\operatorname{Im} F/\operatorname{Re} F$ and $-H(-s)/H(s)$ respectively, but with a 45 degree offset in the latter case. A shift in

the x domain by an amount x_0 results in the Hartley transform

$$H(s)\cos 2\pi s x_0 + H(-s)\sin 2\pi s x_0$$

as compared with the Fourier transform $F(s)\exp(-2\pi i s x_0)$.

The identification of the two types of transform in terms of each other allows one to translate familiar complex Fourier transform theorems to their Hartley version. For instance one finds that the transform of the derivative of $h(x)$ must be:

$$\text{Re } 2i\pi s(1+i)F(s) = \text{Re } 2i\pi s[H(s)+iH(-s)] = -2\pi s H(-s)$$

which shows, again, that the "i" in the complex version is replaced in the Hartley version by the operation of changing the sign of the argument. The operator d^2/dx^2 transforms to $-(2\pi s)^2$, as for Fourier transforms. Care must be taken in multidimensional transforming. While the three-dimensional Laplacian transforms to $-4\pi^2(s_x^2+s_y^2+s_z^2)$, as for Fourier transforms, the three components of grad $h(x, yz)$ become

$$-2\pi s_x H(-s_x, s_y, s_z), \quad -2\pi s_y H(s_x, -s_y, s_z), \quad -2\pi s_z H(s_x, s_y, -s_z)$$

when one Hartley-transforms in x, y and z.

Other theorems can be similarly translated. In particular, one checks that convolution becomes multiplication for Hartley transforms as for complex Fourier transforms. In the common case of one of the convolutants possessing a specific parity (strictly even or strictly odd), one real multiplication per real transform element suffices. In the general, but practically rare case of neither convolutant having specific parity, one has two multiplications (and a halving of scales, usually absorbed elsewhere in one's calculation) per real element. This is comparable to the effort in convolving complex transforms where there are four real multiplications per pair of resulting real elements in the most general case, while practically often fewer multiplications suffice.

The connection formulas (2, 3) above readily yield the inversion of the Hartley transform:

$$\begin{aligned} h(x) &= \int e^{2i\pi s x}\frac{H(s)+iH(-s)}{1+i}\,ds \\ &= \int \left[\frac{1-i}{2}\cos 2\pi s x + \frac{1+i}{2}\sin 2\pi s x\right][H(s)+iH(-s)]\,ds \\ &= \int [\cos 2\pi s x + \sin 2\pi s x]H(s)\,ds, \end{aligned}$$

using the conjugate of (1) and the parities of sine and cosine. That the Hartley transform is strictly self-reciprocal is one of its plusses. For the complex Fourier transform one can achieve self-reciprocity only by taking the unconventional step of defining as "transform" the conjugate of what is normally called the Fourier transform.

In the discrete versions of either the Hartley or the complex transforms a negative index, by virtue of the periodicity, means an index running backwards through the array. In a formula such as (2), with the variable s now discretised into an index, this index need only be run through half the array length in order to cover all the information contained in the array $H(s)$. This shows how the Hartley transform has the potential of eliminating the 2:1 wastage of the complex Fourier transform. The details are to be presented in the next section.

4. The radix-2 case as an example: recursion algorithm. As an illustration, and for the purpose of producing a useful simple program, we demonstrate the translation from a complex FFT to a real Fast Hartley Transform for the radix-2 case where N, the number of real data, is a power 2. This should not be taken to mean that such a translation could not be done for radix-3, radix-4, etc. In fact, the broad principle of replacing all i's by an instruction to run the array index backwards should accomplish the translation on all cases. However, the details would become unnecessarily complicated for a demonstration and a fast Hartley program has not yet been written for any radix other than 2.

At the heart of the FFT lies the factoring or "decimation" principle which, for radix 2, takes the form of a simple recursion statement, namely that a Discrete Fourier Transform, "DFT", of length $2N$ can be decomposed into two DFT's each of length N by means of N complex multiplications and $2N$ complex additions [4]. Thus a 1024-point FFT becomes two 512-point FFT's and then four 256-point FFT's etc., down to 256 four-point FFT's. The number of operations is of the order $N \log_2 N$, and we shall be concerned with the factor in front of this "$N \log_2 N$". (A four-point FFT is trivial—it requires only additions: see § 7.) Throughout this paper, N will be assumed to be a power of 2.

For the purposes of the ensuing derivations, it is convenient to define the discrete Hartley and Fourier N-point transforms as follows:

$$H(m) = \sum h(n)\left(\cos\frac{2\pi nm}{N} + \sin\frac{2\pi nm}{N}\right), \qquad F(m) = \sum h(n)\, e^{-2i\pi nm/N}.$$

These transforms are unnormalised and hence not self-reciprocal. To achieve self-reciprocity, we shall eventually introduce a factor $N^{-1/2}$: see § 7. In the recursion formula, we denote transforms over $2N$ points by a tilde and we break up the sum into two separate sums, over the even and the odd-indexed data:

$$\begin{aligned}\tilde{F}(m) &= \sum_{n'=0}^{2N-1} h(n')\, e^{-2\pi n'm/2N} \\ &= \sum_{n=0}^{N-1} h(2n)\, e^{-2i\pi nm/N} + e^{-i\pi m/N} \sum_{n=0}^{N-1} h(2n+1)\, e^{-2i\pi nm/N}.\end{aligned}$$

We recognise the two sums to be the discrete N-point Fourier transforms over the even and the odd data respectively, to be denoted $F_e(m)$ and $F_o(m)$:

$$\tilde{F}(m) = F_e(m) + e^{-i\pi m/N} F_o(m). \tag{4}$$

The Hartley version of this formula follows from substituting the identification (2) and cancelling the denominators $(1+i)$:

$$\tilde{H}(m) + i\tilde{H}(-m) = H_e(m) + iH_e(-m) + e^{-i\pi m/N}[H_o(m) + iH_o(-m)]. \tag{5}$$

These recursion formulas prove that an $N \log_2 N$ algorithm exists for both the complex and the Hartley discrete transforms. Taking the real part of the Hartley recursion formula gives:

$$\tilde{H}(m) = H_e(m) + \cos\pi\frac{m}{N} H_o(m) + \sin\pi\frac{m}{N} H_o(-m) \tag{6}$$

and the imaginary part states the same relation, with "$-m$" in place of "m". In this form the Hartley recursion formula was first documented by R. Bracewell [5] who thereby established the existence of an $N \log_2 N$ algorithm for Hartley transforms.

The combination of a full complex $2N$-point transform from two N-point transforms requires the evaluation of (4) for $2N$ contiguous values of m, say from $1-N$ to N. For the Hartley transform, (5) need only be evaluated for the range $0<m<N$ and only for either the real part or the imaginary part when $m=0$ and $m=N$. Thus the labor in each step of the recursion is exactly half that for a full complex transform.

In (5) the recursion step for the real Hartley transform is written in terms of complex arithmetic. The real form (6) would seem more appropriate. However, the work done in executing (6) for a positive value of m and the corresponding negative value of m is exactly the same as executing (5) by complex arithmetic—and this fact allowed us to make the comparison between the full complex and the Hartley transform. Moreover, (5) shows clearly how an existing FFT program must be modified in order to serve as a Fast Hartley Transform: obviously a re-indexing job. Lastly (see § 5) there are certain economies available for the multiplication by $e^{-i\pi m/N}$ which one would miss if one evaluated (6) separately and independently for positive and negative m.

An obvious economy, exploited in any good FFT, consists of pairing the execution of (4) for any m with that for $\pm N+m$. Since F_e and F_o have the period N, we find:

$$\tilde{F}(\pm N+m)=F_e(m)-e^{i\pi m/N}F_o(m)$$

and the product $e^{-i\pi m/N}F_o(m)$ need only be evaluated once for the two cases. To generate the $2N$ complex numbers $\tilde{F}(1-N)$, $\tilde{F}(2-N)\cdots\tilde{F}(N)$, one needs to perform only N complex multiplications and $2N$ complex additions. A complex multiplication takes four real multiplications and two real additions. It follows that per real datum delivered in one recursion step there is just one real multiplication, and there are 1.5 real additions. An N-point FFT calls for $\log_2 N$ recursion steps: to generate its $2N$ real transform data (N complex pairs) takes $2N\log_2 N$ real mults and $3N\log_2 N$ real adds.

We can apply the economy of pairing m with $\pm N+m$ also to the Hartley recursion step (5). In terms of the real formula this means that one must process four cases together: m, $-m$, $-N+m$, $N-m$. The number of mults and adds per real datum returned in one recursion step is the same as for the complex FFT, namely one mult and 1.5 adds. But since an N-point Hartley transform processes only N real data, the number of mults overall is $N\log_2 N$, the number of adds $1.5N\log_2 N$: the work is just half that for a complex N-point FFT.

For the sake of completeness, we display here the four relevant real equations explicitly, thereby confirming the operations count. We also use the periodicities ($2N$ on the left, N on the right) in order to get all indices nonnegative:

$$\tilde{H}(m)=H_e(m)+\left[\cos\pi\frac{m}{N}H_o(m)+\sin\pi\frac{m}{N}H_o(N-m)\right],$$

$$\tilde{H}(N+m)=H_e(m)-\left[\cos\pi\frac{m}{N}H_o(m)+\sin\pi\frac{m}{N}H_o(N-m)\right],$$

$$\tilde{H}(N-m)=H_e(N-m)+\left[\sin\pi\frac{m}{N}H_o(m)-\cos\pi\frac{m}{N}H_o(N-m)\right],$$

$$\tilde{H}(2N-m)=H_e(N-m)-\left[\sin\pi\frac{m}{N}H_o(m)-\cos\pi\frac{m}{N}H_o(N-m)\right].$$

The index m runs only between 0 and $N/2$ now. Notice that the index runs backwards

in the last two equations. If one were to illustrate that data flow in the form of the traditional butterfly diagram (see, for instance, Brigham [11]), the butterfly would start with its wings widely spread and finally fold them together. The backward running index is also a feature used in the supplementary stage of real-data Fourier transforms which employ the pairing method, described at the end of § 2.

In order to avoid unnecessary multiplications by 1.0 and 0.0, it is worth recording and programming separately the limiting cases $m = 0$ and $m = N/2$, namely:

$$\tilde{H}(0) = H_e(0) + H_o(0), \qquad \tilde{H}\left(\frac{N}{2}\right) = H_e\left(\frac{N}{2}\right) + H_o\left(\frac{N}{2}\right);$$

$$\tilde{H}(N) = H_e(0) - H_o(0), \qquad \tilde{H}\left(3\frac{N}{2}\right) = H_e\left(\frac{N}{2}\right) - H_o\left(\frac{N}{2}\right).$$

Unlike in the case of Bergland's [10] algorithm, this exception is not necessary for the correctness of the results—it is just a minor economy. One would conclude that the indexing for Hartley transforms is not directly related to Bergland's indexing for real transforms.

It should be emphasised that our operations count only covers floating point operations and does not include fixed point adds required for indexing. It also assumes that all sines and cosines have been pretabulated (see § 8).

5. Eliminating one multiplication in every four. The rest of this paper is devoted to building a useful code from what has been presented so far. Some of what follows has no relevance to the comparison between Fourier and Hartley transforms but concerns the optimisation of either. However, further economies in both types of transform depend largely on the computer which is used for their execution—as well as on the intelligence of one's compiler, if the program is written in a high-level language.

Even the floating-point-operations count presented above may become irrelevant. In many systems one finds that data management costs exceed the arithmetic costs. On a CRAY machine, for instance, the timing is almost entirely determined by the number of fetches and stores of the array elements: the mults and adds can be overlapped with these and with each other. A vectorised radix-2 FFT written by the author for a CRAY is available through the LIBRIS facility of the NMFECC (Livermore). It transforms 64 parallel rows of N complex data in parallel, as required in multidimensional field transformations. The execution time is 64 times $4N \log_2 N$ cycles, plus a few percent overhead, because there is one fetch and one store per real datum per recursion step. By going to the trouble of programming radix-4 steps one could improve on this speed a little: see Singleton [7], for instance.

On computers and systems for which arithmetic costs dominate over data management costs, the total number of floating point operations ("flops") is more often more significant than what fraction of these are adds and what fraction mults. (Notice that we have avoided divisions altogether.) In our case, this figure is 2.5 flops per real datum per recursion step. The time for a floating-point add is often almost the same as for a floating-point mult, due to the complications in floating-point number representation.

However, there are computers and languages for which multiplications are at a premium. For these we offer the following economy: one can multiply by $e^{-i\pi m/N}$ in three real mults rather than four, at the cost of introducing another add [6].

Instead of prerecording $\sin(\pi m/N)$ and $\cos(\pi m/N)$, as needed for $2N$-point transforms, one pre-records $\sin(\pi m/N)$ and the half-angle tangent, $\tan(\pi m/2N)$.

The identity $\cos(\pi m/N) = 1 - \sin(\pi m/N)\tan(\pi m/2N)$ and the imaginary part of the rotation $X' + iY' = (X + iY)\, e^{-i\pi m/N}$ give

$$X'' \equiv \frac{Y - Y'}{\sin(\pi m/N)} = X + Y \tan\frac{\pi m}{2N}.$$

Likewise from the reverse rotation $X + iY = (X' + iY')\, e^{i\pi m/N}$,

$$X' - Y' \tan\frac{\pi m}{2N} = \frac{Y - Y'}{\sin(\pi m/N)} = X''$$

so that the rotation algorithm becomes:

$$X'' = X + Y \tan\frac{\pi m}{2N}, \quad Y' = Y - X'' \sin\pi\frac{m}{N}, \quad X' = X'' + Y' \tan\frac{\pi m}{2N},$$

and instead of four mults and two adds, we have three of each. The X's can overwrite each other, and so can the Y's. With the angle $\pi m/N$ below $\pi/2$, the half-tangent stays below 1. The operations count for either of the transforms now stands at 0.75 mults and 1.75 adds per real datum per recursion step.

This method of complex multiplication is implemented in the appended Fortran/Basic program, in the innermost loop. However, its merit must be weighed in each case, taking into account the idiosyncracies of one's computer, one's language and one's compiler. Furthermore, the preparation of the new trigonometric table may become an important consideration when the FFT is not called many times in one's program. A division is now required for each table entry (see § 8).

6. Array economy: bit-reversed indexing. The recursion formula (6) takes the N data of H_e and the N data of H_o, to combine them into the $2N$ data of $\tilde{H}$. The two arrays H_e and H_o were built up in the preceding step from four arrays each of length $N/2$. Since these earlier arrays will not be required again, one may overwrite them with $\tilde{H}$.

Evidently, then, two full arrays are sufficient for the entire recursion process: the information is tossed back and forth between them in successive steps. If the original data have to be preserved, one must copy them out before transforming.

The question arises: can one get away with a single array? Can the $\tilde{H}$ data be written straight to where they came from, i.e. into the contributing elements of the H_e and H_o arrays? Obviously, one wants all three arrays to be recorded in their proper order as given by their indices. Assuming that H_o is listed immediately following H_e, the allocation:

$$H_e(m) \leftarrow \tilde{H}(m),$$
$$H_e(N - m) \leftarrow \tilde{H}(N - m),$$
$$H_o(m) \leftarrow \tilde{H}(N + m),$$
$$H_o(N - m) \leftarrow \tilde{H}(2N - m),$$

is consistent. With this placement of the H_e and H_o arrays, the index of the combined array (the index of $\tilde{H}$) shows the parity, "e" or "o", in its leading bit. But H_e and H_o are obtained as N-point transforms from those elements in the original h-array whose index had these parities respectively, i.e. whose index showed the parity in its trailing bit.

The original elements must therefore be re-arranged, prior to the recursion process. The trailing bit of their original index must become the leading bit of their new index.

This must apply at all levels of the recursion process. For instance, within the H_e array the first $N/2$ elements must originate from those elements of the h-array whose index was divisible by 4 and the next $N/2$ elements must originate from those whose index, while even, is not divisible by 4.

It follows that as a bit pattern, the new index must be the mirror image of the old. The indices are generated from each other by "bit-reversal". Transforming begins with permutation of the elements in the form of rather less than $N/2$ swaps. (Some of the indices have symmetrical bit patterns.)

The most awkward task in the permutation phase is the construction of the bit-reversed indices. In principle, reversing a bit pattern seems a very simple task compared with, say that of squaring a number. But in practice, no computer offers even a machine instruction to perform this task. High level langauges do not provide for any form of bit manipulation at all. Language extensions sometimes allow shifts and logicals. So do assembler languages: this is helpful for the quick construction of bit-reversed indices. Bit-reversal algorithms written in high-level languages tend to be painfully slow.

The appended program incorporates a bit-reversal algorithm which makes use of the fact that the original index increments by one in the permutation loop. For an N-point transform the bit-reversed index then "increments" by $N/2$ where "incrementing" must be done with a rightward rather than leftward carry. That can be achieved by subtracting, not adding, $N/2$, $N/4$, $N/8$ etc. in each step until the result goes negative, at which point the number last subtracted is added twice.

Since high-level languages generally do not allow the sequential construction of $N/2$, $N/4$, $N/8$ etc. by shifting and would calculate this sequence by long divisions, we supply a table of powers of 2, generated once, along with the trig tables, on first call of the FHT.

The favorable outcome of this effort is that beyond these tables no other spare array is needed. The transform can be placed in the original array and this is, of course, only half as long for an N-point Hartley transform as for an N-point complex transform.

7. Normalisation and the four-point transform. The discrete Hartley and complex Fourier transforms which were used in the recursion step are not self-reciprocal. The inverse Hartley transform would be the same but for a division by N. The inverse discrete complex Fourier transform requires a change of the sign of i as well.

One of the conveniences of the continuous Hartley transform is its strict reversibility. To achieve the same for the discrete Hartley transform, we have incorporated a multiplication by $N^{-1/2}$ for our N-point transform. This is conveniently coupled with the four-point transform which one eventually reaches in the recursion process.

It would be wasteful to take the recursion any further than the four-point level because the four-point discrete transform is already free of all multiplications. After permutation to bit-inverted order (which means swapping $h(1)$ with $h(2)$ when $N=4$), the four-point transform looks like this:

$$H(0)=[h(0)+h(2)]+[h(1)+h(3)],$$
$$H(1)=[h(0)+h(2)]-[h(1)+h(3)],$$
$$H(2)=[h(0)-h(2)]+[h(1)-h(3)],$$
$$H(3)=[h(0)-h(2)]-[h(1)-h(3)].$$

In each of these four equations we introduce a multiplication by $R=N^{-1/2}$. This will

then make the FHT fully reversible. There are two additions and one multiplication per resulting element—a little more work than in a single recursion step.

8. Pretabulation of trigonometric functions. In-situ evaluation of sines and cosines is inefficient since in most applications a Hartley transform is called more than once in a program. Typically, at least one forward transform and the corresponding back-transform will be required. Even in a single transform, half of the trig functions are used more than once. The trigonometric functions of $2\pi m/N$, with m between 0 and $N/4$, should therefore be pretabulated before the first call to an N-point FHT. Only the sines are needed, cosines can be read from the table backward.

One may think that this is a trivial task: most systems have the necessary library functions available. However, it would be wasteful to call these library functions which are provided for getting sine and cosine of some arbitrary angle, usually in radians, and which are very slow. Here we need an array of sines and cosines for regularly spaced angles, integer fractions of a right angle.

For instance, by just supplying $e^{-2i\pi/N}$ and repeatedly multiplying with it, the required tables could be built in N real multiplications. This method is not advisable for large N because round-off errors are cumulative and the final sine or cosine values might deviate noticeably from the desired 1.0 and 0.0 for the angle $\pi/2$. The method is linearly unstable, but could be used for moderate N and with high-precision arithmetic.

However, a stable method is available which requires approximately $N/4$ multiplications and additions and hence is faster than library function calls. This is, again, in the nature of a recursive algorithm and involves successive halving of angles. (When one wants to avail oneself of the economy described in § 5, there are $N/4$ more additions and $N/4$ divisions as well.)

Such recursive bisection has been used by the author for his Fourier transforms since the late sixties. He also used it in the original version of his fast Poisson algorithm [12] where it is on record. Oliver [13] analysed various methods of creating trigonometric tables and found recursive bisection the best. He notes that Hopgood and Litherland [14] used it in a program for Chebyshev quadrature, but perhaps the credit for recursive bisection of angles ought to go to Archimedes.

One begins with a very coarse table of sines, conveniently the sines of 0, $\pi/8$, $\pi/4$, $3\pi/8$ and $\pi/2$, which are listed directly. Then one creates the sines of odd multiples of $\pi/16$ by means of the halving formula:

$$\sin\alpha = .5\sec\beta[\sin(\alpha-\beta)+\sin(\alpha+\beta)]$$

with $\beta = \pi/16$. From the resulting table at spacing $\pi/16$ one proceeds to a table with spacing $\pi/32$ by filling in the sines of odd multiples of this, again with the aid of the above formula and with $\beta = \pi/32$. One continues until one has reached the spacing π/N.

Apart from the initial coarse table, one also needs a record of the half-secants, $.5\sec(\pi/16)$, $.5\sec(\pi/32)$ etc. How long this record has to be depends on the precision to which one wants to work. For instance, keeping eight decimal places, only the first eleven entries differ from 0.50000000. After that, they stay at this value.

In fact, one can dispense with a list of half-secants altogether, except for the first item $.5\sec(\pi/16) = .50979558$, if one is content with this eight-place precision. The recursion formula:

$$.5\sec\frac{\beta}{2} = .7001 - \frac{.16016004}{.3004 + .5\sec\beta}$$

delivers the needed subsequent half-secants to eight places. The precision is typical of many micros, minis and other small systems, and it is better than IBM single precision. (An exact recursion formula for half-angle secants involves square roots.)

The recursive method of generating the trig tables has been incorporated in the appended program, along with this approximate algorithm for generating half-secants.

9. Comments/remarks on the Fortran/Basic program. The program supplied herewith represents an attempt to compromise between simplicity and efficiency of execution. It is intended for users of small systems, minis and micros. For powerful vector machines one would write rather different versions of a Fast Hartley Transform, or one could avail oneself of existing packaged complex transforms of real data (see references [7] and [9]), followed by the subtraction of the imaginary from the real parts—see § 3.

In our program, some extra efficiency is achieved by means of the various tricks described in the preceding sections. Simplicity is a more elusive requirement. One wants the code to be readily intelligible to both humans and computers. This can cause conflicts. For example, all older Fortran compilers refuse to recognise a zero index, and so do some Basic interpreters. In the case of Fourier transforms this is a great nuisance. In spite of a recent change for the better [15], we have adhered to the older convention: $F(1)$ in the transformed array holds the zeroth harmonic, $F(N/2+1)$ holds the alternating harmonic and $F(N)$ holds harmonic "-1".

To please the computers, we must also refrain from indulging in elegance. There are snappy and concise FFT programs in the form of just equation (4) in a triple nest of loops. These could easily be made into FHT programs. But they rely on library routines, an intelligent compiler and the availability of complex arithmetic. They are not the best for speed. And some of these elegant codes evade the permutation problem by returning transforms in bit-reversed order.

In the hope that even the dumbest compilers would understand the appended code, it was written in the most primitive form of Fortran (typically with only one variable in the index). There are no calls to library functions, and divisions, time consuming on some computers, occur only in the pretabulation, not in the FHT proper.

As a result, the code is also readily translatable into other high-level languages. It is, in fact, almost a Basic code as well as a Fortran code. The author has had specific requests for an FHT in Basic. Here are the few instructions for converting it to Basic:

Change "DO LI = I0, I1, I2" to "FOR I = I0 TO I1 STEP I2" and change the "CONTINUE" in line L to "NEXT I".

Change "IF (X.LE.Y) GO TO" to "IF X<=Y GO TO" and similarly for the other relational operators, .LT.,.EQ.,.GT.,.GE.

Change "DIMENSION" to "DIM" and label the dimension statements, deleting the "F(1)", from the dimension statement in the subroutine; replace the "CALL" to the subroutine by "GOSUB 30".

Delete the COMMON statements and the comments, beginning with "C", which were inserted merely to frame the loops. Rephrase the WRITE statements and the FORMAT descriptions in the style expected by your particular dialect of Basic.

One way in which convertibility to Basic affected the Fortran version was the introduction of line numbers for all the statements, not only those directly referred to in the code. This provides the added advantage that specific sections of the code can readily be identified in the ensuing descriptive comments—sorry, "remarks".

The program is in three parts: a calling section, lines 1-25, the pretabulation, lines 28-85, and the actual FHT, lines 100-168. The calling program generates the function

$$100(1+x)\exp(-\pi x^2).$$

This should transform into itself: it is well-known (see Bracewell [8]) that $\exp(-\pi x^2)$ transforms into itself. The derivative theorem—§ 2—then shows that x times this also transforms into itself. The scaling to discrete transforms is $x = n/N^{1/2}$, $s = m/N^{1/2}$ with $N = 64$. There is no call to an "exp" library routine. Instead, the exponential is generated progressively, starting with an initial Q of value $e^{-\pi/64} = .95209793$ and then reducing this appropriately: the multiplier R has the value $e^{-\pi/32}$.

The F-array is written out before and after the call to the FHT subroutine. The purpose of $N0$ is to put on record the last value of N for which the subroutine had been called. If the FHT is called for the same N again, the pretabulation is skipped (statement 31).

This pretabulation begins with the loop 36-39 which just creates a table of powers of 2, for use in the bit-inverter and elsewhere, in the array M. Note that $M(L+1) = 2^L$. Also, $L0 = \log_2 N$, statement 41. The M-array need only be of length $L0$. Integers $N1$, $N2$, $N3$, $N4$ are multiples of $N/16$. These are the indices for which the sines are listed in the S-array. The indices correspond to $\pi/8$, $\pi/4$, $3\pi/8$ and $\pi/2$ as angles. Since sine and cosine of angle zero are not required, there was no need to offset the index by one here. The S- and T-arrays must be of length $N/4$. The coarse table is in 53-56.

Loop 62-72 tabulates this table in progressively finer intervals: the inner loop 68-72 actually performs the multiplications by the half-secants "H". Statement 74 generates successive half-secants to eight-digit precision. The quantity "R" is here the normalising factor $N^{-1/2}$ needed for reversibility. It is generated as such directly when N is an even power of 2 by multiplication with G which alternates between 1.0 and 0.5 (see 63). When N is an odd power of 2, statement 79 will put things right. This way square rooting is avoided altogether. Loop 82-85 generates the half-angle tangents.

The actual FHT, statements 100-168, begins with the permutation loop, 103-115. The conditional return in line 111 ensures that the same pair is not swapped twice, the swap taking place in lines 112-114. K is the bit-reversed index to I. It is constructed in the inner loop 106-108 where $M(J)$ goes through the values $N/2$, $N/4$, $N/8$ etc. as described in § 6.

The loop 119-128 performs the $N/4$ four-point FHT's, with normalisation: see § 7. The large outer loop 133-166 then takes one through the recursion steps. L doubles from 4 to $N/2$, $L2$ doubles from 8 to N. $J0$ goes down by halving, from $N/8$ to 1. This is the index spacing in the trig tables for each recursion step.

The loop 137-163 goes through the separate blocks within each of which the recursion step has to be performed. There are $N/8$ such blocks in the first level, and only one in the last level.

In each block, the actual recursion formula (5) is applied. As suggested at the end of § 4, the cases of zero angle of turn (statements 138-142) and angle $\pi/2$ (statements 159-162) have been separated from the general case, with the angle between 0 and $\pi/2$: in the innermost loop runs through these angles (lines 145-157). The multiplication by $e^{-i\pi m/N}$ is performed in lines 149-151, in accordance with the equations shown in § 5. The "m" in the text is identifiable as the index J in the program. The index I of the F-array increments while K descends, thus running backwards through the array. In terms of the specific equations displayed at the end of § 4, the index identification

is:

$$m \leftrightarrow I, \quad N-m \leftrightarrow K, \quad N+m \leftrightarrow I1 = I+L, \quad 2N-m \leftrightarrow K1 = K+L.$$

In the general recursion step, "L" plays the role of the "N" in the illustration. The length of the innermost loop is one less than $L/2$.

The program has been tested in both the Fortran and the Basic version on diverse computers including several micros. It has indeed returned a transform agreeing with the original to about one part in 10^8. Timing measurements made by R. Bracewell on an HP 85 have confirmed that execution times vary as $N \log_2 N$. More precisely, it was found that they go like $N(\log_2 N - 1)$ since the recursion process does not have to be taken all the way but stops with the four-point transforms.

In due course, the program is to be made publicly available over various communication channels (such as the ARPA network). In the meantime, those interested might like to copy it into their computers by hand, with possibly their own variations on the theme. Acknowledgment of the author in any use of the subroutine would be appreciated.

10. The program.

```
      COMMON N0
      DIMENSION F(64)
   3  N0=0
   4  P=100.
   5  Q=.95209793
   6  R=Q*Q
   7  X=0.
   8  F(1)=P
C       /···············\
  10     DO 17 I=1, 32
  11     X=X+.125
  12     P=P*Q
  13     Q=Q*R
  14     F(I+1)=(1.+X)*P
  15     I1=65-I
  16     F(I1)=(1.-X)*P
  17     CONTINUE
C       \···············/
  19  WRITE(5, 20)F
  20  FORMAT(1X, 8 F9.5)
  21  N=64
  22  CALL FHT(N, F)
  23  WRITE(5, 20)F
  24  STOP
      END
C
C
      SUBROUTINE FHT(N, F)
      COMMON N0
      DIMENSION F(1), T(16), S(16), M(6)
  31  IF(N.EQ.N0) GO TO 100
  32  N0=N
  33  K=1
  34  L=0
C       /···············\
  36     M(L+1)=K
  37     K=K+K
  38     L=L+1
  39     IF(K.LT.N) GO TO 36
```

```
C       \.................../
 41   L0 = L
 42   L = L - 3
 43   N1 = M(L)
 44   N2 = N1 + N1
 45   N3 = N2 + N1
 46   N4 = N3 + N1
 47   WRITE(5.49)
 48   WRITE(5.51)N4, N4, L0
 49   FORMAT(/' FAST HARTLEY TRANSFORM—COPYRIGHT 6-21-83
      & BY OSCAR BUNEMAN, STANFORD UNIVERSITY')
 51   FORMAT(/' THE S, T AND M ARRAYS MUST HAVE
      & DIMENSIONS', I4,',',I4,',',I4/)
 53   S(N1) = .38268343
 54   S(N2) = .70710678
 55   S(N3) = .92387953
 56   S(N4) = 1.
 57   N4 = N4 - 1
 58   H = .50979558
 59   R = .25
 60   G = 1.
C       /.................................\
 62       R = R * G
 63       G = 1.5 - G
 64       L = L - 1
 65       I = M(L)
 66       A = 0.
C           /.................\
 68           DO 72 K = I, N4, N1
 69           K1 = K + I
 70           S(K) = H * (S(K1) + A)
 71           A = S(K1)
 72           CONTINUE
C           \.................../
 74       H = .7001 - .16016004/(H + .3004)
 75       N1 = I
 76       IF (N1.GT.1) GO TO 62
C       \.................................../
 78   IF(G.EQ.1.) GO TO 80
 79   R = R * .70710678
 80   K = N4
C       /.............\
 82       DO 85 I = 1.N4
 83       T(I) = S(I)/(1. + S(K))
 84       K = K - 1
 85       CONTINUE
C       \.............../
C  PRETABULATION COMPLETED
C  FAST HARTLY TRANSFORM
C  PERMUTATION:
100   K = 0
101   I = 0
C       /.........................\
103       I = I + 1
104       J = L0 + 1
C           /.................\
106           J = J - 1
107           K = K - M(J)
108           IF(K.GE.0) GO TO 106
C           \.................../
```

```
      K = K + M(J) + M(J)
      IF(I.LE.K) GO TO 103
      G = F(I + 1)
      F(I + 1) = F(K + 1)
      F(K + 1) = G
      IF(I.LT.N - 2) GO TO 103
C     \......................./
C   FOUR-POINT TRANSFORM LOOP, FOLLOWED BY RECURSION STEPS:
C     /.............\
      DO 128 I = 1, N, 4
      A = F(I    ) - F(I + 1)
      B = F(I + 2) - F(I + 3)
      G = F(I    ) + F(I + 1)
      H = F(I + 2) + F(I + 3)
      F(I    ) = R * (G + H)
      F(I + 2) = R * (G - H)
      F(I + 1) = R * (A + B)
      F(I + 3) = R * (A - B)
      CONTINUE
C     \.............../
  130 L1 = L0 - 1
      L = 4
C     /.................................\
      L2 = L + L
      L1 = L1 - 1
      J0 = M(L1)
C       /.........................\
        DO 163 I0 = 1, N, L2
        I = I0
        I1 = I + L
        A = F(I1)
        F(I1) = F(I) - A
        F(I) = F(I) + A
        K = I1 - 1
C         /.....................\
          DO 157 J = J0, N4, J0
          I = I + 1
          I1 = I + L
          K1 = K + L
          A = F(I1) + F(K1) * T(J)
          B = F(K1) - A * S(J)
          A = A + B * T(J)
          F(I1) = F(I) - A
          F(I) = F(I) + A
          F(K1) = F(K) + B
          F(K) = F(K) - B
          K = K - 1
          CONTINUE
C         \...................../
        K1 = K + L
        A = F(K1)
        F(K1) = F(K) - A
        F(K) = F(K) + A
        CONTINUE
C       \........................../
      L = L2
      IF (L.LT.N) GO TO 133
C     \................................./
  168 RETURN
      END
```

11. Acknowledgment. The author must express his gratitude to Ronald Bracewell who realised that there is such a thing as a Fast Hartley Transform and who inspired the author to pull all the stops when composing an optimal version for general use.

REFERENCES

[1] R. V. L. HARTLEY, *A more symmetrical Fourier analysis applied to transmission problems*, Proc. IRE, 30 (1942), pp. 144–150.
[2] R. N. BRACEWELL, *The discrete Hartley transform*, J. Opt. Soc. Amer., 73 (1983), pp. 1832–1835.
[3] O. BUNEMAN et al., *Principles and capabilities of* 3-d, *e-m particle simulations*, J. Comput. Phys., 38 (1980), pp. 1–44.
[4] J. W. COOLEY AND J. W. TUKEY, *An algorithm for the machine calculation of complex Fourier series*, Math. Comput., 19 (1965), pp. 297–301.
[5] R. N. BRACEWELL, *The fast Hartley transform*, submitted for publication.
[6] O. BUNEMAN, *Inversion of the Helmholtz (or Laplace–Poisson) operator for slab geometry*, J. Comput. Phys., 12 (1973), pp. 124–130.
[7] R. C. SINGLETON, *Mixed radix fast Fourier transforms*, Programs for Digital Signal Processing, pp. 14.1–14.18, IEEE Press, New York, 1979.
[8] R. N. BRACEWELL, *The Fourier Transform and Its Applications*, McGraw-Hill, New York, 1965, 2nd ed. 1978.
[9] P. N. SWARZTRAUBER, *Vectorizing the FFTs*, in Parallel Computations, Academic Press, New York, 1982, pp. 51–83.
[10] G. D. BERGLAND, *A fast Fourier transform algorithm for real-valued series*, Comm. ACM, 11 (1968), pp. 703–710.
[11] E. O. BRIGHAM, *The Fast Fourier Transform*, Prentice Hall, Englewood Cliffs, NJ, 1974.
[12] O. BUNEMAN, *A compact non-iterative Poisson solver*, SUIPR report 294, Stanford Univ., Stanford, CA, 1969.
[13] J. OLIVER, *Stable methods for evaluating* $\cos(i\pi/n)$, J. Inst. Maths Appl., 16 (1975), pp. 247–258.
[14] F. R. A. HOPGOOD AND C. LITHERLAND, Comm. ACM, 10 (1966), pp. 270.
[15] ANSI X3.9-1978 FORTRAN 77 standard.

SIAM J. SCI. STAT. COMPUT.
Vol. 7, No. 2, April 1986

020

NUMERICAL PROCEDURES FOR SURFACE FITTING OF SCATTERED DATA BY RADIAL FUNCTIONS*

NIRA DYN†, DAVID LEVIN† AND SAMUEL RIPPA†

Abstract. In many applications one encounters the problem of approximating surfaces from data given on a set of scattered points in a two-dimensional domain. The global interpolation methods with Duchon's "thin plate splines" and Hardy's multiquadrics are considered to be of high quality; however, their application is limited, due to computational difficulties, to ~150 data points. In this work we develop some efficient iterative schemes for computing global approximation surfaces interpolating a given smooth data. The suggested iterative procedures can, in principle, handle any number of data points, according to computer capacity. These procedures are extensions of a previous work by Dyn and Levin on iterative methods for computing thin-plate spline interpolants for data given on a square grid. Here the procedures are improved significantly and generalized to the case of data given in a general configuration.

The major theme of this work is the development of an iterative scheme for the construction of a smooth surface, presented by global basis functions, which approximates only the smooth components of a set of scattered noisy data. The novelty in the suggested method is in the construction of an iterative procedure for low-pass filtering based on detailed spectral properties of a preconditioned matrix. The general concepts of this approach can also be used in designing iterative computation procedures for many other problems.

The interpolation and smoothing procedures are tested, and the theoretical results are verified, by many numerical experiments.

Key words. thin-plate splines, Hardy multiquadrics, shifted logarithmics, preconditioning, roughness, iterative methods, *S*-property of a spectrum, DeVore polynomials, low-pass filtering

1. Introduction. Given a data set (x_j, y_j, f_j), $j=1, 2, \cdots, N$, the interpolation problem consists of finding a smooth surface $S(x, y)$ such that

$$S(x_j, y_j) = f_j, \qquad j=1, 2, \cdots, N. \tag{1.1}$$

Duchon [4] introduced a class of roughness measures and solved the interpolation problem so that these roughness measures are minimized. The resulting surfaces are termed "surface splines" and are of the form

$$S_m(x, y) = \sum_{i=1}^{N} \alpha_i \psi(r_i(x, y)) + P_m(x, y), \qquad m \geqq 2, \tag{1.2}$$

where $r_i^2(x, y) = (x-x_i)^2 + (y-y_i)^2$,

$$\psi(r) = r^{2(m-1)} \log r, \tag{1.3}$$

and P_m is a polynomial of degree $m-1$. The interpolating surface spline satisfies the interpolation conditions (1.1) and minimizes the roughness measure

$$J_m(u) = \iint_{R^2} \sum_{i=0}^{m} \binom{m}{i} \left(\frac{\partial^m u}{\partial x^i \, \partial y^{m-i}} \right)^2 dx\, dy, \qquad m \geqq 2. \tag{1.4}$$

Let

$$Q_m = \text{span}\{x^i y^j | i+j < m\} \equiv \text{span}\{q_1(x, y), \cdots, q_M(x, y)\}, \qquad M = \binom{m+1}{2}.$$

* Received by the editors January 24, 1984, and in revised form November 1, 1984. This research was supported by the United States-Israel Binational Science Foundation.

† School of Mathematical Sciences, Tel-Aviv University, Tel Aviv, Israel.

Then the system of equations defining the interpolating surface spline is

$$\begin{aligned} &\sum_{i=1}^{N} \alpha_i \psi(r_{ij}) + \sum_{k=1}^{M} \gamma_k q_k(x_j, y_j) = f_j, \qquad 1 \leqq j \leqq N, \\ &\sum_{i=1}^{N} \alpha_i q_k(x_i, y_i) = 0, \qquad 1 \leqq k \leqq M, \end{aligned} \tag{1.5}$$

where $r_{ij} = r_{ji} = r_i(x_j, y_j)$. This system has a unique solution if the only polynomial in Q_m vanishing at all the data points is the zero polynomial. In the following this is the underlying assumption.

A most popular member of this class of surface splines is the "thin plate spline" (TPS) S_2, which minimizes the bending energy ((1.4) with $m=2$) of an infinite thin plate clamped at the data points. Its basis functions are the radial functions $\psi(r) = r^2 \log r$ and the monomials 1, x and y.

Another class of radial basis functions, which is also very successful and popular [6], is the class of Hardy multiquadrics, namely, $\psi(r) = (r^2 + d^2)^{+1/2}$. Just recently it has been shown by Micchelli [9] that interpolation with the two types of Hardy multiquadrics is well posed, and also with the $(r^2 + d^2)^{\pm 1/2}$ basis functions augmented by a zero order polynomial, i.e. by a constant.

In this work we introduce a new class of radial basis functions $\psi(r) = \log(r^2 + d^2)$ which can be viewed as a first order surface spline with a Hardy-type shift d. The well-posedness of the interpolation problem with these basis functions augmented by a constant ((1.2) with $m=1$), for any distribution of data points, is a special case of a general result of Micchelli [9] concerning the solvability of the system (1.5) for a wide class of radial basis functions.

A major deficiency of all the above mentioned methods is due to the fact that all the systems (1.5) are full systems that become very difficult to solve, and sometimes very ill-conditioned, as N increases. Also, due to the nature of the radial functions, the common iterative methods cannot be used to solve these systems.

In [5] the authors suggested a method for preconditioning systems of equations originating from integral equations and from surface interpolation. The purpose was to increase the diagonal elements in these systems so that the transformed systems could be solved by iteration. The preconditioning in [5] is applied to systems for interpolation on a square grid, and only Richardson-type iterative procedures are considered.

In the present work we generalize the preconditioning method to systems for interpolation of general scattered data. A number of iterative schemes for solving the transformed systems are considered. Following an idea due to Agee (see [2]), the possibility of smoothing by carrying out a few iterations on the preconditioned interpolation system is also investigated. Some heuristic arguments and many numerical experiments indicate that most of the iteration matrices obtained by the preconditioning we use share the same nice spectral structure. Namely, that small eigenvalues correspond to "rough" eigenvectors and large eigenvalues to "smooth" eigenvectors. (The meaning of these terms will be clarified below.) Assuming this special structure of the iteration matrix, we develop some fast iterative methods for interpolating smooth data, and some new iterative methods for smoothing noisy data. The smoothing technique is of the type of a low-pass filter, reproducing a global surface in terms of radial basis functions, which approximates only the smooth components of the data. In this sense, it is similar to the method of smoothing with cross-validation ([11] and references

therein). Yet the advantage of our smoothing technique is in its applicability to large sets of scattered data.

The various interpolation and smoothing methods developed here are extensively investigated and verified by many computer simulations (see also [10]), and different choices of radial basis functions are examined and cross compared. All the numerical experiments are carried out for sets of data points with a nearly uniform density.

As shown in [5] there is a strong similarity between systems for interpolation by radial functions, and systems originating from integral equations with singular kernels. Therefore, all the tools for iterative solution and smoothing developed here can be adapted to the solution of integral equations of the first kind with singular kernels, with possibly a noisy right-hand side.

2. Preconditioning operators for general configurations of data points. The basic motivation to the preconditioning procedure is already described in [5]. Given a system of interpolation equations (1.5) with radial basis functions $\psi(r)$ as above, we make use of the fact that there exists a k such that

$$\Delta^k\psi(r) \downarrow 0 \quad \text{as } r \text{ increases,} \tag{2.1}$$

and

$$\Delta^k\psi(r) \to \infty \quad \text{or} \gg 1 \quad \text{as } r \to 0, \tag{2.2}$$

where Δ^k is the k-iterated Laplacian, $\Delta = \partial^2/\partial x^2 + \partial^2/\partial y^2$.

For example,

$$\Delta^{2m} r^{2(m-1)} \log r = c_m \delta(r) \tag{2.3}$$

where δ is the Dirac-δ function,

$$\Delta(r^2+d^2)^{\pm 1/2} = \pm 2(r^2+d^2)^{-1\pm 1/2}\left[1+\left(-1\pm\frac{1}{2}\right)\frac{r^2}{r^2+d^2}\right] \tag{2.4}$$

and

$$\Delta \log (r^2+d^2) = \frac{4d^2}{(r^2+d^2)^2}. \tag{2.5}$$

The parameter d is usually taken of the order of the average distance between neighbouring data points [6]. The system (1.5) for the surface splines can be written as

$$\begin{pmatrix} A & E \\ E^T & 0 \end{pmatrix}\begin{pmatrix} \boldsymbol{\alpha} \\ \boldsymbol{\gamma} \end{pmatrix} = \begin{pmatrix} \mathbf{f} \\ \mathbf{0} \end{pmatrix} \tag{2.6}$$

where $\boldsymbol{\alpha} = (\alpha_1, \cdots, \alpha_N)^T$, $\boldsymbol{\gamma} = (\gamma_1, \cdots, \gamma_M)^T$, $\mathbf{f} = (f_1, \cdots, f_N)^T$, $A_{ij} = \psi(r_{ij})$, $1 \leqq i, j \leqq N$, $E_{ik} = q_k(x_i, y_i)$, $1 \leqq k \leqq M$, $1 \leqq i \leqq N$, and rank $E = M$. With the basis functions $\psi(r) = (r^2+d^2)^{1/2}$ and $\psi(r) = \log(r^2+d^2)$ we construct interpolants $u(x, y) = \sum_{i=1}^N \alpha_i \psi(r_i(x, y)) + \gamma_1$ such that $\sum_{i=1}^N \alpha_i = 0$. The corresponding systems are of the form (2.6) with $M = 1$, $q_1(x, y) \equiv 1$. These systems have a unique solution for any configuration of data points [9]. By the preconditioning we would like to increase the diagonal elements relative to the off-diagonal elements, in the bulk of the system, namely in A. The jth column of A is a discrete form of the radial function $\psi(r_j(x, y))$, evaluated at the data points $(x_i, y_i)_{i=1}^N$. Therefore, by (2.1) and (2.2), we attempt to increase the diagonal elements of the system by operating on A with a matrix C which is a discrete version of the iterated Laplacian operator Δ^k. We require also the preconditioning matrix to annihilate the polynomial part of the system (2.6), i.e., $CE = 0$. If, moreover,

rank $C = N - M$, then the polynomial part of the solution can be restored from the solution $\boldsymbol{\alpha}$ of the transformed system.

Following [5], we derive the preconditioning matrices by using discrete analogues of the iterated Green's formulae:

$$a_m(u, v) = \iint_{\Omega} \sum_{i=0}^{m} \binom{m}{i} \left(\frac{\partial^m u}{\partial x^i \, \partial y^{m-i}}\right)\left(\frac{\partial^m v}{\partial x^i \, \partial y^{m-i}}\right) dx\, dy \tag{2.7}$$

$$= (-1)^m \iint_{\Omega} (\Delta^m u)\, v\, dx\, dy + \text{boundary terms.}$$

The null space of the functional $a_m(u, u)$ is Q_m, the space of polynomials of total degree $< m$. Firstly, a "nice" triangulation T with vertices at the data points is determined (see e.g. [8]). Then we define a discrete analogue of $a_m(u, v)$ as:

$$A_m(\mathbf{u}, \mathbf{v}) = \sum_{j=1}^{t} \sum_{i=0}^{m} \binom{m}{i} \left(\frac{\delta^m u}{\delta x^i \, \delta y^{m-i}}\right)_j \left(\frac{\delta^m v}{\delta x^i \, \delta y^{m-i}}\right)_j T_j, \tag{2.8}$$

where $(\delta^m u / \delta x^i \, \delta y^{m-i})_j$ is a finite difference approximation to $\partial^m u / \partial x^i \, \partial y^{m-i}$ on the jth triangle, which vanishes on Q_m, namely for $\mathbf{u} = E\boldsymbol{\gamma}$. The triangles indexed by $1, \cdots, t$ with area $T_1, \cdots, T_t$ are "interior triangles of order m" in the triangulation T. (The term "interior triangle of order m" is defined inductively: all triangles in T are "interior of order 1"; a triangle is "interior of order m" if it has three neighboring triangles in T (sharing one side with it) which are "interior of order $m-1$".)

Motivated by (2.7), we define the discrete analogue Δ_h^m of Δ^m by the identity

$$A_m(\mathbf{u}, \mathbf{v}) = \sum_{j=1}^{N} (\Delta_h^m \mathbf{u})_j v_j \equiv \mathbf{v}^T C \mathbf{u}. \tag{2.9}$$

In fact $(\Delta_h^m \mathbf{u})_j$ approximates $(\Delta^m u)(x_j, y_j) \cdot S_j$, where S_j is an appropriate area attached to (x_j, y_j), $j = 1, \cdots, N$. These finite difference approximations $(\Delta_h^m)_j$, $j = 1, \cdots, N$, arranged in a matrix C of order $N \times N$, comprise the preconditioning matrix.

By definition (2.8) of A_m, the matrix C in (2.9) has the properties:

$$C = C^T, \tag{2.10}$$

$$\mathbf{u}^T C \mathbf{u} \geqq 0, \qquad \mathbf{u} \in R^N, \tag{2.11}$$

$$C\mathbf{u} = 0 \quad \Leftrightarrow \quad \mathbf{u} = E\boldsymbol{\gamma}, \boldsymbol{\gamma} \in R^M. \tag{2.12}$$

The $\Rightarrow$ direction in the last property does not follow from the above requirements on the operators in (2.8). In constructing these operators one should guarantee that

$$\left(\frac{\delta^m \mathbf{u}}{\delta x^i \, \delta y^{m-i}}\right)_j = 0 \text{ for } 0 \leqq i \leqq m,\ 1 \leqq j \leqq t \quad \Rightarrow \quad \mathbf{u} = E\boldsymbol{\gamma}, \boldsymbol{\gamma} \in R^M. \tag{2.13}$$

Since $a_m(u, v)$ as given by (2.7) is the roughness measure of the function u, we view $A_m(\mathbf{u}, \mathbf{u})$ of (2.8) as the analogous discrete roughness measure of the vector of function values $\mathbf{u} = (u_j)_{j=1}^N = (u(x_j, y_j))_{j=1}^N$. Using (2.9), we obtain for the roughness $R(\mathbf{u})$ of the vector $\mathbf{u}$:

$$R(\mathbf{u}) = R_m(u) \equiv A_m(\mathbf{u}, \mathbf{u}) = \mathbf{u}^T C \mathbf{u}. \tag{2.14}$$

This measure of roughness of function values on the grid of data points is used in comparing smoothed versions of sets of noisy data. (See Theorem 3.3 and § 5.)

The construction of the matrix C for data on a square grid is described in [5]. Since for scattered data points we have investigated numerically only the cases $m = 1, 2$, we present here the procedures for the construction of the difference operators in (2.8), only for these two cases. For $m = 1$, $(\delta \mathbf{u}/\delta x, \delta \mathbf{u}/\delta y)_j$ is taken as the gradient of the linear function interpolating the data at the vertices of the jth triangle of the triangulation T, thus satisfying (2.13). The resulting matrix C is appropriate for preconditioning systems of interpolation with the Hardy multiquadrics $\{\psi_i = (r_i^2 + d^2)^{1/2}\}$ and with the shifted logarithmic functions.

For systems of interpolation with thin-plate splines, $m = 2$, and we define proper approximations to the partial derivatives u_{xx}, u_{xy}, u_{yy} as follows: for the jth triangle we take the set of vertices of all its three neighbouring triangles, which consists of 5 or 6 points; we construct a quadratic polynomial $P_j(x, y) = \sum_{\mu+\nu \leq 2} \alpha_{\mu\nu} x^\mu y^\nu$ by a least-squares fit to the values of $\mathbf{u}$ at these vertices, choosing among all possible polynomials the one minimizing $\sum_{\mu+\nu=2} \alpha_{\mu\nu}^2$; finally we take the discrete second order finite differences as the corresponding derivatives of $P_j(x, y)$. It is easy to verify that by this construction (2.13) is valid, and thus C satisfies properties (2.10)-(2.12).

Applying the preconditioning matrix C to the system (2.6), we obtain the system

$$CA\boldsymbol{\alpha} = C\mathbf{f} \tag{2.15}$$

which is a singular system. However by properties (2.10)-(2.12) it has a unique solution $\boldsymbol{\alpha}$ satisfying $E^T\boldsymbol{\alpha} = \mathbf{0}$.

Tables 1-2 present condition numbers of systems of type (2.6) and the ones of the corresponding conditioned systems (2.15), for three families of radial functions and several configurations of data points. The improvement in the condition number is significant in all the cases.

In the following sections we develop iterative methods for computing the solution $\boldsymbol{\alpha}$ of (2.15) satisfying $E^T\boldsymbol{\alpha} = \mathbf{0}$. The polynomial part of the solution of (2.6), namely the coefficients $\boldsymbol{\gamma}$, can be recovered from $\boldsymbol{\alpha}$ by solving any nonsingular part of the system $E\boldsymbol{\gamma} = \mathbf{f} - A\boldsymbol{\alpha}$, i.e. by fitting a polynomial to the residuals $\mathbf{f} - A\boldsymbol{\alpha}$. For the case of thin-plate splines on a uniform square grid, CA is diagonal dominant in most of its rows, and simple Richardson-type iterations converge quite rapidly [5]. In §§ 3, 4 more sophisticated iteration schemes are devised, based on the special spectral properties of the matrix AC, for the case of scattered data points.

3. The basic iterative schemes. Interpolation of a set of data (x_i, y_i, f_i), $i = 1, \cdots, N$, by radial functions of the above mentioned types, requires the solution of systems of the form (1.5), or, in the notation introduced in § 2:

$$\begin{aligned} &A\boldsymbol{\alpha} + E\boldsymbol{\gamma} = \mathbf{f} \quad (A^T = A), \\ &E^T\boldsymbol{\alpha} = \mathbf{0}. \end{aligned} \tag{3.1}$$

As shown in [4] for the surface splines, and by Micchelli [9] also for Hardy multiquadrics and the shifted logarithmic functions, all these systems share the property that

$$\varepsilon \mathbf{x}^T A \mathbf{x} > 0 \quad \text{if } E^T\mathbf{x} = 0, \quad x \neq 0, \tag{3.2}$$

with $\varepsilon = +1$ or -1. Namely, A is conditionally positive or negative definite with respect to E. In the following we assume that A is conditionally positive definite, replacing A by $-A$ if necessary.

The iterative schemes developed in this work are an adaptation and extension of the classical polynomial type schemes, such as the conjugate gradient and the Chebyshev schemes [7], to systems of the form (3.1) with the property (3.2).

Applying the preconditioning matrix C of § 2, with the properties

$$C^T = C, \quad \mathbf{x}^T C\mathbf{x} \geqq 0 \quad \text{for all } \mathbf{x} \in R^N, \tag{3.3}$$

$$C\mathbf{x} = 0 \quad \Leftrightarrow \quad \mathbf{x} = E\boldsymbol{\gamma}, \boldsymbol{\gamma} \in R^M, \tag{3.4}$$

to the system (3.1), we obtain the singular system

$$CA\boldsymbol{\alpha} = C\mathbf{f}, \tag{3.5}$$

which by (3.2)-(3.4) has a unique solution $\boldsymbol{\alpha}$ satisfying $E^T\boldsymbol{\alpha} = 0$.

In the following we investigate the convergence properties of general polynomial type iteration schemes [7] for the solution of the singular system (3.5). In these schemes the iterants $\boldsymbol{\alpha}^{(k)}$ satisfy the condition $E^T\boldsymbol{\alpha}^{(k)} = 0$, and the errors $\mathbf{e}^{(k)} = \boldsymbol{\alpha}^{(k)} - \boldsymbol{\alpha}$ are of polynomial form:

$$\mathbf{e}^{(k)} = P_k(CA)\mathbf{e}^{(0)}, \quad \mathbf{e}^{(0)} = -\boldsymbol{\alpha}, \quad P_k(0) = 1. \tag{3.6}$$

It follows from (3.1) and the relation $AP_k(CA) = P_k(AC)A$ that the residuals $\mathbf{r}^{(k)} = \mathbf{f} - A\boldsymbol{\alpha}^{(k)}$ are of the form:

$$\mathbf{r}^{(k)} = P_k(AC)\mathbf{r}^{(0)}, \qquad \mathbf{r}^{(0)} = \mathbf{f}. \tag{3.7}$$

The study of specific polynomials $P_k(w)$ is postponed to §§ 4, 5, where in addition to the classical polynomials $P_k(w)$ we present new polynomials, devised on the basis of the spectral properties of the matrix AC. The schemes obtained with the nonclassical polynomials yield a significant improvement in the rate of convergence of the iterations, and are producing good smoothing surfaces in case of noisy data.

The polynomials considered in §§ 4, 5 correspond to either a one-step or a two-step recurrence relation for $\boldsymbol{\alpha}^{(k)}$. The one-step schemes are of the form:

$$\boldsymbol{\alpha}^{(k+1)} = \boldsymbol{\alpha}^{(k)} + w_k C[\mathbf{f} - A\boldsymbol{\alpha}^{(k)}], \qquad k = 0, 1, 2, \cdots, \tag{3.8}$$

i.e. $\boldsymbol{\alpha}^{(k-1)} = \boldsymbol{\alpha}^{(k)} - w_k CA\mathbf{e}^{(k)}$ or $\mathbf{e}^{(k+1)} = (I - w_k CA)\mathbf{e}^{(k)} = \cdots = P_k(CA)\mathbf{e}^{(0)}$ with

$$P_k(w) = \prod_{i=0}^{k-1} (1 - ww_i). \tag{3.9}$$

The two-step schemes are:

$$\boldsymbol{\alpha}^{(k+1)} = \rho_{k+1}[\boldsymbol{\alpha}^{(k)} + w_k C(\mathbf{f} - A\boldsymbol{\alpha}^{(k)})] + (1 - \rho_{k+1})\boldsymbol{\alpha}^{(k-1)}, \qquad k = 1, 2, \cdots \tag{3.10}$$

with $P_k(w)$ defined by the recurrence relation [7]:

$$P_{k+1}(w) = \rho_{k+1}(1 - w_k w)P_k(w) + (1 - \rho_{k+1})P_{k-1}(w), \qquad k = 1, 2, \cdots. \tag{3.11}$$

We define a scheme to be convergent if there exists a subsequence $k_1 < k_2 < \cdots$ such that

$$\lim_{j \to \infty} \mathbf{e}^{(k_j)} = 0. \tag{3.12}$$

Then by (3.5) $C\mathbf{r}^{(k_j)} \to 0$, which by (3.4) guarantees the existence of $\boldsymbol{\gamma} \in R^M$ such that

$$\lim_{j \to \infty} \mathbf{r}^{(k_j)} = E\boldsymbol{\gamma},$$

thus providing a solution to the system (3.1).

The choice of the parameters $\{w_i\}$ in (3.8) or $\{\rho_i, w_i\}$ in (3.10) determines the convergence properties of the scheme, and depends strongly on the spectral properties of the matrices CA and AC, which by the symmetry of A and C satisfy $(CA)^T = AC$.

In the following we denote by $\mathbb{E}$ the subspace spanned by the columns of E: $E_{\cdot 1}, \cdots, E_{\cdot M}$.

THEOREM 3.1. *The eigenvalues of the matrices CA and AC are real and satisfy*

$$0 = \lambda_1 = \cdots = \lambda_M < \lambda_{M+1} < \cdots < \lambda_N. \tag{3.13}$$

The eigenvectors of AC $\mathbf{u}^{(1)}, \cdots, \mathbf{u}^{(N)}$ span R^N *and are orthogonal with respect to the matrix C*:

$$\mathbf{u}^{(j)T} C \mathbf{u}^{(i)} = \delta_{ij} (\mathbf{u}^{(i)T} C \mathbf{u}^{(i)}), \qquad i, j = 1, \cdots, N, \tag{3.14}$$

with $\mathbf{u}^{(j)} = E_{\cdot j}$, $j = 1, \cdots, M$. *The eigenvectors of CA corresponding to* $\lambda_{M+1}, \cdots, \lambda_N$ span $\mathbb{E}^{\perp}$ *and are of the form*

$$\mathbf{v}^{(j)} = C \mathbf{u}^{(j)}, \qquad j = M+1, \cdots, N. \tag{3.15}$$

Proof. Since $\mathbf{x}^T C \mathbf{x} \geqq 0$, $\mathbf{x} \in R^N$ with strict inequality if $\mathbf{x} \in \mathbb{E}^{\perp}$, the matrix $C^{1/2}$ exists, has the characterization (3.2)-(3.4) with C replaced by $C^{1/2}$, and has an inverse over $\mathbb{E}^{\perp}$. Therefore, the eigenvalues of $C^{1/2} A C^{1/2}$ are of the form $0 = \lambda_1 = \cdots = \lambda_M < \lambda_{M+1} \leqq \cdots \leqq \lambda_N$, and the eigenvectors are $\mathbf{w}^{(j)} = E_{\cdot j}$, $j = 1, \cdots, M$, $\mathbf{w}^{(j)} \in \mathbb{E}^{\perp}$, $j = M+1, \cdots, N$. Also the vectors $\hat{\mathbf{u}}^{(j)}$ satisfying

$$C^{1/2} \hat{\mathbf{u}}^{(j)} = \mathbf{w}^{(j)}, \quad \hat{\mathbf{u}}^{(j)} \in \mathbb{E}^{\perp}, \quad j = M+1, \cdots, N,$$

are well defined and linearly independent.

Now by the definition of $\hat{\mathbf{u}}^{(j)}$

$$AC \hat{\mathbf{u}}^{(j)} = AC^{1/2} \mathbf{w}^{(j)} = \lambda_j [\hat{\mathbf{u}}^{(j)} + \tilde{\mathbf{u}}^{(j)}], \qquad \tilde{\mathbf{u}}^{(j)} \in \mathbb{E},$$

where the last equality holds since $C^{1/2}[AC^{1/2} w^{(j)}] = \lambda_j \mathbf{w}^{(j)}$. Obviously $\tilde{\mathbf{u}}^{(j)} = ((1/\lambda_j) AC - I) \hat{\mathbf{u}}^{(j)}$, $j = M+1, \cdots, N$. Hence

$$AC(\hat{\mathbf{u}}^{(j)} + \tilde{\mathbf{u}}^{(j)}] = AC \hat{\mathbf{u}}^{(j)} = \lambda_j [\hat{\mathbf{u}}^{(j)} + \tilde{\mathbf{u}}^{(j)}]$$

and the vector $\mathbf{u}^{(j)} = \hat{\mathbf{u}}^{(j)} + \tilde{\mathbf{u}}^{(j)} = (1/\lambda_j) AC \hat{\mathbf{u}}^{(j)}$ is an eigenvector of AC corresponding to the eigenvalue λ_j. Moreover $\mathbf{u}^{(j)}$, $j = M+1, \cdots, N$ are linearly independent since so are $\hat{\mathbf{u}}^{(j)} \in \mathbb{E}^{\perp}$, $j = M+1, \cdots, N$, while $\tilde{\mathbf{u}}^{(j)} \in \mathbb{E}$, $j = M+1, \cdots, N$.

The vectors $\mathbf{u}^{(j)} = E_{\cdot j}$, $j = 1, \cdots, M$, are eigenvectors of AC corresponding to the zero eigenvalue, and span $\mathbb{E}$. Hence the vectors $\mathbf{u}^{(j)}$, $j = 1, \cdots, N$ span R^N.

The vectors $\mathbf{w}^{(j)}$ are orthogonal since $C^{1/2} A C^{1/2}$ is a symmetric matrix. The orthogonality in terms of the vectors $\mathbf{u}^{(j)}$ reads

$$\mathbf{w}^{(i)T} \mathbf{w}^{(j)} = \hat{\mathbf{u}}^{(i)T} C \hat{\mathbf{u}}^{(j)} = \mathbf{u}^{(i)T} C \mathbf{u}^{(j)} = \delta_{ij} [\mathbf{u}^{(i)T} C \mathbf{u}^{(i)}], \qquad i, j = M+1, \cdots, N.$$

This together with $CE = 0$ implies (3.14).

Now the vectors $\mathbf{v}^{(j)} = C \mathbf{u}^{(j)} \in \mathbb{E}^{\perp}$, $j = M+1, \cdots, N$ satisfy

$$CA \mathbf{v}^{(j)} = C[AC \mathbf{u}^{(j)}] = C \lambda_j \mathbf{u}^{(j)} = \lambda_j \mathbf{v}^{(j)}$$

and are linearly independent since so are $\hat{\mathbf{u}}^{(j)} \in \mathbb{E}^{\perp}$, $j = M+1, \cdots, N$, and C is invertible over $\mathbb{E}^{\perp}$. □

THEOREM 3.2. *A scheme satisfying* (3.6) *and* (3.7) *is convergent if there exists a subsequence* $k_1 < k_2 < k_3 < \cdots$ *such that*

$$\lim_{j \to \infty} \max_{M+1 \leqq i \leqq N} |P_{k_j}(\lambda_i)| = 0 \tag{3.16}$$

with $\lambda_{M+1}, \cdots, \lambda_N$ *as in Theorem* 3.1.

Proof. Let $r^{(0)} = \sum_{i=1}^{N} \theta_i \mathbf{u}^{(i)}$ where $\mathbf{u}^{(i)}$, $i = 1, \cdots, N$ are as in Theorem 3.1. Then by (3.6), (3.7) and (3.13)

$$\mathbf{r}^{(k_j)} = \sum_{i=1}^{N} \theta_i P_{k_j}(AC)\mathbf{u}^{(i)} = \sum_{i=1}^{M} \theta_i \mathbf{u}^{(i)} + \sum_{i=M+1}^{N} \theta_i P_{k_j}(\lambda_i)\mathbf{u}^{(i)},$$

which in view of (3.16) yields

$$\lim_{j\to\infty} \mathbf{r}^{(k_j)} = \sum_{i=1}^{M} \theta_i \mathbf{u}^{(i)}.$$

Since $\mathbf{u}^{(i)} = E_{\cdot i}$, $i = 1, \cdots, M$, $\lim_{j\to\infty} \mathbf{r}^{(k_j)} = E\boldsymbol{\gamma}$. On the other hand by (3.6) and (3.15), $\mathbf{e}^{(0)} = -\boldsymbol{\alpha} = \sum_{i=M+1}^{N} \xi_i (C\mathbf{u}^{(i)})$,

$$\boldsymbol{\alpha}^{(k_j)} - \boldsymbol{\alpha} = \mathbf{e}^{(k_j)} = \sum_{i=M+1}^{N} \xi_i P_{k_j}(\lambda_i) C\mathbf{u}^{(i)}, \qquad j = 1, 2, \cdots,$$

and in view of (3.16) $\lim_{j\to\infty} \mathbf{e}^{(k_j)} = \mathbf{0}$.

A cyclic choice of the sequence $\{w_i\}_{i=1}^{\infty}$ in (3.8)

$$w_{ks+j} = w_j, \quad j = 1, \cdots, s, \quad k = 1, 2, \cdots, \tag{3.17}$$

corresponds to the sequence $k_j = js$ in Theorem 3.2.

A sufficient condition for the convergence of the scheme (3.8) in this case is

$$\max_{M+1 \leqq j \leqq N} |P_s(\lambda_j)| < 1 \tag{3.18}$$

with an average rate of convergence [7]

$$\tau = [\max_{M+1 \leqq j \leqq N} |P_s(\lambda_j)|]^{1/s}. \tag{3.19}$$

An important feature of the scheme (3.6), (3.7) is:

THEOREM 3.3. *Let $k_1 \leqq k_2 \leqq \cdots$, be a subsequence such that*

$$0 \leqq P_{k_{j+1}}(\lambda_i) \leqq P_{k_j}(\lambda_i) \leqq 1, \qquad M+1 \leqq i \leqq N, \quad j = 1, 2, \cdots, \tag{3.20}$$

with at least one index i_0, $M+1 \leqq i_0 \leqq N$ for which the inequalities (3.20) *are strict. Then the approximants produced by the scheme* (3.6), (3.7)

$$g_{k_j}(x, y) = \sum_{i=1}^{N} \alpha_i^{(k_j)} \psi_i(x, y) + \sum_{i=1}^{M} \gamma_i^{(k_j)} q_i(x, y) \tag{3.21}$$

with $\boldsymbol{\gamma}^{(k_j)}$ arbitrary, have discrete roughness (as defined in (2.14)):

$$R(g_{k_j}) = \sum_{\nu,\mu=1}^{N} g_{k_j}(x_\nu, y_\nu) C_{\nu\mu} g_{k_j}(x_\mu, y_\mu) = (A\boldsymbol{\alpha}^{(k_j)})^T C (A\boldsymbol{\alpha}^{(k_j)}), \tag{3.22}$$

which are monotonically increasing

$$R(g_{k_j}) < R(g_{k_{j+1}}), \qquad j = 1, 2, 3, \cdots, \tag{3.23}$$

and bounded from above by the roughness of the original data:

$$R(g_{k_j}) \leqq R(\mathbf{f}) = \mathbf{f}^T C \mathbf{f}. \tag{3.24}$$

Proof. The vector of function values of g_{k_j} at the data points is

$$\mathbf{g}^{(k_j)} \equiv A\boldsymbol{\alpha}^{(k_j)} + E\boldsymbol{\gamma}^{(k_j)} = \mathbf{f} - \mathbf{r}^{(k_j)} + E\boldsymbol{\gamma}^{(k_j)}.$$

Then as in the Proof of Theorem 3.2 $\mathbf{f}=\mathbf{r}^{(0)}=\sum_{i=1}^{N}\theta_i\mathbf{u}^{(i)}$ and by (3.7)

$$\mathbf{f}-\mathbf{r}^{(k_j)}=[I-P_{k_j}(AC)]\mathbf{f}=\sum_{i=1}^{N}[1-P_{k_j}(\lambda_i)]\theta_i\mathbf{u}^{(i)}.$$

Therefore in view of (3.14)

$$R(g_{k_j})=\mathbf{g}^{(k_j)T}C\mathbf{g}^{(k_j)}=[\mathbf{f}-\mathbf{r}^{(k_j)}]^TC[\mathbf{f}-\mathbf{r}^{(k_j)}]$$

$$=\sum_{i=M+1}^{N}[1-P_{k_j}(\lambda_i)]^2\theta_i^2R(\mathbf{u}^{(i)}).$$

while

$$R(\mathbf{f})=\sum_{i=M+1}^{N}\theta_i^2R(\mathbf{u}^{(i)}).$$

Hence by (3.20)

$$R(\mathbf{f})-R(g_{k_j})=\sum_{i=M+1}^{N}\theta_i^2\{1-[1-P_{k_j}(\lambda_i)]^2\}R(\mathbf{u}^{(i)})>0,$$

$$R(g_{k_{j+1}})-R(g_{k_j})=\sum_{i=M+1}^{N}\theta_i^2\{[1-P_{k_{j+1}}(\lambda_i)]^2-[1-P_{k_j}(\lambda_i)]^2\}R(\mathbf{u}^{(i)})>0.$$

Remark. In the case of a cyclic choice of the sequence $\{w_i\}$ as in (3.17), condition (3.20) is satisfied if

$$0\leqq P_s(\lambda_i)\leqq 1,\qquad i=M+1,\cdots,N, \tag{3.25}$$

with strict inequalities in (3.25) for at least one i_0, $M+1\leqq i_0\leqq N$. Condition (3.25) is satisfied by the subsequence $k_j=2js$, for any cyclic choice of the sequence $\{w_i\}$ such that condition (3.18) holds, since

$$P_{2js}(w)=[P_s(w)]^{2j}\geqq 0. \tag{3.26}$$

Theorem 3.3 reveals the smoothing nature of the process (3.6), (3.7), which provides approximants with smoother values on the set of data points than the original data. This property will be further investigated in § 5 where smoothing methods for noisy data are developed.

4. Methods for interpolation. The choice of the polynomials $P_k(w)$ in the scheme (3.6), (3.7), which guarantees good convergence rates in the solution of the system (3.5), is the subject of this section.

It follows from the analysis in § 3 that the functional behaviour of the polynomials $\{P_k(w)\}$ is significant only in the interval $[\lambda_{M+1},\lambda_N]$. In [5] the application of the classical Richardson scheme [7] $P_k(w)=(1-w_0w)^k$ is investigated. Here we test and compare the more efficient classical schemes:

(a) *The optimal Chebyshev scheme.* In this scheme, $P_k(w)$ for $k\geqq 1$ is taken to be $T_k(w)$—the Chebyshev polynomial of degree k normalized to the interval $[\lambda_{M+1},\lambda_N]$. The parameters in (3.10) for this scheme are [7]:

$$\rho_1=1,\quad \rho_2=(1-\tfrac{1}{2}\sigma^2)^{-1},\quad \rho_{k+1}=(1-\tfrac{1}{4}\sigma^2\rho_k),\quad w_k=2(\lambda_{M+1}+\lambda_N)^{-1} \tag{4.1}$$

where $\sigma=(\lambda_N-\lambda_{M+1})/(\lambda_N+\lambda_{M+1})$, and the average rate of convergence at the kth iteration is bounded by [7]:

$$\tau\leqq[T_k(1/\sigma)]^{-1/k}=\frac{2\theta^{1/2}}{(1+\theta^k)^{1/k}},\qquad \theta=\frac{1-\sqrt{1-\sigma^2}}{1+\sqrt{1-\sigma^2}}. \tag{4.2}$$

The numerical results obtained with this scheme (Tables 4, 5) also shed light on the efficiency of the cyclic Chebyshev scheme [7], defined by $P_{ks}(w) = [T_s(w)]^k$, $k \geqq 1$, where $s > 0$ is a fixed integer.

The application of these schemes requires good estimates of the spectral boundaries λ_{M+1}, λ_N. Methods for computing these estimates are briefly discussed at the end of this section.

(b) *Conjugate gradient scheme.* This scheme applies for systems with a nonnegative (nonpositive) definite matrix [7]. Therefore the preconditioning of the system (3.1) leading to (3.5) is not appropriate. The correct preconditioning is

$$C^{1/2}AC^{1/2}\boldsymbol{\beta} = C^{1/2}\mathbf{f}, \qquad \boldsymbol{\alpha} = C^{1/2}\boldsymbol{\beta}. \tag{4.3}$$

It is important to note that although it is possible to get a nonnegative (nonpositive) definite matrix in (3.5) by the substitution $\boldsymbol{\beta} = C\boldsymbol{\alpha}$, the preconditioning CAC is not of the right order. The application of $C^{1/2}$ on both sides of A results in finite differences of the correct order (the same as those obtained in CA), due to the radial structure of the basis functions. The conjugate gradient scheme for the system (4.3) is of the form (3.10):

$$\boldsymbol{\beta}^{(k+1)} = \rho_{k+1}[\boldsymbol{\beta}^{(k)} + w_k C^{1/2}(\mathbf{f} - AC^{1/2}\boldsymbol{\beta}^{(k)})] + (1-\rho_{k+1})\boldsymbol{\beta}^{(k+1)}, \qquad k \geqq 0 \tag{4.4}$$

with

$$w_k = \frac{\eta_k}{\mathbf{t}^{(k)T}C^{1/2}AC^{1/2}\mathbf{t}^{(k)}}, \qquad \rho_{k+1} = \left[1 - \frac{w_k}{w_{k-1}}\frac{\eta_k}{\eta_{k-1}}\frac{1}{\rho_k}\right]^{-1}, \tag{4.5}$$

where $\rho_1 = 1$, $\boldsymbol{\beta}^{(0)} = 0$, $\mathbf{r}^{(0)} = C^{1/2}\mathbf{f}$, $\boldsymbol{\beta}^{(-1)} = \mathbf{0}$, and

$$\mathbf{t}^{(k)} = C^{1/2}\mathbf{r}^{(k)} = C^{1/2}[\mathbf{f} - AC^{1/2}\boldsymbol{\beta}^{(k)}], \qquad \eta_k = \mathbf{t}^{(k)T}\mathbf{t}^{(k)}. \tag{4.6}$$

The application of this scheme in terms of $\boldsymbol{\alpha}^{(k)} = C^{1/2}\boldsymbol{\beta}^{(k)}$ avoids the explicit computation of $C^{1/2}$. Multiplying (4.4) by $C^{1/2}$ from the left, we obtain the relation

$$\boldsymbol{\alpha}^{(k+1)} = \rho_{k+1}[\boldsymbol{\alpha}^{(k)} + w_k C(\mathbf{f} - A\boldsymbol{\alpha}^{(k)})] + (1-\rho_{k+1})\boldsymbol{\alpha}^{(k-1)}, \tag{4.7}$$

which is identical with (3.10). The parameters (4.10), (4.11) are computed in terms of C and $\boldsymbol{\alpha}^{(k)}$ as follows:

$$\mathbf{r}^{(k)} = \mathbf{f} - A\boldsymbol{\alpha}^{(k)}, \quad \eta_k = \mathbf{r}^{(k)T}C\mathbf{r}^{(k)}, \quad w_k = \frac{\eta_k}{\mathbf{r}^{(k)T}CAC\mathbf{r}^{(k)}} \tag{4.8}$$

with ρ_{k+1} as in (4.5). The initial values are $\rho_1 = 1$, $\boldsymbol{\alpha}^{(0)} = \boldsymbol{\alpha}^{(-1)} = \mathbf{0}$. The computation of (4.7) and (4.8) can be organized to require only one application of C and one of A in each iteration: Given $\mathbf{r}^{(k-1)}$, $\boldsymbol{\alpha}^{(k-1)}$, η_{k-1}, w_{k-1}, ρ_k, $\mathbf{r}^{(k)}$, $\boldsymbol{\alpha}^{(k)}$, compute:

$$\begin{aligned}
&\mathbf{x}^{(k)} = C\mathbf{r}^{(k)}, \quad \mathbf{y}^{(k)} = A\mathbf{x}^{(k)}, \quad \eta_k = \mathbf{r}^{(k)T}\mathbf{x}^{(k)},\\
&w_k = \frac{\eta_k}{\mathbf{x}^{(k)T}\mathbf{y}^{(k)}}, \qquad \rho_{k+1} = \left[1 - \frac{w_k\eta_k}{w_{k-1}\eta_{k-1}}\frac{1}{\rho_k}\right]^{-1},\\
&\boldsymbol{\alpha}^{(k+1)} = \rho_{k+1}[\boldsymbol{\alpha}^{(k)} + w_k\mathbf{x}^{(k)}] + (1-\rho_{k+1})\boldsymbol{\alpha}^{(k-1)},\\
&\mathbf{r}^{(k+1)} = \rho_{k+1}[\mathbf{r}^{(k)} - w_k\mathbf{y}^{(k)}] + (1-\rho_{k+1})\mathbf{r}^{(k-1)}.
\end{aligned} \tag{4.9}$$

The conjugate gradient (CG) scheme achieves rates of convergence comparable with the optimal Chebyshev (OC) scheme, while it does not require estimation of the spectral boundaries $[\lambda_{M+1}, \lambda_N]$.

Tables 4, 5 display the reduction in residuals by the CG and OC schemes for certain interpolation problems.

The iteration schemes considered above do not exploit the specific structure of the preconditioned problem (3.5). The preconditioning with the matrix C yields a matrix AC in (3.7) with special spectral properties. Let $\mathbf{u}^{(1)}, \cdots, \mathbf{u}^{(N)}$ be the eigenvectors of AC as in Theorem 3.1, normalized by

$$R(\mathbf{u}^{(j)}) = \mathbf{u}^{(j)T} C \mathbf{u}^{(j)} = 1, \qquad j = M+1, \cdots, N. \tag{4.10}$$

With this normalization (3.14) becomes

$$\mathbf{u}^{(i)T} C \mathbf{u}^{(j)} = \delta_{ij}, \qquad i, j = M+1, \cdots, N. \tag{4.11}$$

For any vector $\mathbf{f} \in R^N$, $\mathbf{f} = \sum_{i=1}^{N} \theta_i \mathbf{u}^{(i)}$ where by (4.11)

$$\theta_i = \mathbf{f}^T C \mathbf{u}^{(i)}, \qquad i = M+1, \cdots, N, \tag{4.12}$$

and

$$R(\mathbf{f}) = \mathbf{f}^T C \mathbf{f} = \sum_{i=M+1}^{N} \theta_i^2. \tag{4.13}$$

Since by (3.1) $\mathbf{f} - A\boldsymbol{\alpha} \in \mathbb{E}$,

$$A\boldsymbol{\alpha} = \sum_{i=1}^{M} \tilde{\theta}_i \mathbf{u}^{(i)} + \sum_{i=M+1}^{N} \theta_i \mathbf{u}^{(i)}, \tag{4.14}$$

and the scheme attempts to retrieve the component $\sum_{i=M+1}^{N} \theta_i \mathbf{u}^{(i)}$ of $A\boldsymbol{\alpha}$. Thus regarding the residuals in (3.7)

$$\mathbf{r}^{(k)} = P_k(AC)\mathbf{f} = \sum_{i=M+1}^{N} \theta_i P_k(\lambda_i)\mathbf{u}^{(i)} + \sum_{i=1}^{M} \theta_i \mathbf{u}^{(i)}, \tag{4.15}$$

the aim of the scheme is to reduce the sum $\sum_{i=M+1}^{N} \theta_i P_k(\lambda_i)\mathbf{u}^{(i)}$ below a certain small value.

In many applications the data $\mathbf{f}$ is taken from a surface which is smooth relative to the grid of data points, namely a surface exhibiting small changes along distances of the order of the typical distance h between neighboring data points. It is evident from numerical testing, that for such vectors $\mathbf{f}$ the components θ_i corresponding to the first eigenvalues are negligible in their contribution to $R(\mathbf{f})$. Thus the significant coefficients among $\{\theta_i | M+1 \leqq i \leqq N\}$ in (4.13) are $\{\theta_i | N(\mathbf{f}) \leqq i \leqq N\}$, where $N(\mathbf{f})$ increases with the relative smoothness of the data. This property of the spectrum of AC will be referred to as the S-property.

Numerical evidence of the S-property of the spectrum of AC, and the dependence of $N(\mathbf{f})$ on the relative smoothness of $\mathbf{f}$ are presented in subsection 6.3.

The numerical experiments [10] indicate that the S-property of AC corresponding to the basis functions $\psi(r) = \sqrt{r^2 + d^2}$ and $\psi(r) = \log(r^2 + d^2)$ depends on the size of d. For $d = h/2$ the separation of the spectrum is inferior to that for $d = h$. For the TPS the S-property of AC is not found. Yet the S-property is found for $\psi(r) = (r^2 + d^2)\log(r^2 + d^2)$ the shifted TPS, with $d = h$ (the average grid distance) [10]. This basis of radial functions determines a nonsingular system of the form (2.6) [9], as well as all the "shifted surface splines".

Our conjecture about the S-property of the spectrum of AC may also be stated as: the smaller positive eigenvalues of AC correspond to the rougher eigenvectors. This conjecture is borrowed from the univariate case for which the S-property is fully

understood: starting with the radial basis functions $\psi_i(x) = |x - t_i|^m$, $i = 1, \cdots, N$, and preconditioning by divided differences of order m at the points $t_1, \cdots, t_N$, the resulting matrix AC is of the form $\{B_j(t_i)\}_{i,j=1}^N$ where $B_1, \cdots, B_N$ are the univariate B-splines corresponding to the knots $t_1, \cdots, t_N$. This matrix is known to be totally positive [1] and therefore the smoother eigenvectors (eigenvectors with fewer sign changes) correspond to larger eigenvalues [2].

Assuming that the spectrum of AC possesses the S-property, we can modify the choice of the polynomials $P_k(w)$ in (3.6) and (3.7) to achieve better iteration schemes. Since $\lambda_{N(\mathbf{f})}$ is the smallest "active" eigenvalue of a given data $\mathbf{f}$, it can replace λ_{M+1} in the determination of the parameters of the optimal Chebyshev (OC) scheme, namely in (4.1) $\sigma' = (\lambda_N - \lambda_{N(\mathbf{f})})/(\lambda_N + \lambda_{N(\mathbf{f})})$ replaces σ. It is clear from (4.2) that the convergence rate of this OC scheme improves with the increase of $N(\mathbf{f})$. This is demonstrated by numerical tests in Tables 3, 4, 5. Yet it is the CG scheme which achieves the best rates. Although the parameters of the CG scheme are independent of the interval of "active" eigenvalues in $\mathbf{f}$, its performance improves significantly with the increase of $N(\mathbf{f})$. The dependence of the performance of the CG scheme on the actual number of "active" eigenvectors in the data is well known [7].

We conclude this section by a brief description of the application of the power method [7] to the estimation of $\lambda_{M+1}, \lambda_N, \lambda_{N(\mathbf{f})}$.

A good initial vector for estimating λ_N, $\mathbf{x}^{(0)} = \sum_{i=1}^N \theta_i \mathbf{u}^{(i)}$ should have a large coefficient θ_N relative to the other coefficients. Thus $\mathbf{x}^{(0)}$ should be a smooth vector relative to the grid. Using the orthogonality (3.14), the approximation of λ_N is computed recursively as follows:

Starting with $\mathbf{x}^{(0)}$, we define the sequence $\{\mathbf{x}^{(i)}\}$ by

$$\begin{aligned} \mathbf{y}^{(i)} &= \mathbf{x}^{(i)} \cdot (\mathbf{x}^{(i)T} C \mathbf{x}^{(i)})^{-1/2}, \\ \mathbf{x}^{(i+1)} &= AC\mathbf{y}^{(i)}, \end{aligned} \tag{4.16}$$

and a sequence of approximations to λ_N

$$\lambda_N^{(i+1)} = \mathbf{x}^{(i+1)T} C \mathbf{y}^{(i)}. \tag{4.17}$$

To find the smallest nonzero eigenvalue of AC, λ_{M+1}, we apply the power method to the matrix $CA - \lambda_N I$, with an initial vector $\mathbf{x}^{(0)} = C\mathbf{y}^{(0)} \in \mathbb{E}^\perp$, where $\mathbf{y}^{(0)}$ is a "rough" vector. For example, a vector with random elements is a good choice. Actually by this method we find the smallest nonzero eigenvalue corresponding to the eigenvector expansion of $\mathbf{y}^{(0)}$, $\mathbf{y}^{(0)} = \sum \theta_i \mathbf{u}^{(i)}$. Starting the procedure with $\mathbf{y}^{(0)} = \mathbf{f}$, we get, after few iterations, a good estimate of $\lambda_{N(\mathbf{f})}$ [10].

5. Methods for smoothing. The major object of this work is to develop methods for smoothing noisy data measured at scattered data points. The S-property of the spectrum of the iteration matrix AC is the basis of our approach to smoothing.

Let us assume that the data is of the form

$$\mathbf{f} = \mathbf{g} + \mathbf{h} \tag{5.1}$$

with $\mathbf{g}$ the smooth part, i.e. taken from a smooth bivariate function, and $\mathbf{h}$ the noisy part of $\mathbf{f}$. By the S-property of the spectrum of AC we know that $\mathbf{g}$ can be expanded in the eigenvectors of AC as

$$\mathbf{g} = \sum_{i=1}^{M} \theta_i \mathbf{u}^{(i)} + \sum_{i=N(\mathbf{g})}^{N} \theta_i \mathbf{u}^{(i)}, \tag{5.2}$$

with $N(\mathbf{g}) > M+1$. We further assume that the noisy part $\mathbf{h}$ has the expansion

$$\mathbf{h} = \sum_{i=M+1}^{N(\mathbf{g})-1} \theta_i \mathbf{u}^{(i)}. \tag{5.3}$$

Now suppose we perform iterations of the form (3.8) with a polynomial $P_k(w)$ such that $P_k(0) = 1$ and

$$\begin{aligned} P_k(\lambda_i) &= 0, \qquad N \geqq i \geqq N(\mathbf{g}), \\ P_k(\lambda_i) &= 1, \qquad 0 < i < N(\mathbf{g}). \end{aligned} \tag{5.4}$$

Then, by (3.7) and by (5.1)-(5.4)

$$\mathbf{r}^{(k)} = P_k(AC)\mathbf{f} = \sum_{i=1}^{N} \theta_i P_k(\lambda_i)\mathbf{u}^{(i)} = \sum_{i=1}^{M} \theta_i \mathbf{u}^{(i)} + \sum_{i=M+1}^{N(\mathbf{g})-1} \theta_i \mathbf{u}^{(i)} = \sum_{i=1}^{M} \theta_i \mathbf{u}^{(i)} + \mathbf{h}. \tag{5.5}$$

Since $\mathbf{r}^{(k)} = \mathbf{f} - A\boldsymbol{\alpha}^{(k)}$, (5.5) implies that $A\boldsymbol{\alpha}^{(k)} = \mathbf{g} - \sum_{i=1}^{M} \theta_i \mathbf{u}^{(i)}$, i.e. the noisy part $\mathbf{h}$ is completely annihilated by a scheme with $P_k(w)$ satisfying (5.4).

In order to retrieve the polynomial part of $\mathbf{g}$, $\sum_{i=1}^{M} \theta_i E_{\cdot i}$ from

$$r^{(k)} = \sum_{i=1}^{M} \theta_i E_{\cdot i} + \sum_{i=M+1}^{N(\mathbf{g})-1} \theta_i \mathbf{u}^{(i)},$$

the residuals $\mathbf{r}^{(k)}$ are fitted by a vector $E\boldsymbol{\gamma}^{(k)}$ in the least-squares sense. Since the vectors $\{\mathbf{u}^{(i)},\ M+1 \leqq i \leqq N(\mathbf{g})-1\}$ are rough relative to the grid of points and the vectors $\{E_{\cdot i},\ 1 \leqq i \leqq M\}$ are smooth, the method of least-squares yields the desired separation. The resulting solution of the smoothing problem

$$\sum_{i=1}^{N} \alpha_i^{(k)} \psi(r_i(x, y)) + \sum_{i=1}^{M} \gamma_i^{(k)} q_i(x, y), \qquad \boldsymbol{\alpha} \in \mathbb{E}^{\perp},$$

approximates well the smooth part $\mathbf{g}$ of the data $\mathbf{f}$ on the set of data points.

In view of the S-property of the spectrum of AC the above smoothing technique is actually a low pass filter, or smooth-pass filter, since all the rough components in $\mathbf{f}$, i.e. components with eigenvalues $< \lambda_{N(\mathbf{g})}$ are annihilated.

In practice the separation of the smooth part from the noisy part in the data as displayed by (5.2)-(5.3) is not sharp, and (5.2)-(5.3) can be regarded as representing the significant parts of $\mathbf{g}$ and $\mathbf{h}$ only. Also $N(g)$ is not known and it is impossible to generate a polynomial of a practical low order so that (5.4) is satisfied exactly.

Let us suppose that we are given a cut-off value λ_c (not necessarily an eigenvalue of AC). Then we would like to find a low-order polynomial $P_k(w)$ approximating the step function

$$U(w) = \begin{cases} 0, & \lambda_c \leqq w \leqq \lambda_N, \\ 0, & 0 \leqq w \leqq \lambda_c, \end{cases} \tag{5.6}$$

and for consistency we must require $P_k(0) = 1$.

DeVore [3] developed polynomial approximations to step functions for the purpose of analyzing shape preserving splines. These are polynomials on $[-1, 1]$ obtained from the Chebyshev polynomials by the relation

$$D_{m,r,j}(x) = c \int_{-1}^{x} \left[\frac{T_m(t)}{t - t_j} \right]^r dt, \quad t_j = \cos \frac{2j-1}{2m} \pi, \quad j = 1, 2, \cdots, m. \tag{5.7}$$

The integrand in (5.7) is an approximation to $\delta(t - t_j)$, and thus $D_{m,r,j}(x)$ approximates

a step function with a jump from 0 to 1 at the point t_j. The parameter c is chosen so that $D_{m,r,j}(1)=1$. We note that in [3] only m odd and $t_j=0$ are considered. The parameter m controls the width of the increasing part around t_j, and r determines the quality of the approximation to the constant values $U=0$ and $U=1$. Figures 7, 8 display some of these polynomials and exhibit very nice approximations to step functions. We note that DeVore polynomials with an even parameter r are monotone, thus satisfying the requirements of Theorem 3.3.

We use DeVore polynomials to get the desired polynomial approximation to $U(w)$. First we approximate λ_N by the power method (4.16)-(4.17); then we choose appropriate m and j such that $t_j \simeq 2\lambda_c/\lambda_N - 1$. The smoothing polynomial is now defined as

$$P_k(w) = 1 - D_{m,r,j}\left(\frac{2w}{\lambda_N} - 1\right), \qquad k=(m-1)r+1. \tag{5.8}$$

This polynomial determines the parameters $\{w_i\}_{i=1}^{k}$ for the iterations (3.8): $w_i = 1/Z_i$, $i=1,\cdots,k$ where the Z_i's are roots of $P_k(w)$. Some of these roots might be complex, but then they come as complex conjugates, and the iteration can be performed as a two-step real iteration.

In § 6.5 we demonstrate the results obtained in smoothing noisy data with a scheme based on DeVore polynomials. For those with even r the discrete roughness of the computed surface is always below the roughness of the noisy data, by Theorem 3.3.

DeVore polynomials can be used also in smoothing univariate noisy data by splines, due to the S-property of the matrix of B-splines. Agee's procedure [2] is not a low-pass filter, since it corresponds to a Richardson scheme with $w_0=1$, $P_k(w)=(1-w)^k$ in (3.7). In fact this procedure converges too rapidly to the interpolating spline.

6. Numerical examples, tables and figures.

6.1. Test grids (Figs. 1–3) and test functions. For the numerical experiments we have used three types of data sets:

I. Square grid on the square $[-\frac{1}{2},\frac{1}{2}]\times[-\frac{1}{2},\frac{1}{2}]$, Fig. 1.

II. A grid on the circle $x^2+y^2\leqq\frac{1}{4}$, Fig. 2.

III. A grid on the sector $x, y\geqq 0$ of the unit circle, Fig. 3.

The grids II and III are defined by simple transformations of grid I:

$$\text{I}\to\text{II}: \qquad (x,y)\to\frac{\max(|x|,|y|)}{\sqrt{x^2+y^2}}\cdot(x,y)$$

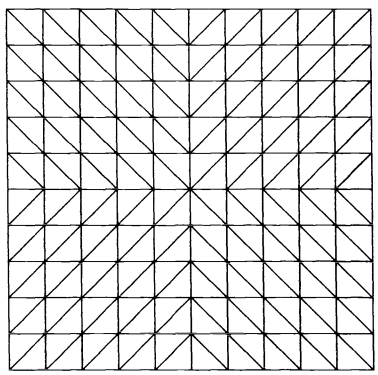


FIG. 1. *Grid* I.

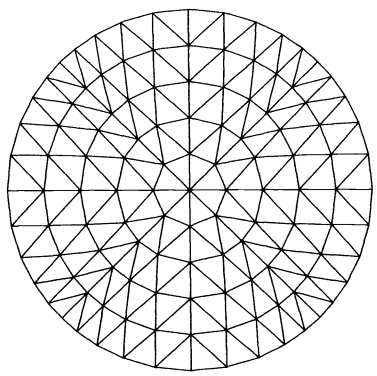


FIG. 2. *Grid* II.

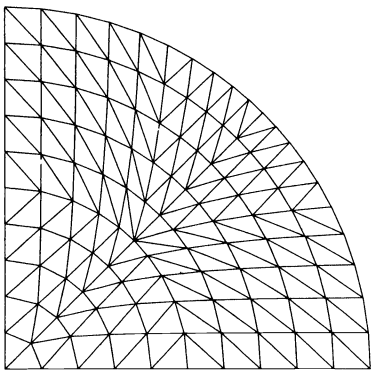


FIG. 3. *Grid* III.

$$\text{I} \to \text{III}: \qquad (x, y) \to \frac{\max\left(\left|x+\tfrac{1}{2}\right|, \left|y+\tfrac{1}{2}\right|\right)}{\sqrt{(x+\tfrac{1}{2})^2+(y+\tfrac{1}{2})^2}} \cdot (x+\tfrac{1}{2}, y+\tfrac{1}{2})$$

By this method of grid generation we get a nontrivial mesh of evenly distributed points, and for the triangulation we can still use the simple data base for triangulation on the square grid.

The test functions are taken from [6]:

$$f_1(x, y) = 0.5 \exp\left[-\frac{(9x-2)^2+(9y-2)^2}{4}\right] + 0.75 \exp\left[-\frac{(9x+1)^2}{49} - \frac{9y+1}{10}\right]$$
$$+0.5 \exp\left[-\frac{(9x-7)^2+(9y-3)^2}{4}\right] - 0.2 \exp\left[-(9x-4)^2-(9y-7)^2\right],$$

$$f_2(x, y) = \tfrac{1}{9}\left[\tanh(9y-9x)+1\right],$$

$$f_5(x, y) = \tfrac{1}{3} \exp\left[-\tfrac{81}{4}((x-\tfrac{1}{2})^2+(y-\tfrac{1}{2})^2)\right],$$

$$f_6(x, y) = \tfrac{1}{9}[64-81((x-\tfrac{1}{2})^2+(y-\tfrac{1}{2})^2)]^{1/2}-\tfrac{1}{2}.$$

These functions are defined on $[0, 1]\times[0, 1]$ and we use the appropriate translation to transform them to $[-\frac{1}{2}, \frac{1}{2}]\times[-\frac{1}{2}, \frac{1}{2}]$ for grids I and II. On each of these grids the functions are normalized so that $\max|f_i| = 1$.

6.2. The effect of conditioning (Tables 1–2). Tables 1-2 summarize the condition numbers of the original linear system (2.6) and the conditioned system (2.15), for interpolation by three different sets of basis functions: TPS, Hardy multiquadrics (HM)

TABLE 1
Condition numbers of the original systems (2.6).

	Grid I		Grid II		Grid III	
Basis functions \ N	49	121	49	121	49	121
TPS	1181	6764	1885	12633	2644	16107
HM	10556	7274	17059	107333	35059	218667
SL	605	1868	1467	6633	2721	11542

TABLE 2
Condition numbers of the conditioned systems (2.15).

	Grid I		Grid II		Grid III	
Basis functions \ N	49	121	49	121	49	121
TPS	4.3	5.1	3.4	3.9	41.5	116.7
HM	69.2	126	222.8	576.0	393.0	976.5
SL	14.7	19.5	43.7	81.1	66.5	119.4

and shifted logarithmics (SL). The shift d in the last two sets of basis functions equals the average mesh size. The data points are taken on three different grids and their number is 49 and 121.

6.3. Evidence of the *S*-property (Figs. 4–6, Table 3). The following figures demonstrate the S-property of AC for the case of the HM functions. Similar clear results were obtained with the shifted logarithmic functions and the shifted TPS [10].

The eigenvalues of AC for Hardy multiquadrics on mesh II with $N = 121$, are displayed in Fig. 4. For the same case we examine the magnitude of the coefficients in the eigenvector expansions of data f taken from functions:

$$\sin(\pi kx)\sin(\pi ly).$$

It is clear that as k^2+l^2 increases, the function becomes rougher. Indeed we find that as k^2+l^2 increases, the significant coefficients in the expansion correspond to eigenvalues which are closer to zero. The graphs in Fig. 5 bound from above the coefficients $\{\theta_i\}$ for $k = l = 1, 3, 5, 7, 9, 11$.

Figure 6 displays upper bounds of the coefficients in the eigenvector expansion of f_1, f_2, f_5, f_6 and a random function (f_7), on mesh II with $N = 121$. For the smooth functions f_5 and f_6 the cut-off index $N(f)$ is very high, >96, while for the random function f_7 we observe that all the significant coefficients correspond to λ_i's with $i < 72$.

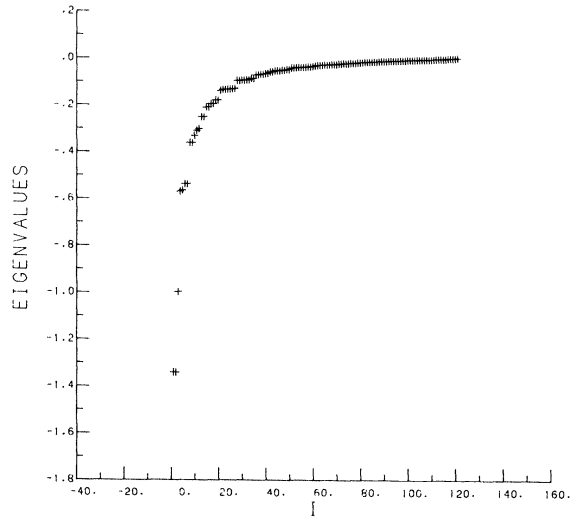


FIG. 4. *The eigenvalues of* AC *for Hardy multiquadrics on mesh* II, $N = 121$.

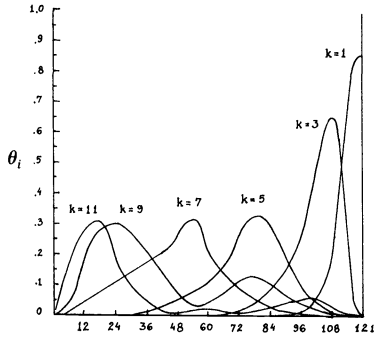


FIG. 5. *Upper bounds for the magnitude of the coefficients in the eigenvector expansions of the functions* $\sin(\pi kx)\sin(\pi ky)$, $k = 1, 3, 5, 7, 9, 11$.

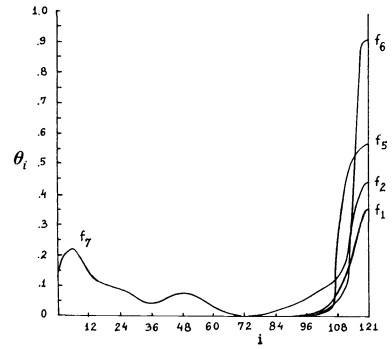


FIG. 6. *Upper bounds for the magnitude of the coefficients in the eigenvector expansions of the functions* f_1, f_2, f_5, f_6, f_7.

TABLE 3
Cut-off index $N(\mathbf{f})$ *for different functions on mesh* II, $N = 121$.

	$\sin(x)\sin(y)$	$\sin(3x)\sin(3y)$	$\sin(5x)\sin(5y)$	f_1	f_2	f_5	f_6
HM	90	60	30	90	72	96	96
SL	80	60	40	86	72	110	100

6.4. Comparison of iterative methods for interpolation (Tables 4–5). Tables 4, 5 summarize a set of numerical experiments which compare the reduction in the residuals by the three iteration schemes: Conjugate Gradient (CG), Optimal Chebyshev on $[\lambda_{M+1}, \lambda_N]$ (OC_1) and Optimal Chebyshev on $[\lambda_{N(\mathbf{f})}, \lambda_N]$ (OC_2). The tested methods of interpolation involve the radial basis functions as in Table 1. The data is prescribed on $N = 121$ points of Grid II. It is taken from the test functions $f_1(x, y)$, $f_5(x, y)$—in Table 4, and from $\bar{f}_1(x, y) = f_1(x, y) + \varepsilon_1(x, y)$ and $\bar{f}_5(x, y) = f_5(x, y) + \varepsilon_5(x, y)$—in Table 5, where $\varepsilon_i(x, y)$ is a uniformly random noise in $[-0.1, 0.1]$. For the noisy data case OC_2 is identical with OC_1.

The reduction in the residuals is measured by two quantities:

(a) $R^{1/2}(\mathbf{r}^{(k)})/R^{1/2}(\mathbf{f})$, where $\mathbf{r}^{(k)} = \mathbf{f} - A\boldsymbol{\alpha}^{(k)}$, and

(b) $\|S^{(k)}\|_2/\|C\mathbf{f}\|_2$, where $\mathbf{S}^{(k)} = C\mathbf{f} - CA\alpha^{(k)}$ is the residual vector of the conditioned system (2.15).

TABLE 4

Comparison of the reduction in the residuals by different iteration schemes in case of smooth data.

		$f_1(x, y)$						$f_5(x, y)$					
Basis functions	k	$R^{1/2}(\mathbf{r}^{(k)})/R^{1/2}(\mathbf{f})$			$\|S^{(k)}\|_2/\|C\mathbf{f}\|_2$			$R^{1/2}(\mathbf{r}^{(k)})/R^{1/2}(\mathbf{f})$			$\|\mathbf{S}^{(k)}\|_2/\|C\mathbf{f}\|_2$		
		CG	OC_1	OC_2	CG	OC_1	OC_2	CG	OC_1	OC_2	CG	OC_1	OC_2
HM	5	.7E−1	.4E0	.7E−1	.9E−1	.5E−1	.2E−1	.4E−1	.3E0	.3E−1	.4E−3	.1E0	.2E−1
	10	.1E−1	.1E0	.4E−1	.2E−1	.1E−1	.1E−1	.4E−4	.9E−1	.1E−1	.1E−3	.5E−1	.6E−2
	20	.1E−2	.9E−2	.2E−1	.2E−2	.2E−2	.7E−2	.3E−5	.8E−2	.2E−2	.8E−5	.3E−2	.1E−2
	40	.3E−4	.9E−3	.6E−2	.7E−4	.5E−3	.3E−2	.1E−6	.6E−4	.3E−3	.2E−6	.4E−4	.3E−3
SL	5	.1E−1	.2E0	.1E−1	.3E−1	.5E−1	.5E−2	.3E−3	.4E0	.2E−2	.8E−3	.1E0	.1E−2
	10	.2E−2	.3E−1	.4E−2	.5E−2	.6E−2	.2E−2	.5E−5	.7E−1	.2E−3	.2E−4	.3E−1	.3E−3
	20	.5E−4	.2E−2	.1E−2	.1E−3	.2E−3	.6E−3	.4E−7	.4E−2	.7E−4	.1E−6	.1E−2	.1E−3
	40	CON	.6E−5	.3E−3	CON	.2E−5	.1E−3	CON	.2E−4	.1E−4	CON	.6E−5	.2E−4
TPS	5	.2E−2	.5E−2	—	.3E−2	.1E−1	—	.3E−3	.4E−2	—	.1E−2	.8E−2	—
	10	.8E−5	.2E−4	—	.2E−4	.4E−4	—	.7E−6	.2E−4	—	.3E−5	.6E−4	—
	15	CON	.1E−5	—	CON	.1E−6	—	CON	.7E−7	—	CON	.1E−6	—
	20	—	.1E−5	—	—	.5E−9	—	—	.3E−7	—	—	.7E−9	—

TABLE 5
Comparison of the reduction in the residuals by different iteration schemes in case of noisy data.

		$f_1(x,y)+\varepsilon_1(x,y)$				$f_5(x,y)+\varepsilon_5(x,y)$			
Basis functions	k	$R^{1/2}(\mathbf{r}^{(k)})/R^{1/2}(\mathbf{f})$		$\|\mathbf{S}^{(k)}\|_2/\|C\mathbf{f}\|_2$		$R^{1/2}(\mathbf{r}^{(k)})/R^{1/2}(\mathbf{f})$		$\|\mathbf{S}^{(k)}\|_2/\|C\mathbf{f}\|_2$	
		CG	OC_1	CG	OC_1	CG	OC_1	CG	OC_1
HM	5	.1E1	.7E0	.8E0	.4E0	.8E0	.5E0	.7E0	.8E0
	10	.5E0	.5E0	.4E0	.3E0	.3E0	.4E0	.3E0	.6E0
	20	.1E0	.3E0	.1E0	.2E0	.1E0	.3E0	.1E0	.4E0
	40	.2E−2	.2E0	.8E−3	.1E0	.5E−2	.2E0	.6E−3	.3E0
SL	5	.6E0	.3E0	.4E0	.2E0	.3E0	.3E0	.5E0	.4E0
	10	.2E0	.2E0	.2E0	.1E0	.1E0	.1E0	.1E0	.2E0
	20	.4E−2	.8E−1	.3E−2	.5E−1	.2E−2	.7E−1	.2E−2	.9E−1
	40	.1E−5	.2E−1	.9E−6	.1E−1	.7E−6	.2E−1	.6E−6	.2E−1
TPS	5	.4E−2	.6E−2	.2E−2	.5E−2	.3E−2	.6E−2	.2E−2	.5E−2
	10	.8E−5	.2E−4	.4E−5	.1E−4	.7E−5	.2E−4	.4E−5	.1E−4
	15	CON	.3E−6	CON	.7E−7	.1E−7	.8E−7	.1E−7	.7E−7
	20	—	.2E−6	—	.3E−9	CON	.2E−7	CON	.3E−9

In the above tables CON denotes convergence, and specifies a situation in which $R(\mathbf{r}^{(k)})$ is of the order 10^{-14} and is negative due to round-off errors. At this stage the CG scheme has to be stopped.

Conclusions from Tables 4, 5. It is evident from the numbers in Table 4 that the smoother the data (f_5 is smoother than f_1), the faster the convergence of the CG scheme and the OC_2 scheme. These two schemes theoretically have similar rates of convergence, since the CG scheme depends on the actual distribution of coefficients in the eigenvector expansion of the data [7]. Yet the CG scheme is obviously superior to the OC_2 scheme. This can be explained by the fact that the smooth data has small but not negligible components in $\{\mathbf{u}^{(j)}, M+1 \leqq j \leqq N(\mathbf{f})\}$, and the OC_2 scheme increases these components in $\mathbf{r}^{(k)}$.

On the other hand, as is expected, the improvement in the performance of the OC_1 scheme for a smoother data is only minor.

Table 5 demonstrates the superiority of the CG scheme over the OC_1 scheme when the data is rough and has components in all the eigenvectors of AC. The deterioration in the performances of both schemes in case of noisy data is significant.

These tables are in agreement with Tables 1-2. Also here the TPS basis exhibits significantly faster convergence rates when compared to the HM or SL basis.

6.5. Smoothing with DeVore polynomials (Figs. 7–11). In Figs. 7, 8 we give examples of approximations to step functions by DeVore polynomials:

The final example deals with smoothing data on grid II with $N = 121$, using Hardy multiquadrics as basis functions. The data is taken from the function f_1 with the addition of a noise which is uniformly random on $[-0.1, 0.1]$. Figure 9 displays the surface defined by the function f_1 and Fig. 10 the surface obtained by interpolating the noisy data with Hardy multiquadrics. Using iterations as described in § 5, based on DeVore polynomial $D_{7,2,2}$ we obtained the surface displayed in Fig. 11.

Table 6 summarizes several smoothing experiments on grid II with $N = 121$, by comparing the maximal deviation over 1600 points in the domain, between the approximated function (without noise) and four approximating surfaces: the interpolating

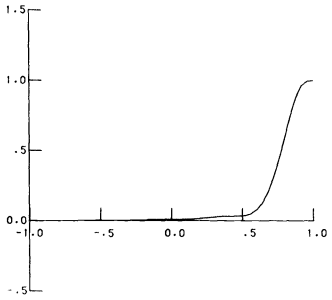


FIG. 7. *DeVore polynomial* $D_{7,2,2}$.

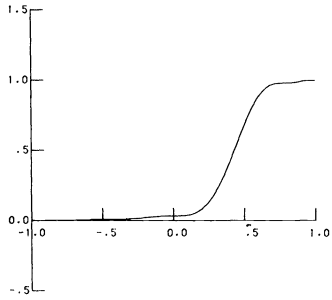


FIG. 8. *DeVore polynomial* $D_{7,2,3}$.

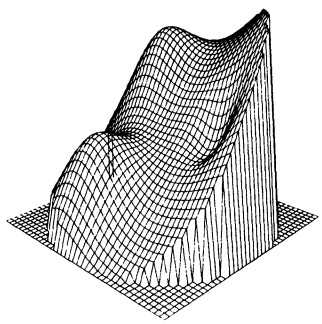


FIG. 9. *Test function* $f1$.

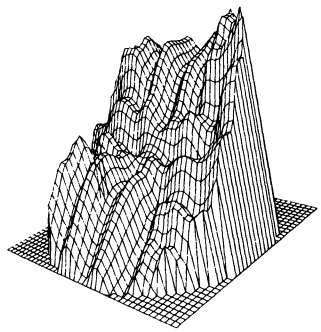


FIG. 10. *Interpolation to* $f1$ *with noise.*

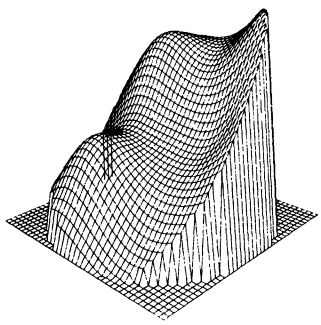


FIG. 11. *Smoothing achieved with DeVore polynomial* $D_{7,2,2}$.

TABLE 6

Comparison of the maximal deviation in approximation by interpolating and smoothing surfaces.

Basis functions	Approximated function	IE	IN	SN_1	SN_2
HM	f_1	.02	.14	.05	.16
SL	f_1	.02	.13	.06	.05
HM	f_5	.001	.12	.05	.31
SL	f_5	.001	.12	.05	.03

surface to the exact data (IE), the interpolating surface to the noisy data (IN), the smoothing surface to the noisy data obtained by DeVore polynomial $D_{7,2,2}$ (SN_1), and the smoothing surface to the noisy data obtained by DeVore polynomial $D_{7,2,3}$(SN_2). It is evident from Table 6 that smoothing with the SL basis is less sensitive to the location of the step in the DeVore polynomial as compared to the HM basis.

Further numerical tests can be found in [10].

REFERENCES

[1] C. DE BOOR, *A Practical Guide to Splines*, Springer-Verlag, New York-Heidelberg-Berlin, 1978.

[2] ———, *How does Agee's smoothing method work*? ARO report 79-3, Proc. 1979 Army Numerical Analysis and Computer Conference, pp. 299-302.

[3] R. A. DEVORE, *Monotone approximation by polynomials*, SIAM J. Math. Anal., 8 (1977), pp. 906-921.

[4] J. DUCHON, *Interpolation des functions de duex variables suivant le principe de la flexion des plaques minces*, RAIRO Anal. Numér., 10 (1976), pp. 5-12.

[5] N. DYN AND D. LEVIN, *Iterative solution of systems originating from integral equations and surface interpolation*, SIAM J. Numer. Anal., 20 (1983), pp. 377-390.

[6] R. FRANKE, *Scattered data interpolation: Tests of some methods*, Math. Comp., 38 (1982), pp. 181-200.

[7] L. A. HAGEMAN AND D. M. YOUNG, *Applied Iterative Methods*, Academic Press, New York, 1981.

[8] C. L. LAWSON, *Software for C^1 surface interpolation*, in Mathematical Software III, J. R. Rice, ed., Academic Press, New York, 1977.

[9] C. MICCHELLI, *Interpolation of scattered data: Distance matrices and conditionally positive definite functions*, to appear.

[10] S. RIPPA, *Interpolation and smoothing of scattered data by radial basis functions*, M.Sc. Thesis, Tel Aviv Univ., Tel Aviv, Israel, 1984.

[11] G. WAHBA, *Surface fitting with scattered data on Euclidean d-spaces and on the sphere*, Rocky Mountain J. Math., 14, No. 4 (1984), pp. 281-299.

SIAM J. SCI. STAT. COMPUT.
Vol. 7, No. 2, April 1986

021

LOCALIZATION OF SEARCH IN QUASI-MONTE CARLO METHODS FOR GLOBAL OPTIMIZATION*

HARALD NIEDERREITER† AND PAUL PEART‡

Abstract. Quasi-random search methods for the extreme values of multivariable functions suffer from the defect that they often require a very large number of function evaluations. We introduce and analyze the device of "localization of search" that speeds up these methods considerably.

Key words. nondifferentiable optimization, quasi-random search methods

1. Introduction. In nondifferentiable optimization the Monte Carlo method of random search can be used to approximate the global optimum of a function (see [15, Chap. 7]). A deterministic analogue of random search, which may be thought of as quasi-Monte Carlo optimization or quasi-random search, was introduced by Niederreiter [9]. Let f be a bounded real-valued function defined on the bounded subset E of $\mathbb{R}^s$, $s \geqq 1$, and let $x_1, \cdots, x_N$ be points in E. Then

$$m_N = \max_{1 \leqq n \leqq N} f(x_n)$$

is taken as an approximation for the correct value M of the supremum of f over E. The error between M and m_N can be bounded in terms of the modulus of continuity

$$\omega(t) = \sup_{\substack{x,y \in E \\ d(x,y) \leqq t}} |f(x) - f(y)|, \qquad t \geqq 0,$$

and in terms of the dispersion

$$d_N = d_N(E) = \sup_{x \in E} \min_{1 \leqq n \leqq N} d(x, x_n)$$

of $x_1, \cdots, x_N$, where d is a given metric on E. In fact, we have

$$M - m_N \leqq \omega(d_N) \tag{1}$$

according to [9, Thm. 1]. In this sense, the dispersion can be viewed as a measure for the effectiveness of the points $x_1, \cdots, x_N$ in searching the domain E.

If f is continuous on E and we take a dense sequence $x_1, x_2, \cdots$ of points in E, then it is easy to see that the search method described above converges, i.e. that $\lim_{N \to \infty} m_N = M$. However, the rate of convergence is in general very slow, since many points x_n can be expected to be rather far away from that part of E where the values of f are close to M. Therefore, in order to obtain accurate results for some functions, it may be necessary to carry out a very large number of function evaluations, and this can be an expensive exercise. In this paper we indicate how the method of "crude search" described above can be modified by using "localization of search" so as to speed up the convergence considerably.

2. Localization of search. Without loss of generality, we normalize the domain E to be the s-dimensional unit cube $I^s = [0, 1]^s$. Other domains of interest can be

* Received by the editors April 28, 1983, and in revised form August 1, 1984.
† Mathematical Institute, Austrian Academy of Sciences, A-1010 Vienna, Austria.
‡ Department of Mathematics, University of the West Indies, Kingston 7, Jamaica.

transformed into I^s by a suitable parametrization. It is convenient to work with the metric

$$d(y, z) = \max_{1 \leqq j \leqq s} |y_j - z_j|$$

for $y = (y_1, \cdots, y_s)$, $z = (z_1, \cdots, z_s) \in \mathbb{R}^s$, although we may proceed similarly with other metrics. Let $C(a; \varepsilon)$ denote the s-dimensional cube with edge length $2\varepsilon > 0$ and center $a \in I^s$, i.e.

$$C(a; \varepsilon) = \{y \in \mathbb{R}^s : d(a, y) \leqq \varepsilon\}.$$

For $C = C(a; \varepsilon)$ let $g_C : I^s \to C$ be the function defined by

$$g_C(x) = a + \varepsilon(2x - e) \quad \text{for } x \in I^s,$$

where $e = (1, \cdots, 1) \in \mathbb{R}^s$. Now let $x_1, \cdots, x_N$ be points in I^s, put

$$S(C) = \{1 \leqq n \leqq N : g_C(x_n) \in I^s\}$$

and define

$$m_N(C) = \max\left(f(a), \max_{n \in S(C)} f(g_C(x_n))\right).$$

Furthermore, let $x^*(C)$ be one of the points in the set $\{a\} \cup \{g_C(x_n) : n \in S(C)\}$ for which

$$m_N(C) = f(x^*(C)).$$

Let $\varepsilon_1, \varepsilon_2, \cdots$ be a sequence of positive numbers tending to zero. We define a sequence $C_0, C_1, \cdots$ of cubes by setting

$$C_0 = I^s,$$
$$C_i = C(x^*(C_{i-1}); \varepsilon_i) \quad \text{for } i = 1, 2, \cdots.$$

The nondecreasing sequence $m_N(C_0), m_N(C_1), \cdots$ is then taken as a sequence of approximations for the supremum M of f over I^s. Obvious modifications lead to a method for approximating the infimum of f over I^s.

In practice we will often use $\varepsilon_i = \varepsilon^i$ for $i = 1, 2, \cdots$ with a fixed ε satisfying $0 < \varepsilon < \frac{1}{2}$. The success of the method depends to some extent on the proper relationship between the dispersion d_N of the points $x_1, \cdots, x_N$ in I^s and the value of ε. In particular, the condition $d_N < \varepsilon$ should be observed. If f is continuous on I^s, so that $M = f(x^*)$ for some $x^* \in I^s$, then $d_N < \varepsilon$ and the definition of d_N imply that there is a point x_k of the set $\{x_1, \cdots, x_N\}$ satisfying $d(x^*, x_k) < \varepsilon$. If x_k can also be chosen so that $x_k = x^*(C_0)$, as will happen in most practical cases, then $d(x^*, x_k) < \varepsilon$ implies that $x^* \in C_1$. A similar heuristic reasoning can be applied to the general step from C_{i-1} to C_i, so in nonpathological cases we can expect that $x^* \in C_i$ for all $i \geqq 1$, i.e. that the method converges. In contrast to the crude search method, which demonstrably converges, it seems difficult to obtain a rigorous and satisfactory convergence result for localization of search. Even in the context of a stochastic model such a result seems hard to prove, as was pointed out by Solis and Wets [18, p. 21]. See also Devroye [3] for a related stochastic model.

When implementing localization of search, it is of course desirable to take N as small as possible to help in minimizing costs. On the other hand, N should be chosen large enough so that $d_N < \varepsilon$ holds. Since $d_N \geqq \frac{1}{2}N^{-1/s}$ is an absolute lower bound for

the dispersion of any N points in I^s according to [10], it follows that N must at least be of an order of magnitude ε^{-s}. More detailed information will be given in § 3 on the basis of numerical experiments.

In each step of the method of localization of search, the error between the supremum of f over C_i and $m_N(C_i)$ can be bounded effectively by (1). If the point sets $x_1, \cdots, x_N$ proposed in § 3 are used, then the dispersion can be calculated or at least bounded nontrivially from above. We note also that since the mapping $g_C : I^s \to C = C(a; \varepsilon)$ satisfies $d(x, y) = 2\varepsilon d(g_C(x), g_C(y))$ for all $x, y \in I^s$, the dispersion $d_N(C_i)$ of the points $g_{C_i}(x_1), \cdots, g_{C_i}(x_N)$ in C_i is given by $d_N(C_i) = 2\varepsilon_i d_N$ for $i \geqq 1$, where d_N is the dispersion of the points $x_1, \cdots, x_N$ in I^s. For the complicated functions f for which quasi-Monte Carlo optimization has to be applied, it may be difficult or impossible to obtain precise values of the modulus of continuity $\omega(t)$. However, it suffices for our purposes to work with upper bounds for $\omega(t)$, which may be more readily available.

A typical case in which convergence of the method of localization of search may be hard to achieve is the following. Suppose $x^* \in I^s$ is a point at which f attains its maximum over I^s and that f also has a local maximum at $u \in I^s$, with $f(x^*) - f(u)$ being a small positive number. Then localization of search could lead into a "wrong track", i.e. the sequence $x^*(C_i)$, $i = 0, 1, \cdots$, could converge to u instead of x^* if N is not sufficiently large. There are various possibilities of countervailing that tendency. One way is to carry out the initial search in I^s with a point set having as small a dispersion as is practicable, or equivalently with as large a value of N as is practicable. The search in the sets $I^s \cap C_i$, $i \geqq 1$, may then be carried out with a smaller number of points. Another possibility is to prevent the cubes C_i from contracting so fast that $x^* \notin C_i$ for some i. This can be done by adjusting the size of the cubes after a suitable number of steps. For instance, we may use $\varepsilon_i = \varepsilon^i$ for $1 \leqq i \leqq k$ with some $0 < \varepsilon < \frac{1}{2}$, then put $\varepsilon_{k+1} = \delta\varepsilon^k$ for some $\delta > 1$ and continue with $\varepsilon_i = \delta\varepsilon^{i-1}$ for $k+2 \leqq i \leqq 2k$, then put $\varepsilon_{2k+1} = \delta^2\varepsilon^{2k-1}$, and so on. We call this procedure "outer iteration".

3. The choice of sample points. Suitable point sets for quasi-random search methods have been considered by Aird and Rice [1], Niederreiter and McCurley [12], Sobol' [16], and in the extensive numerical experiments carried out by Niederreiter and Peart [13]. Theoretical results on the dispersion of point sets can be found in Niederreiter [9]-[11], Peart [14], and Sobol' [17]. For a general background on quasi-random point sets we refer to Niederreiter [8].

If the number N of points in I^s that we want to use is prescribed, then we can work with a set of N points in I^s of minimal dispersion which is constructed as follows. Let r be the unique positive integer with $r^s \leqq N < (r+1)^s$, consider the r^s points $(h_1/2r, \cdots, h_s/2r)$ with the h_j running independently through the set $\{1, 3, 5, \cdots, 2r-1\}$, and add arbitrary points from I^s to get a total of N points. Then $d_N = (2r)^{-1}$, and this is the smallest possible value of the dispersion of any N points in I^s by [11, Thm. 1].

For many practical purposes it is preferable to work with low-dispersion point sets $x_1, \cdots, x_N$ possessing the additional property that initial segments $x_1, \cdots, x_p$ with $p < N$ also have relatively small dispersion. In this way the intermediate results of calculations already yield reasonably good approximations. Furthermore, we then have the flexibility of first running the calculation with $p < N$ sample points and then just adding sample points if more accuracy is desired, thus being able to use the results of the earlier calculation. Point sets that are particularly suitable are those arising from good lattice points and from linear congruential pseudo-random numbers; see [8] for

detailed information on these point sets. It was noted in [13] that these point sets are easy to compute and possess a small dispersion.

Let $x_1, \cdots, x_m$ with $x_n = \{ng/m\}$ be a point set in I^s arising from the good lattice point $g = (g_1, \cdots, g_s) \in \mathbb{Z}^s$ with modulus m. Here $\{\cdot\}$ denotes reduction mod 1 in each coordinate. We consider now the point set $\hat{x}_1, \cdots, \hat{x}_m$ with $\hat{x}_n = \{hng/m\}$ and $h = \lfloor m(\sqrt{5}-1)/2 \rfloor$, where $\lfloor t \rfloor$ denotes the greatest integer $\leqq t$. If h and m are relatively prime, then the point set $\hat{x}_1, \cdots, \hat{x}_m$ is just a rearrangement of $x_1, \cdots, x_m$ and thus has the same dispersion. In published tables of good lattice points (see, for example, Hua and Wang [5] and Korobov [7]) the lattice points g are always listed in normalized form, i.e. with $g_1 = 1$. In this case any segment $x_1, \cdots, x_p$ with, say, $p \leqq 3m/4$ exhibits very poor distribution properties since none of the points has a first coordinate $> \frac{3}{4}$. The effect of introducing the integer h to obtain the point set $\hat{x}_1, \cdots, \hat{x}_m$ is to produce segments $\hat{x}_1, \cdots, \hat{x}_p$ with better distribution properties.

Turning now to point sets arising from linear congruential pseudo-random numbers, we know from [8] that for a suitable multiplier λ the point set $x_1, \cdots, x_\tau$ with

$$x_n = \left\{ \left(\lambda^{n-1} \frac{y_0}{m}, \lambda^n \frac{y_0}{m}, \cdots, \lambda^{n+s-2} \frac{y_0}{m} \right) \right\}$$

is evenly distributed over I^s, where m is the modulus, y_0 is relatively prime to m, and τ is the least positive integer with $\lambda^\tau \equiv 1 \bmod m$. Moreover, the initial segments $x_1, \cdots, x_p$ with $p \leqq \tau$ and p somewhat larger than $m^{1/2}$ are also evenly distributed over I^s. Concretely, the inequality (2) in [10] and the results in [8, § 11] yield effective upper bounds for the dispersion of such segments. Suitable values of λ can be obtained from the tables in Borosh and Niederreiter [2] and Knuth [6, Chap. 3].

There exist infinite sequences such as Halton sequences (see [8]) or Faure sequences (see [4]) with the property that any sufficiently long initial segment is evenly distributed over I^s and has in particular a small dispersion. Both types of sequences have been used successfully in quasi-random search methods (see [12], [13]). A rather different situation arises with respect to another well-known set of quasi-random points. Consider the Hammersley points

$$x(n) = \left(\frac{n}{N}, \phi_{g_1}(n), \cdots, \phi_{g_{s-1}}(n) \right) \in I^s, \qquad n = 0, 1, \cdots, N-1,$$

with ϕ_{g_i} being the radical-inverse function defined in [8, p. 973], and $g_1, \cdots, g_{s-1}$ being pairwise relatively prime integers $\geqq 2$. If all N points of this point set are taken, then they show a very good distribution behavior. However, if we only take the first $p < N$ points, then none of the points has a first coordinate $> (p-1)/N$. To overcome this deficiency, we may use a similar rearrangement as for point sets arising from good lattice points, or we may work with a pseudo-random rearrangement, for example, using the recursion $y_{n+1} \equiv \lambda y_n + r \pmod N$, $n = 0, 1, \cdots$, with integers $0 \leqq y_n < N$ and suitable choices of λ, y_0, and r (see [8, pp. 1000–1001]), and ordering the Hammersley points accordingly. That is, we take the points in the order $x(y_0), x(y_1), \cdots, x(y_{N-1})$.

The sets of N points in I^2 arising from optimal coverings by circles (see [13]) present a similar problem as the Hammersley point sets, in that an initial segment of $p < N$ points is likely to provide a poor covering of I^2. Here the rearrangement trick does not work so well since there is no canonical ordering of the points. Therefore, this point set should only be used if N is prescribed.

The property we considered above, namely that initial segments $x_1, \cdots, x_p$ of the low-dispersion point set $x_1, \cdots, x_N$ also have small dispersion, manifests itself in the

relatively small average number of "change points" encountered in numerical calculations with such point sets. By a "change point" of the point set $x_1, \cdots, x_N$ we mean a point x_n for which $f(x_n) > f(x_p)$ for all $p < n$, where f is the function being maximized. A smaller number of change points leads to a shorter running time of the search method.

We have carried out extensive numerical experiments comparing the crude search method with the method of localization of search. The improvement obtained by the latter method is dramatic. We have also compared the effectiveness of the various point sets described above in the method of localization of search. The conclusions we reached are similar to those in [13], namely that in terms of the total running time the point sets arising from good lattice points and from linear congruential pseudo-random numbers perform best. We have experimented with several combinations of the important parameters ε (= contraction factor from the cube C_i to C_{i+1}) and N (= number of points used to search each cube). For the dimensions $s = 2, 3, 4$ we have considered, the choice $\varepsilon = 0.2$ and $N = 4 \cdot 5^s$ performed satisfactorily for most functions. The procedure of outer iteration (see § 2) improved the performance in the case of strongly oscillating functions. Detailed results of the numerical experiments are available from the authors on request.

REFERENCES

[1] T. J. AIRD AND J. R. RICE, *Systematic search in high dimensional sets*, SIAM J. Numer. Anal., 14 (1977), pp. 296–312.

[2] I. BOROSH AND H. NIEDERREITER, *Optimal multipliers for pseudo-random number generation by the linear congruential method*, BIT, 23 (1983), pp. 65–74.

[3] L. P. DEVROYE, *Progressive global random search of continuous functions*, Math. Programming, 15 (1978), pp. 330–342.

[4] H. FAURE, *Discrépance de suites associées à un système de numération (en dimension s)*, Acta Arith., 41 (1982), pp. 337–351.

[5] L. K. HUA AND Y. WANG, *Applications of Number Theory to Numerical Analysis*, Springer-Verlag, New York, 1981.

[6] D. E. KNUTH, *The Art of Computer Programming, Vol. 2: Seminumerical Algorithms*, 2nd ed., Addison-Wesley, Reading, MA, 1981.

[7] N. M. KOROBOV, *Number-Theoretic Methods in Approximate Analysis*, Fizmatgiz, Moscow, 1963. (In Russian.)

[8] H. NIEDERREITER, *Quasi-Monte Carlo methods and pseudo-random numbers*, Bull. Amer. Math. Soc., 84 (1978), pp. 957–1041.

[9] ———, *A quasi-Monte Carlo method for the approximate computation of the extreme values of a function*, in Studies in Pure Mathematics (To the Memory of Paul Turán), Birkhäuser Verlag, Basel, 1983, pp. 523–529. (Submitted in 1977.)

[10] ———, *On a measure of denseness for sequences*, in Topics in Classical Number Theory (Budapest, 1981), North-Holland, Amsterdam, 1984, pp. 1163–1208.

[11] ———, *Quasi-Monte Carlo methods for global optimization*, Proc. Fourth Pannonian Symposium on Mathematical Statistics, Bad Tatzmannsdorf, 1983, Reidel, Dordrecht-Boston, to appear.

[12] H. NIEDERREITER AND K. MCCURLEY, *Optimization of functions by quasirandom search methods*, Computing, 22 (1979), pp. 119–123.

[13] H. NIEDERREITER AND P. PEART, *A comparative study of quasi-Monte Carlo methods for optimization of functions of several variables*, Caribbean J. Math., 1 (1982), pp. 27–44.

[14] P. PEART, *The dispersion of the Hammersley sequence in the unit square*, Monatsh. Math., 94 (1982), pp. 249–261.

[15] R. Y. RUBINSTEIN, *Simulation and the Monte Carlo Method*, Wiley, New York, 1981.

[16] I. M. SOBOL', *On the systematic search in a hypercube*, SIAM J. Numer. Anal., 16 (1979), pp. 790–793.

[17] ———, *On an estimate of the accuracy of a simple multidimensional search*, Dokl. Akad. Nauk SSSR, 266 (1982), pp. 569–572. (In Russian.)

[18] F. J. SOLIS AND R. J.-B. WETS, *Minimization by random search techniques*, Math. Oper. Res., 6 (1981), pp. 19–30.

SIAM J. SCI. STAT. COMPUT.
Vol. 7, No. 2, April 1986

022

PERIODIC SMOOTHING OF TIME SERIES*

JOHN ALAN McDONALD†

Abstract. To discover and summarize *regular periodic variation* in a time series, $\{y_i, t_i\}$, we may consider approximating the observed y_i with a general smooth periodic function of t_i:

$$y_i \approx f(\omega t_i),$$

where $f(\cdot)$ is a periodic function with period 1 (so that $f(\omega t)$ has period $\lambda = 1/\omega$ as a function of t). To make such an approximation, we need to choose a function $f(\cdot)$; the frequency, ω, may be known, or it may need to be chosen also. The basic suggestion presented in this paper is to choose the function, $f(\cdot)$, by smoothing y_i as a function of $(\omega t_i \bmod 1)$. If ω is unknown, it can be chosen to make $f(\omega t_i)$ a good approximation to y_i.

More complex periodic variation can be modelled using a variation of the *Projection Pursuit Paradigm* (Friedman and Stuetzle (1982a)) to construct approximations of the form

$$y_i \approx \sum_{k=1}^{K} f_k(\omega_k t_i).$$

This approach has been developed as part of the Orion project for statistical graphics at the Stanford Linear Accelerator Center; it is most useful in an interactive graphics environment, like the Orion I workstation (Friedman and Stuetzle (1981b), McDonald (1982a, b, c)). However, given moderate compromises, it can be used with profit in more conventional environments for statistical computing.

Some advantages of periodic smoothing are: The observed times, t_i, need not be equally spaced, which means, in particular, that missing data is not a problem. The function, $f(\cdot)$, and the frequency, ω, can be chosen in a way that is insensitive to occasional gross errors in y_i (and t_i, for that matter). The function, $f(\cdot)$, need not be easily approximated by simple linear combinations of sines and cosines at harmonics of the fundamental frequency, ω. Perhaps most importantly, the model can be constructed and modified interactively, and has a natural graphical representation.

Key words. data analysis, seasonal adjustment, harmonic analysis, Fourier analysis of time series, Lynx series, projection pursuit

1. Introduction.

This paper discusses methods which are appropriate as first steps in the analysis of time series in which we suspect there may be *periodic* variation. These methods are intended primarily for *descriptive* and *exploratory* purposes. The approach was developed as part of the Orion project, whose goal is to develop new graphical and computational methods for data analysis. In particular, these methods were designed for use on powerful, interactive computer graphics workstations, like the Orion I workstation (Friedman and Stuetzle (1981b), McDonald (1982a, b, c)).

Suppose we observe a time series, $\{y_i, t_i\}$. (A real-valued variable, y_i, is observed at times t_i.) To discover and summarize periodic variation in the series we may construct a decomposition of the observed series into *model* plus *residuals* (*signal* plus *noise*):

$$y_i = f(\omega t_i) + r_i \tag{1}$$

where $f(\cdot)$ is a periodic function with period 1 (so that $f(\omega t_i)$ has period $\lambda = 1/\omega$ as a function of t_i).

* Received by the editors March 11, 1985. This research was supported in part by a National Science Foundation Mathematical Sciences Postdoctoral Research Fellowship, the U.S. Department of Energy under contracts DE-AC03-76SF00515 and DE-AT03-81-ER10843, the Office of Naval Research under contract ONR N00014-81-K-0340, and the U.S. Army Research Office under contract DAAG29-82-K-0056.

† Stanford Linear Accelerator Center and Department of Statistics, Stanford University, Stanford, California 94305.

The purpose of descriptive modeling is to summarize some aspect of *structure* in data. In this case, structure means periodic variation. We discover, understand, and interpret structure by examining the model. So a descriptive model is especially valuable if it has a natural graphic representation. Also, we can use the model to remove structure which might otherwise hinder further analysis.

To be more specific, we may be interested in decompositions like (1) in order to:

A. understand the shape of the periodic variation (see the functions, $f(\cdot)$);
B. find interesting, possibly hidden, frequencies, ω;
C. produce an adjusted series, $\{r_i, t_i\}$ that can be reasonably modeled as a stationary process with continuous spectral measure (e.g. ARMA models);
D. forecast the series: for future time, t, predict $y(t) = f(\omega t)$.

The methods discussed here are primarily intended for (A) and (B); we emphasize the ability to discover structure that might not otherwise be seen.

2. Periodic smoothing. The *periodic* dependence of y_i on t_i at frequency ω is equivalent to the dependence of y_i on the circular variable (ωt_i mod 1). So a scatterplot of y_i versus (ωt_i mod 1) is a natural picture of the periodic dependence of y_i on t_i at ω.

In general, to summarize general (nonlinear) dependence in a scatterplot, we use a *scatterplot smoother* (Friedman and Stuetzle (1982b)). I refer to smooths generically as $\mathcal{SM}\{(y_i, x_i)\}$, which is read as "*the smooth of* y_i *as a function of* x_i." The smoother, $\mathcal{SM}$, operates on a set of ordered pairs, $\{(y_i, x_i)\}$ to produce a set of smoothed values, $\{\hat{y}_i\}$.

So, in this case, to choose a general smooth periodic function, $f(\cdot)$, *smooth* y_i as a function of the circular variable (ωt_i mod 1). The function, $f(\cdot)$, is determined at the observed ωt_i by:

$$\{f(\omega t_i)\} = \mathcal{SM}\{(y_i, \omega t_i \bmod 1)\}.$$

For other values of t, $f(\omega t \bmod 1)$ can be determined by interpolating between observed values of (ωt_i mod 1).

2.1. The smoothing algorithm. The particular smoothing algorithm used is a detail, but an important one. The examples in this paper use $\mathcal{SM}_{LRC}$, which is a modified version of the supersmoother, $\mathcal{SM}_{SUP}$, of Friedman and Stuetzle (1982b). $\mathcal{SM}_{LRC}$ is discussed in detail in McDonald and Owen (1983); some important features are as follows.

Running linear fits. $\mathcal{SM}_{SUP}$ is based on *running linear fits* (Cleveland (1979); Friedman and Stuetzle (1982b)), which are similar to *running averages*, $\mathcal{SM}_{AVE}$.

To produce the smoothed value, $\hat{y}_i$, for a point, (y_i, x_i), a running average fits a constant (typically the mean) to the set of y_j's whose corresponding x_j's are in a neighborhood of x_i. A running linear fit fits a straight line, $y = a + b \cdot x$, to the points (y_j, x_j) whose x_j's are in a neighborhood of x_i. The smoothed value, $\hat{y}_i$, is the value of the line at x_i, that is, $\hat{y}_i = a + b \cdot x_i$.

Running linear fits perform better than running averages when the x_i's are irregularly spaced. Running linear fits also do better at extrapolating to the end points of the data range (which does not matter here, because (ωt_i mod 1) is a circular variable).

Split linear fits. The major difference between $\mathcal{SM}_{LRC}$ and $\mathcal{SM}_{SUP}$ is the use of *split linear fits.*

We would like a smoothing algorithm to be able to fit functions that are not *smooth* in the usual sense of having everywhere defined, small second derivatives. In other words, we would like a smoother that does reasonably well at reproducing isolated cusps and points of discontinuity. This is a generally desirable property for smoothers

and is particularly important here—because we want to be able to discover periodic functions that are not well approximated by short Fourier series. If we know that the underlying function is everywhere smooth, in the sense of having small second derivatives, then a *low pass filter* is a reasonable choice for a smoothing algorithm. Using a low pass filter is equivalent to Fourier methods discussed below (Bloomfield (1976), Damsleth and Spjøtvoll (1982)). Periodic smoothing is more expensive than Fourier methods; it is therefore only justified if it solves problems that Fourier methods do not.

In running linear fits (and running averages), we smooth the data by making a guess, $\hat{y}_i$, for what y_i should have been, based on the values of the y_j's whose x_j's are near x_i. $\mathcal{SM}_{SUP}$ uses a symmetric, *central* window for the neighborhood of x_i. The number of observations in the window is called the *span*, s. A central window contains $s/2$ observations to the left of x_i and $s/2$ observations to the right.

If we suspect that there may be a jump or sharp cusp in the signal near x_i, then we should consider making the guess based on an *asymmetric* neighborhood of x_i. The idea is to choose a neighborhood in which the underlying function is better approximated by a line.

To produce the smoothed value, $\hat{y}_i$, for (y_i, x_i), $\mathcal{SM}_{LRC}$ makes three guesses, $\hat{y}_i(L)$, $\hat{y}_i(R)$, and $\hat{y}_i(C)$. $\hat{y}_i(C)$ is derived from a central window, as in $\mathcal{SM}_{SUP}$, $\hat{y}_i(L)$ is found by fitting a straight line to a window to the left of x_i; and $\hat{y}_i(R)$ is derived from a window to the right. This is the *split linear fit* (McDonald and Owen (1983)). How the three guesses are combined to produce the smoothed value, $\hat{y}_i$, is discussed in the next section.

Split linear fits are closely related to *edge preserving* smoothing algorithms invented independently in the image processing literature (Tomita and Tsuji (1977), Nagao and Matsuyama (1977), Haralick and Watson (1979), Scher, Velasco, and Rosenfeld (1980)).

Combining guesses through cross-validation. A feature that $\mathcal{SM}_{LRC}$ inherits from $\mathcal{SM}_{SUP}$ is the use of *local cross-validation* combine a number of guesses for $\hat{y}_i$.

$\mathcal{SM}_{SUP}$ uses cross-validation to choose the span (the size of the window) of the running linear fit. The span determines the degree of smoothing done; it is chosen to adapt to varying signal curvature and noise amplitude.

$\mathcal{SM}_{SUP}$ computes running linear fits, $\hat{y}_i(s)$, for a number (typically 5) of values of the span, s. Each of these guesses is cross-validated, that is, the point, (y_i, x_i), does not enter into the computation of $\hat{y}_i(s)$. The $\mathcal{SM}_{SUP}$ chooses the value of s that minimizes the squared cross-validated residual: $r_i^2(s) = (y_i - \hat{y}_i(s))^2$. To give more stable results, $\mathcal{SM}_{SUP}$ smooths $r_i^2(s)$ for each s (with a running average) before choosing $\hat{y}_i$.

$\mathcal{SM}_{LRC}$ computes $\hat{y}_i$ as a weighted average of cross-validated guesses, $\hat{y}_i(s, L)$, $\hat{y}_i(s, R)$, and $\hat{y}_i(s, C)$, for several values of span and for left, right, and central windows. A weighted average seems to give more stable results, on simulated data, than simply choosing the best guess. The weights used in the examples in this paper are an exponentially decaying function of the absolute cross-validated residuals; more details are in McDonald and Owen (1983).

Circular abscissa. $\mathcal{SM}_{LRC}$ is slightly simplified by the fact that $(\omega t_i \bmod 1)$ is a circular variable; wrapping windows around the end points of the interval is, in fact, easier than dealing with end effects.

Rejection rule for outliers. To make $\mathcal{SM}_{LRC}$ resistant to occasional aberrant observations, the first pass is an outlier rejection rule (Friedman and Stuetzle (1982b)), based on residuals from a running median smooth (Mosteller and Tukey (1977)). The rejection rule is inherited unchanged from $\mathcal{SM}_{SUP}$.

2.2. Example: the LYNX data. A classic time series that shows regular periodic oscillations is the MacKenzie River Lynx data. The raw series consists of the annual catches of lynx furs for the Hudson Bay Company in the MacKenzie river district in northwest Canada from 1821 to 1934. The data was compiled by Elton and Nicholson (1942). This series has received considerable attention from population biologists and statisticians (for example: Bulmer (1974), Cambell and Walker (1977), Damsleth and Spjøtvoll (1982), May (1974)). The series is of particular interest because similar, though less well documented, oscillations are reported in many other species of mammals in Canada and northern Eurasia (Bulmer (1974)). Many analyses have focused on predator-prey interactions of the lynx and the snowshow hare, another species with an oscillating population (for example, see Fig. 4.4 in May (1974)).

Figure 1 is a plot of log (# furs) over time. The period of the oscillation is about 9.5 years. Figure 2 is a plot of y_i versus (t_i mod 9.5). We plot y_i versus (t_i mod λ), rather than y_i versus (ωt_i mod 1), so that the horizontal scale is in years rather than the fractions of a cycle. The data is plotted twice, to show two cycles of the periodic function. This makes it easier to see the shape of the periodic variation, without arbitrary breaks at the sides of the scatterplot.

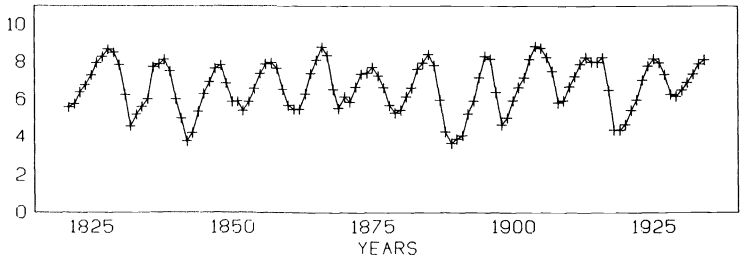


FIG. 1. *Mackenzie River lynx data, original series.* Log (*counts*) *of lynx furs trapped in the MacKenzie river district for the years* 1821-1934.

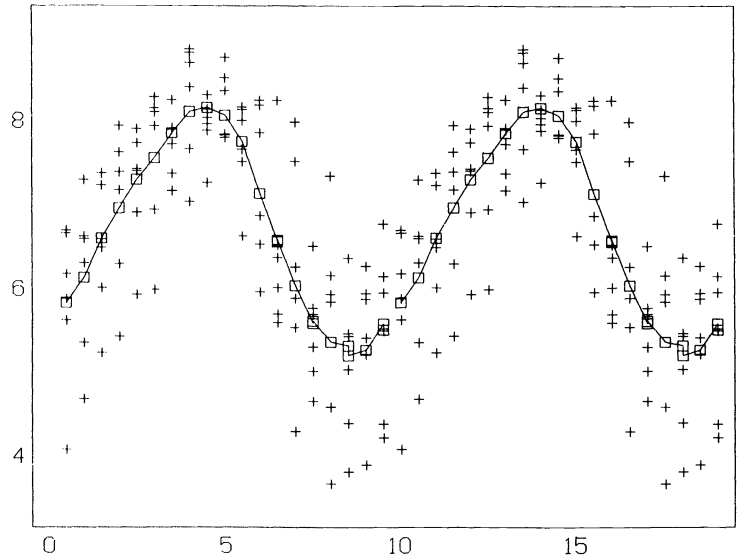


FIG. 2. *MacKenzie River lynx data.* Log (*counts*) *vs. t* mod 9.5 *years. Pluses = original data; squares = smooth.*

In all the figures, observed data is plotted as crosses; smoothed values are plotted as squares; observations flagged as outliers by the smoother are represented by stars; residuals are plotted as diamonds.

In Fig. 2, the shape of the smooth is not far from sinusoidal; it is slightly skewed, with a slow rise and a more rapid fall. The apparent skewness is interesting in the context of predator-prey models; oscillatory solutions to a simple two-species Lotka-Volterra model (May (1974, Fig. 3.2)) show similar skewness. This is an example of the sort of qualitative feature of the periodic function that is easy to detect with periodic smoothing and would be more difficult to see in a Fourier decomposition or a fitted ARMA model.

Unfortunately, the Lotka-Volterra model is not a reasonable one for the lynx-hare series, because its oscillatory solutions are unstable under small perturbations. Also, because similar oscillations occur in many other species, a two species predator-prey model seems unlikely to be a completely adequate explanation.

3. Choosing a frequency. Consider again the decomposition (1), but now suppose ω is unknown. I suggest two alternatives for choosing ω: manually, with an interactive graphics system, or automatically, following an explicit algorithm.

3.1. Interactive search. On a graphics system like the Orion I workstation (Friedman and Stuetzle (1981b), McDonald (1982)), we can discover interesting values of ω interactively. The system provides an input device (a trackerball) that can vary the value of ω continuously. At each moment the display shows a scatterplot of y_i versus ($\omega t_i \bmod 1$) for the current value of ω. The scatterplot is enhanced with the curve of the smooth, $\mathcal{SM}_{LRC}\{(y_i, \omega t_i \bmod 1)\}$. On *Orion* I, it is possible to update the picture quickly enough (2-10 times per second) for a user to manually search frequency space for interesting values of ω.[1]

3.2. Automatic search. A more objective approach would be to choose ω to maximize some numerical criterion of fit, $\mathcal{F}(\omega)$. $\mathcal{F}(\omega)$ plays the role of the power spectrum (Bloomfield (1976)) in Fourier analysis, so it is referred to here as the *pseudo-spectrum.*

A well chosen pseudo-spectrum is also useful for interactive search; it identifies potentially interesting frequencies, thereby greatly reducing the time spent searching over ω.

A natural choice for the pseudo-spectrum, by analogy with the periodogram, is:

$$\mathcal{F}(\omega) = R^2(\omega),$$

where

$$R^2(\omega) = 1 - \frac{\sum_i (y_i - \hat{y}_i(\omega))^2}{\mathcal{VAR}(y_i)}$$

and

$$\{\hat{y}_i(\omega)\} = \mathcal{SM}_{LRC}\{(y_i, \omega t_i \bmod 1)\}.$$

Unfortunately, this simple analogy does not work. Roughly speaking, $\mathcal{SM}_{LRC}$ projects the observed y_i onto the space of smooth functions that are periodic with frequency ω. Difficulties arise here that do not arise with the periodogram because the

[1] The use of an interactive graphics system in this way to look at periodic variation in time series has been suggested independently by Juan Carlos Lerman (1982).

spaces of general smooth periodic functions at different frequencies are not mutually orthogonal in the way that sines and cosines are.

Figure 3 shows a plot of $R^2(\omega)$ versus ω for the Lynx data. We see a peak at what should be the dominant frequency: $\omega_0 \approx 0.105 = 1.0/9.5$. We also see many other peaks. The peaks to the left of 0.105 are at the frequencies: $\omega_0/2$, $\omega_0/3$, $\omega_0/4$, etc. These frequencies correspond approximately to periods of $2 \times 9.5 = 19.0$, $3 \times 9.5 = 28.5$, and so on.

Suppose we have data that can well be approximated by a function whose *true* frequency is ω_0. Frequencies of the form ω_0/m, m an integer, are called the *subharmonics* of ω_0. If we look at a scatterplot of y_i vs $(\omega_0/m) \cdot t_i \bmod 1$, we will see m cycles of the periodic function. For m not too large, the smoother will fit m cycles of the data as well as one, so $R^2(\omega_0/m) \simeq R^2(\omega_0)$.

Figure 4 is a plot of y_i versus $(t_i \bmod 29)$ which corresponds to highest peak in Fig. 3. This plot seems to show three cycles of a periodic function (we see six cycles in the plot because the data is repeated twice). That is what we expect, because 29 years is three times the fundamental period of about 9.5 years.

Figure 3 also has many peaks at values of $\omega \gg 0.105$. These peaks correspond to subharmonics of *aliases* (see Bloomfield (1976)) of true frequency ω_0. Suppose ω_a is an alias of ω_0. Then a plot of y_i versus $((\omega_a/m) \cdot t_i \bmod 1)$ will also show m cycles of the underlying periodic function.

To eliminate spurious peaks in the pseudo-spectrum, we need to somehow penalize $\mathcal{F}(\omega)$ for excess cycles in the smooth. One way to do this is to penalize for the *curvature* of $\mathcal{SM}_{LRC}\{(y_i, \omega_i \bmod 1)\}$. If ω_0 is the correct frequency, $\mathcal{SM}_{LRC}\{(y_i, (\omega_0/m)t_i \bmod 1)\}$ will have m times the curvature of $\mathcal{SM}_{LRC}\{(y_i, \omega_0 t_i \bmod 1)\}$.

Experiments with simulated data suggest the following pseudo-spectrum:

$$\mathcal{F}(\omega) = \frac{R^2(\omega)}{\mathcal{C}(\omega)},$$

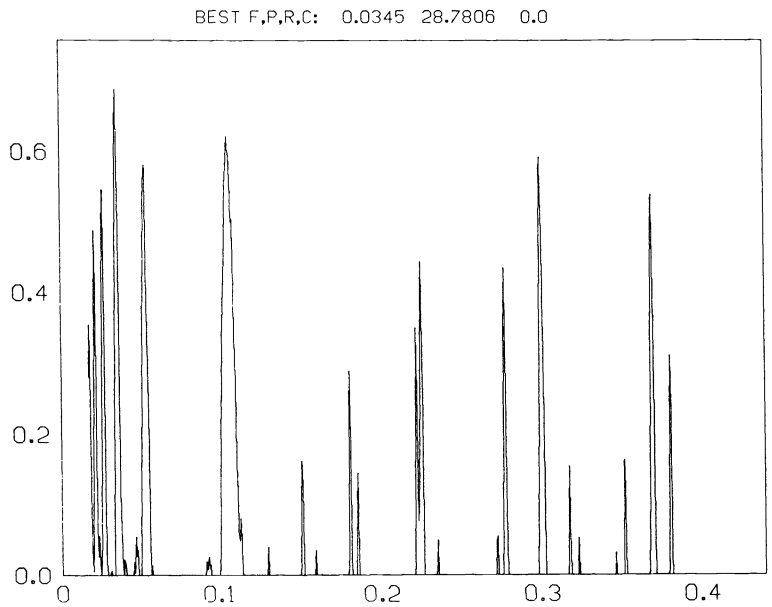


FIG. 3. *MacKenzie River lynx data.* $R^2(\omega)$ vs. ω.

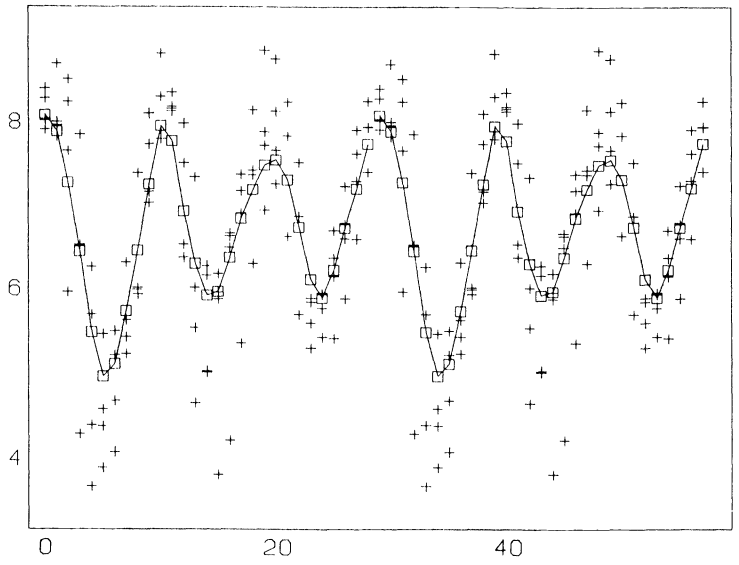


FIG. 4. *MacKenzie River lynx data.* Log (*counts*) *vs.* t mod 29.0 *years. Pluses = original data; squares = smooth.*

where $\mathscr{C}(\omega)$ is a measure of curvature. Curvature is measured by an integrated absolute second derivative. To be more precise:

$$\mathscr{C}(\omega) = \frac{1}{N}\sum_{i=1}^{N} |\mathscr{SM}_{AVE}\{(\hat{y}_i, \omega t_i \bmod 1), s\} - \mathscr{SM}_{AVE}\{(\hat{y}_i, \omega t_i \bmod 1), 3s\}|,$$

where $\mathscr{SM}_{AVE}\{(\cdot, \cdot), s\}$ is a running average of span s. The difference of a running average of span s and a running average of span $3s$ is equivalent to a smoothed second difference filter.

Figure 5 shows a plot of $\mathscr{F}(\omega)$ versus ω. The dominant frequency in this plot, 0.104, corresponds to a period of 9.64 years. Figure 6 shows y_i (crosses) and $\hat{y}_i$ (squares) versus (t_i mod 9.64).

Figure 5 shows that $\mathscr{F}(\omega)$ is still a function with many local maxima. So usual methods for numerical optimization may have difficulty finding the *global* maximum. To be sure of finding the global maximum we can either do an exhaustive search over values of ω or provide a numerical optimizer with a good set of starting values.

Exhaustive search is simple in concept and very reliable. A search with sufficient precision may, however, be too expensive for most statistical computing environments at present. On the other hand, exhaustive search is a perfectly reasonable option on a dedicated graphics system like Orion I. Furthermore, we expect that similar, inexpensive, powerful workstations will be the standard environment for scientific computing within five to ten years.

On a graphics workstation, we need not be restricted to a single choice for $\mathscr{F}(\omega)$. We can display on the screen both $R^2(\omega)$, $\mathscr{C}(\omega)$, and $\mathscr{F}(\omega, \alpha)$, where $\mathscr{F}(\omega, \alpha)$ is something like $R^2(\omega)/\mathscr{C}(\omega)^\alpha$. The value of α can be controlled by an input device such as a trackerball, and the plot of $\mathscr{F}(\omega, \alpha)$ can be recomputed and redrawn in real-time as α is changed.

For the immediate future, an effective and inexpensive alternative to exhaustive search (when t_i are equi-spaced) is to take the 10–20 largest values from the periodogram

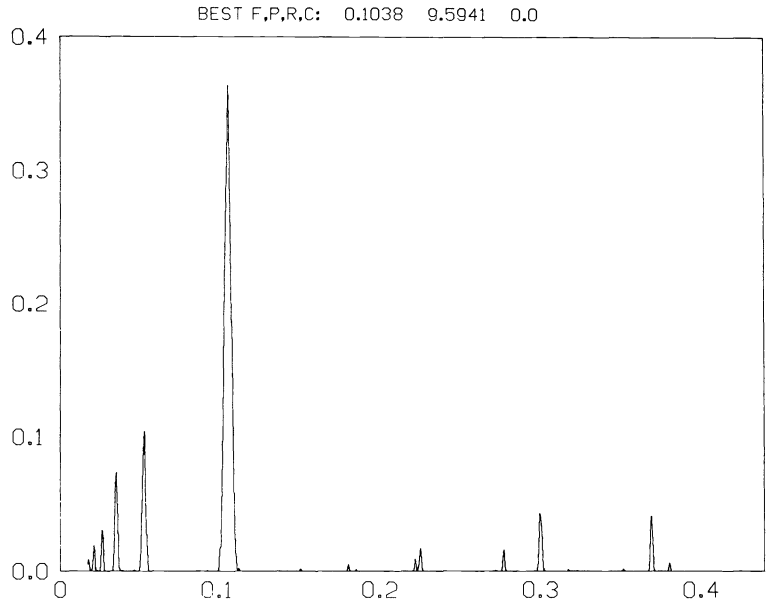


FIG. 5. *MacKenzie River lynx data.* $\mathscr{F}(\omega) = R^2(\omega)/\mathscr{C}(\omega)$ *vs.* ω. *Maximum corresponds to a period of* 9.64 *years.*

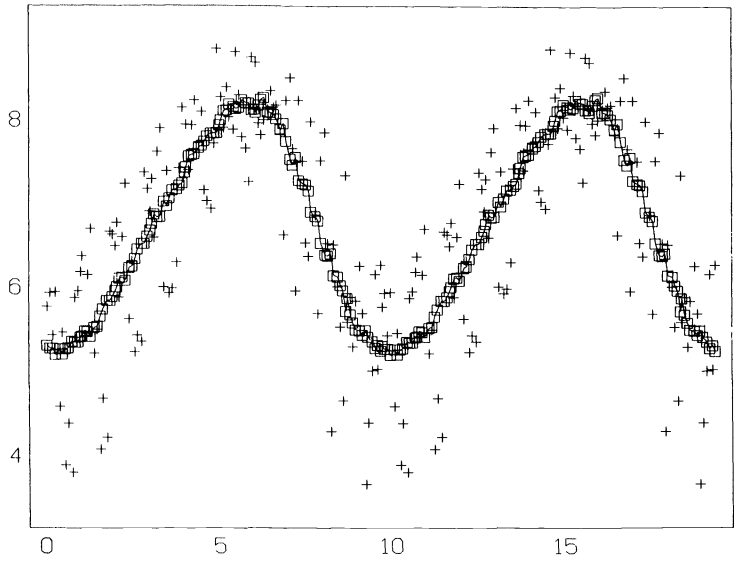


FIG. 6. *MacKenzie River lynx data.* Log (*counts*) *vs. t* mod 9.64 *years. Pluses = original data; squares = smooth.*

(or some other estimate of the power spectrum) as starting values for a local numerical optimization.

4. Comparison with Fourier analysis. A standard, and often successful, approach to constructing decompositions like (1) is based on Fourier analysis (Bloomfield (1976), Damsleth and Spjøtvoll (1982)). To find an unknown frequency, ω, one looks for a peak in an estimate of the power spectrum of the series. The model, $f(\cdot)$, is fit by a

Fourier series at harmonics of the fundamental frequency, ω. That is:

$$f(\omega t) = \sum_{j=1}^{J} \alpha_j \cdot \sin(2\pi j\omega t + \phi_j). \tag{2}$$

(The amplitudes, a_j and the phases, ϕ_j, are typically estimated by least squares.)

The major advantage of Fourier methods over periodic smoothing is computational speed. Also, the power spectrum does not have the problems in the pseudo-spectrum that arise from lack of orthogonality.

The major advantages of periodic smoothing are:

- A smoother can approximate simple functional forms that are difficult to approximate well by a Fourier series of reasonable length.
- A plot of the Fourier approximation may be misleading, due to eye-catching peculiarities related to Gibb's phenomena. An inappropriately chosen smoothing algorithm may also show Gibb's phenomena, which arise from trying to fit a *smooth* function (with continuous second derivatives) to *nonsmooth* data. $\mathcal{SM}_{\mathrm{LRC}}$ was designed to deal with nonsmooth data and does not show Gibb's phenomena.
- Smoothing y_i as a function of (ωt_i mod 1) is not sensitive to whether the t_i are equi-spaced, that is, whether $t_i = t_0 + i \cdot \Delta t$. Most estimates of the power spectrum require t_i to be equi-spaced. In particular, this implies that Fourier methods have difficulty with missing data.
- Most estimates of the power spectrum are sensitive to the effects of a few aberrant observations; with a resistant smoothing algorithm the pseudo-spectrum need not be.
- If the amplitudes and phases are fitted by least squares, then the model, $f(\cdot)$, will also be sensitive to aberrant observations; a periodic smoothing model need not be.

5. Comparison with seasonal adjustment. Periodic smoothing may be viewed as a generalization of additive seasonal adjustment.

If the observed times, t_i, are equi-spaced ($t_i = t_0 + i\Delta t$), and the period, $\lambda = 1/\omega$, is an integer multiple of the spacing ($\lambda = l\Delta t$), then the ωt_i mod 1 will take on only l discrete values. This is typical of problems in seasonal adjustment; the season is usually of integer length (for example, 12 months). In this case, it makes sense to simply average over each of the l values of ωt_i mod 1.

Choosing ω to maximize a numerical criterion, $\mathcal{F}(\omega)$, is a generalization of pre-Fourier methods for uncovering hidden periodicity in time series. Buys-Ballot (Buys-Ballot (1847), Whittaker and Robinson 1924)) suggested essentially the same procedure, for equi-spaced data, restricting consideration to those ω's that correspond to integer length periods, averaging as in seasonal adjustment rather than smoothing, and using R^2 as the measure of fit. In other words, we try seasonal adjustment for all integer length seasons and take the season that fits best.

The use of a smoothing algorithm makes it possible to consider noninteger length periods and series with irregularly spaced times or missing observations.

6. Several frequencies. More complicated periodic variation in time series can be described with a decomposition involving several frequencies:

$$y_i = \sum_{k=1}^{K} f_k(\omega_k t_i) + r_i^K. \tag{3}$$

We construct such a decomposition using a variation of the *Projection Pursuit Paradigm* (Friedman and Stuetzle (1982a)).

The decomposition is constructed sequentially, adding terms until no improvement in fit is possible. We choose the Kth periodic function, $f_K(\cdot)$, by smoothing the residuals, r_i^{K-1}, from the first $K-1$ terms:

$$f_K(\omega_K t_i) = \mathcal{SM}_{LRC}\{(r_i^{K-1}, \omega_K t_i \bmod 1)\}.$$

The Kth frequency, ω_K, is chosen to maximize the fit, $\mathcal{F}(\omega_K)$, of $f_K(\omega_K t_i)$ to r_i^{K-1}.

Given a K term model, we can sometimes improve the fit by *backfitting*, a technique of cyclical optimization used in Projection Pursuit algorithms (Friedman and Stuetzle (1982a)). The basic idea is to refit each term, ω_k and $f_k(\cdot)$, to adjust for variation in the residuals from the other $K-1$ terms.

In cases where the frequencies, ω_k, are known beforehand, the *Projection Pursuit Paradigm* reduces to just the backfitting step.

We can adjust the series for a long term, nonperiodic *trend* with only a minor modification of the *Projection Pursuit Paradigm.* We merely need to include a term, $f_0(t_i) = \mathcal{SM}_{LRC}\{(y_i, t_i)\}$. ($t$ is not treated as a circular variable in the trend term.)

7. Example: the variable star data. Figure 7 is a plot of the so-called "Variable Star" data of Whittaker and Robinson (1924). This data set purports to be observations of the brightness of a variable star over 600 days.

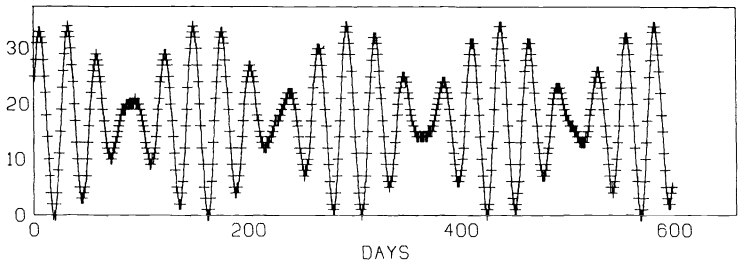


FIG. 7. *Variable star data: original series. The periodogram of the series (Bloomfield, 1976) suggests a two term model with periods 24 and 29 days.*

This data set was analyzed in detail by Bloomfield (1976), using Fourier methods. The periodogram of the data shows peaks at two fundamental frequencies and at their harmonics. The two frequencies correspond to periods of almost exactly 24 and 29 days.

I fit a two term periodic smoothing model to this data. Figure 8 shows a plot of y_i versus (t_i mod 24), together with the fit produced by smoothing. Figure 9 shows a plot of the residuals from the first fit, r_i, versus (t_i mod 29).

Figures 10 and 11 show the result of backfitting (the frequencies are held fixed). The two functions look like discrete approximations to sines. The discreteness accounts for the presence of higher harmonics in a Fourier decomposition. This data appeared originally as an example in a textbook (Whittaker and Robinson (1924)), with no explicit reference. It is likely that it is artificial, constructed (in the nineteen twenties) with not very accurate approximations to sines.

Figures 10 and 11 have another interesting feature: there is one clear outlier, probably due to an error in manual computation of approximate sines. This outlier was also discovered by Bloomfield (1976), using a subtle analysis of the periodogram. Using periodic smoothing, the outlier is obvious. It is not clear whether Bloomfield's approach would have succeeded if there had been more than one outlier. The number of aberrant observations that can be detected by periodic smoothing is determined by

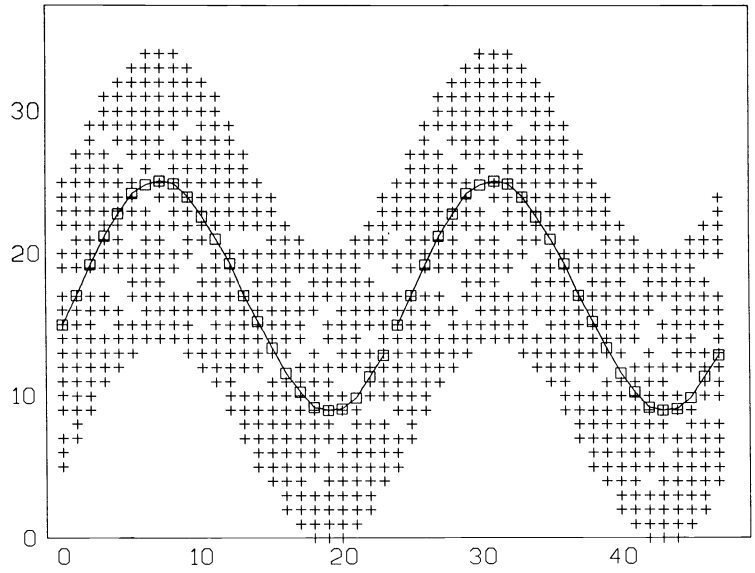


FIG. 8. *Variable star data. Brightness index vs. t* mod 24 *days. Pluses = original data; squares = smooth.*

the *breakdown* properties of the smoothing algorithm. Periodic smoothing is insensitive to a substantial percentage of gross errors if an appropriate smoother is used.

8. Example: the LYNX data continued. Figure 6, recall, is a plot of the MacKenzie River Lynx data versus t mod the fitted period of 9.64 years. The smooth is slightly skewed, with a slow rise and a faster fall. The skewness is of particular interest, as noted above, because solutions to simple predator-prey models have similar shape.

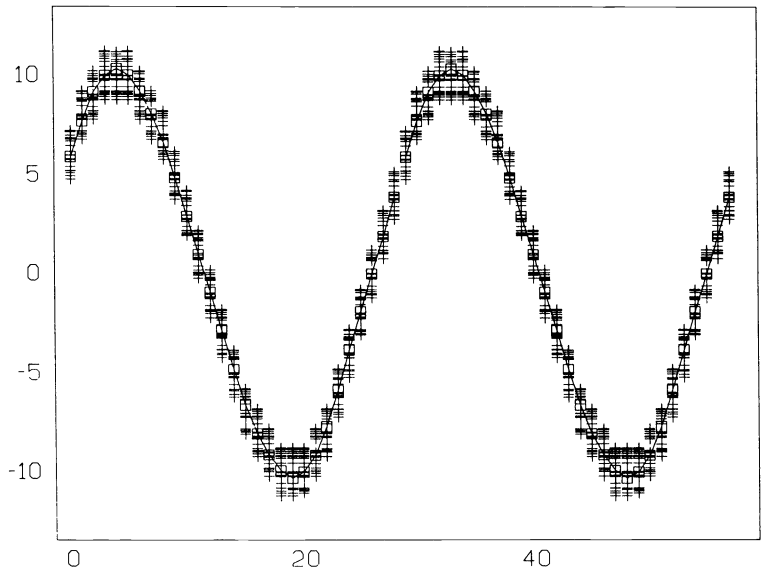


FIG. 9. *Variable star data. Residuals from* 24 *day term vs. t* mod 29 *days. Pluses = residuals from* 24 *day term; squares = smooth.*

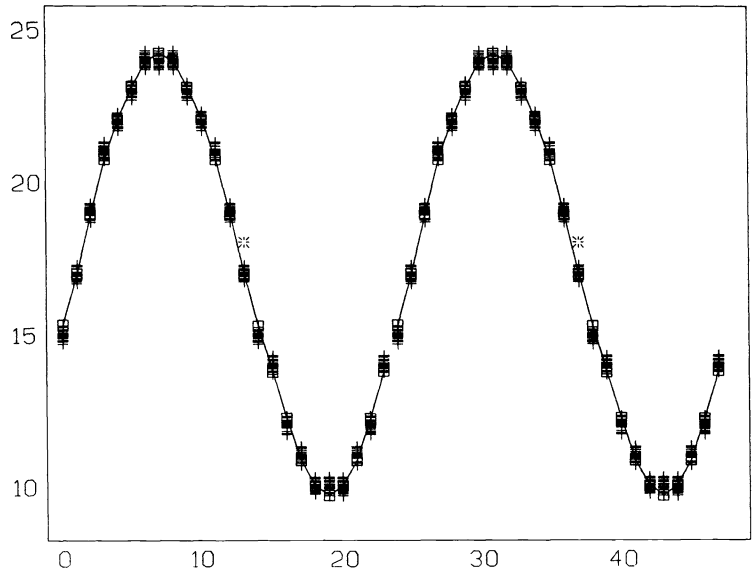


FIG. 10. *Variable star data. 24 day term after backfitting. Pluses = residuals from 29 day term; squares = smooth. Note outlier represented by star.*

Before investing much mental energy in pursuing this interpretation, we should assess how serious the indication of skewness is. In other words, we want to know if the skewness is *significant.*

It is not obvious how to test the significance of the apparent skewness. For any such test, even nonparametric methods like the bootstrap or permutation tests, significance levels will be unreliable, because we identified skewness as an interesting feature in exploration of the data.

What we need is to, somehow, repeat the experiment.

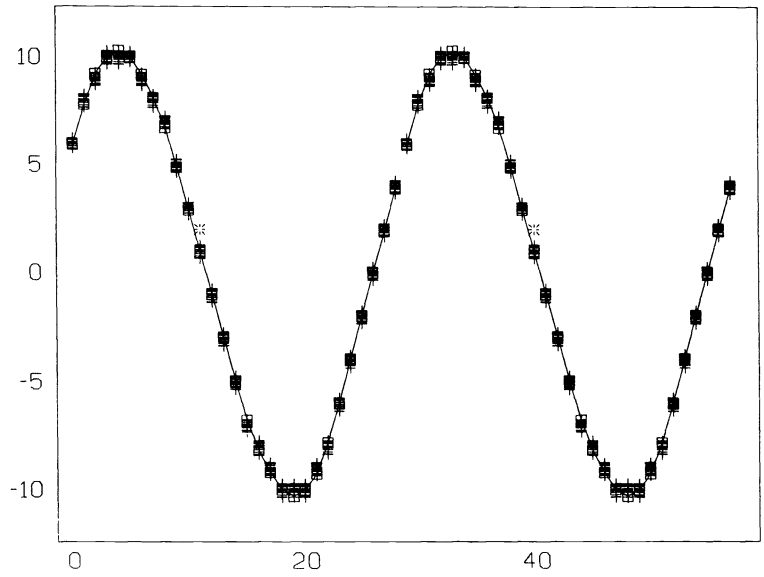


FIG. 11. *Variable star data. 29 day term after backfitting. Pluses = residuals from 24 day term; squares = smooth. Note outlier represented by star.*

Fortunately, we do have repetitions of the "lynx" experiment. Elton and Nicholson (1942) reported the number of lynx furs sold to the Hudson Bay Co. for 11 regions in western Canada, for the years 1821-1934. The MacKenzie River district data has been analyzed repeatedly (for example: Bulmer (1974), May (1974), Cambell and Walker (1977), Damsleth and Spjøtvoll (1982)); the other ten series have been ignored since they were first published. The reason for this is that the MacKenzie River series is the only one of the 11 that has no missing data. Periodic smoothing is not affected by missing observations, so we can repeat the analysis on the other series.

Figures 12-16 show log (# lynx furs) versus years for the 5 most complete of 10 other lynx series: the Athabasca Basin, West Central, Upper Saskatchewan, Winnipeg Basin, and North Central regions. These districts approximately cover the provinces of Manitoba, Alberta, and Saskatchewan. The MacKenzie River district covers a large part of the Northwest Territories. (See maps in Elton and Nicholson (1942).)

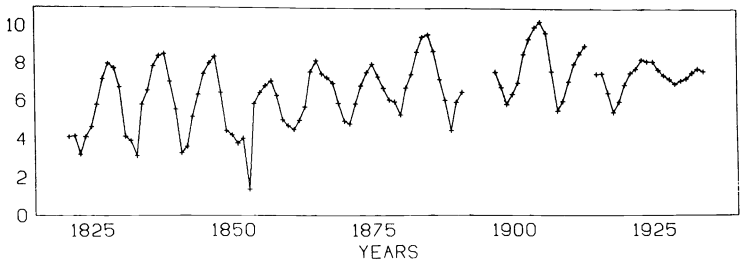


FIG. 12. *Athabasca Basin lynx data, original series.* Log (*counts*) *of lynx furs trapped in the Athabasca Basin district for the years* 1821-1934.

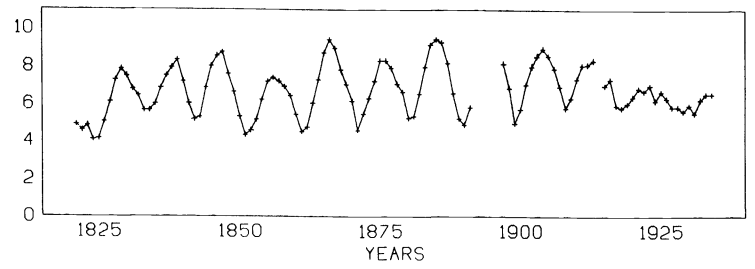


FIG. 13. *West Central lynx data, original series.* Log (*counts*) *of lynx furs trapped in the West Central district for the years* 1821-1934.

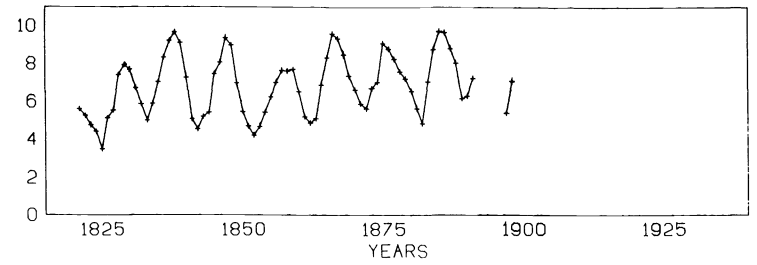


FIG. 14. *Upper Saskatchewan lynx data, original series.* Log (*counts*) *of lynx furs trapped in the Upper Saskatchewan district for the years* 1821-1898.

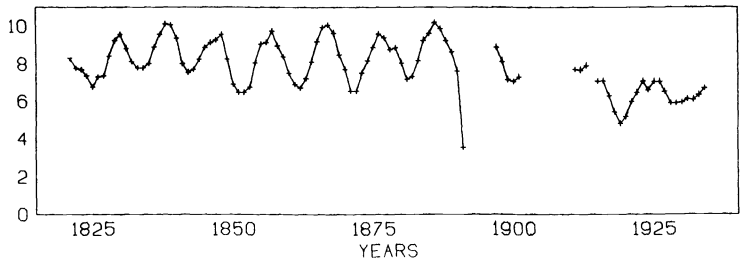


FIG. 15. *Winnipeg Basin lynx data, original series.* Log (*counts*) *of lynx furs trapped in the Winnipeg Basin district for the years* 1821-1934.

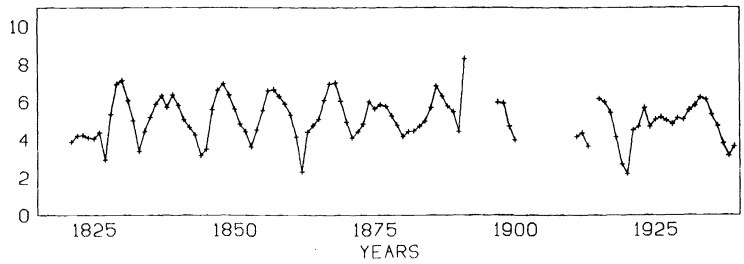


FIG. 16. *North Central lynx data, original series.* Log (*counts*) *of lynx furs trapped in the North Central district for the years* 1821-1939.

Unlike the MacKenzie River data, the other 5 series show clear long term trends. So for our analysis to be reasonable, we need to include a trend term.

If we overlay all six series on the same plot, we see that they stay almost perfectly in phase over the 119 years. So it seems reasonable to fit a periodic term with the same period to all 6 series. To do this, we simply choose ω to maximize $\sum_l R_l^2(\omega)$ where l indexes the series.

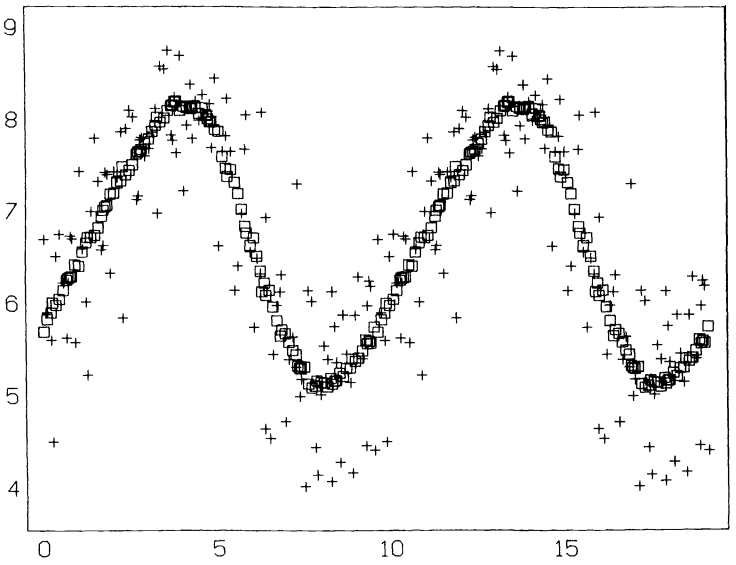


FIG. 17. *MacKenzie River lynx data, periodic term.* Log (*counts*) *vs.* t mod 9.55 *years. Pluses = original data adjusted for trend; squares = smooth.*

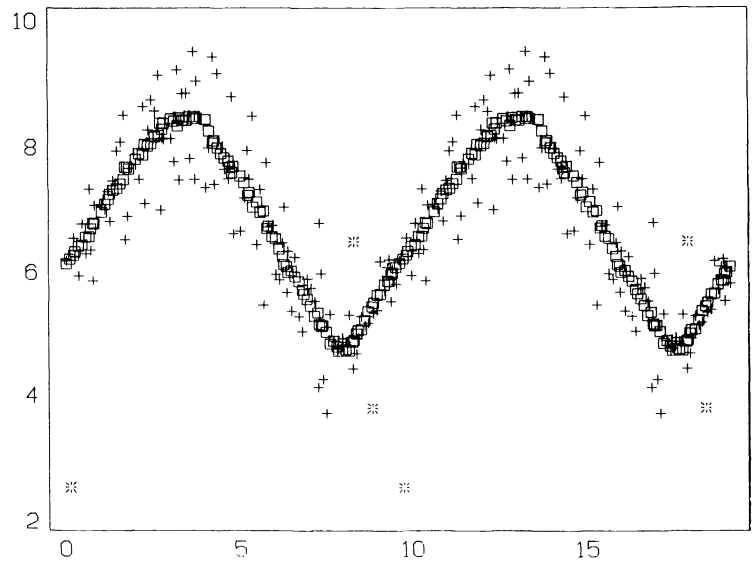


FIG. 18. *Athabasca Basin lynx data, periodic term.* Log (*counts*) *vs. t* mod 9.55 *years. Pluses = original data adjusted for trend; squares = smooth.*

Figures 17-34 show the results of fitting a two term model (one trend term and one periodic term) using the *Projection Pursuit Paradigm* and backfitting described above.

Figures 17-22 show the periodic term for each of the six series. The period, fitted to the six series simultaneously, is 9.55 years, slightly different from the 9.64 years fitted to the MacKenzie River data alone. The MacKenzie River data still look slightly

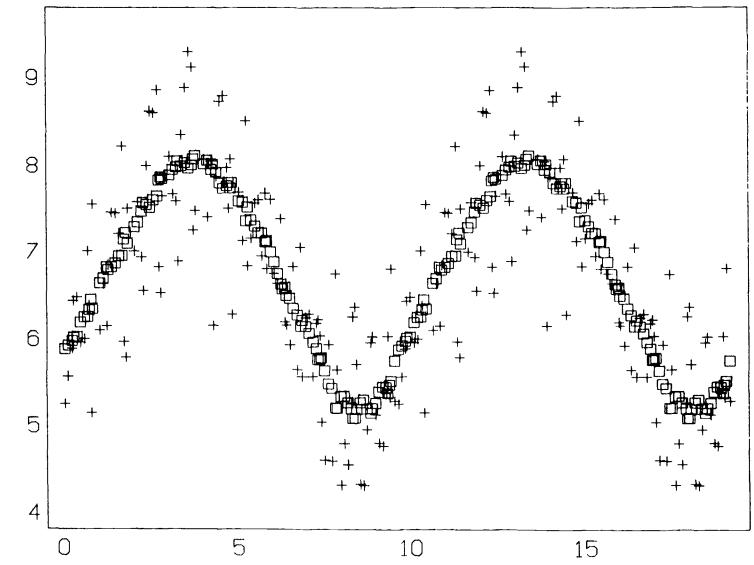


FIG. 19. *West Central lynx data, periodic term.* Log (*counts*) *vs. t* mod 9.55 *years. Pluses = original data adjusted for trend; squares = smooth.*

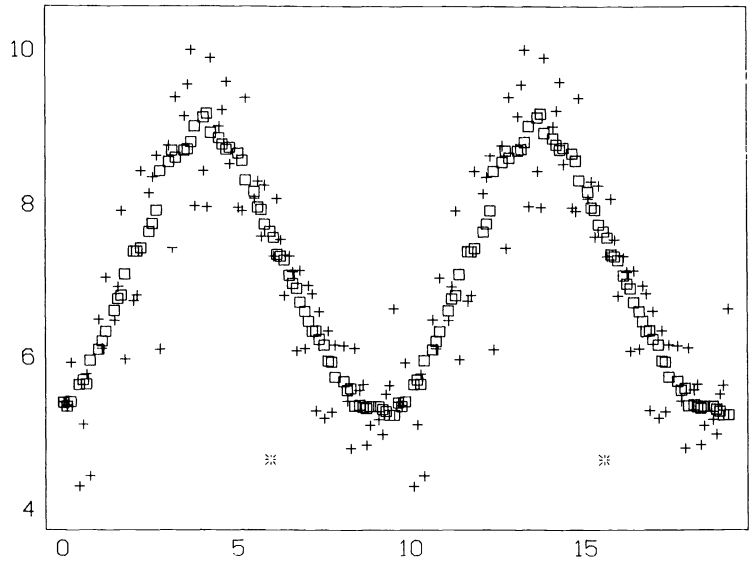


FIG. 20. *Upper Saskatchewan lynx data, periodic term.* Log (*counts*) vs. t mod 9.55 *years. Pluses = original data adjusted for trend; squares = smooth.*

skewed; the other five series are much less skewed, if at all. So the conclusion is that the skewness should not be taken seriously.

Figures 23–28 show the trend terms. There is no obvious pattern in the trends. They may reflect the fact that the boundaries of the regions changed over time (see maps in Elton and Nicholson (1942)).

Figures 29a–34a show original series (pluses) compared with the unwrapped model (squares). Figures 29b–34b show the unwrapped residuals from the model.

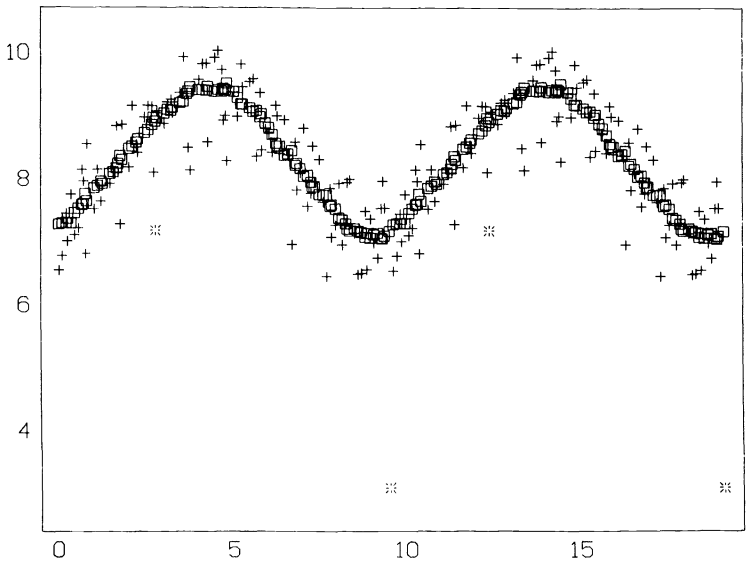


FIG. 21. *Winnipeg Basin lynx data, periodic term.* Log (*counts*) *vs.* t mod 9.55 *years. Pluses = original data adjusted for trend; squares = smooth.*

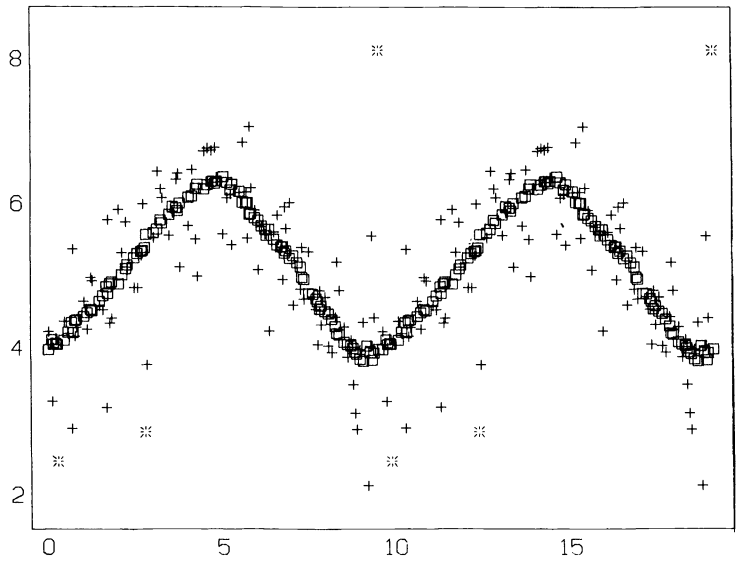


FIG. 22. *North Central lynx data, periodic term.* Log (*counts*) *vs. t* mod 9.55 *years. Pluses = original data adjusted for trend; squares = smooth.*

9. Extensions. It is important to consider more general models than the simple additive models (1) and (2).

9.1. Transform the response. One class of more general models is provided by transforming the response:

$$g(y_i) \approx \sum_{k=1}^{K} f_k(\omega_k t_i).$$

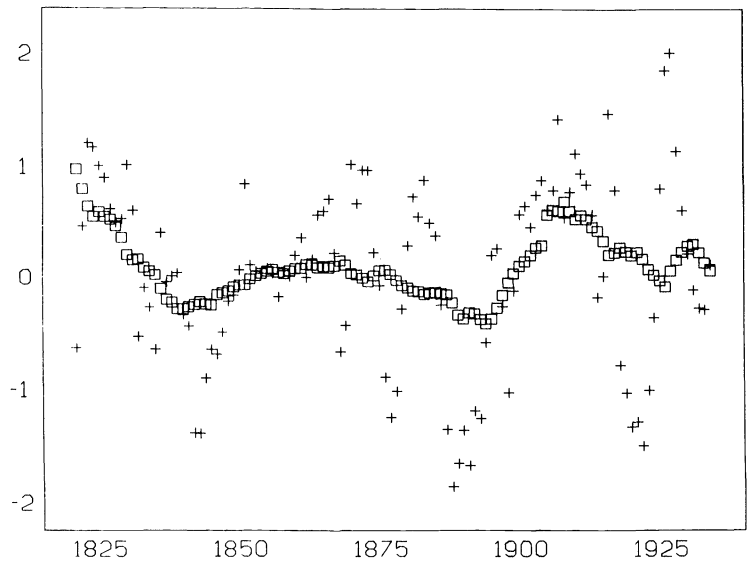


FIG. 23. *MacKenzie River lynx data, trend term.* Log (*counts*) *vs. years. Pluses = residuals from periodic term; squares = trend.*

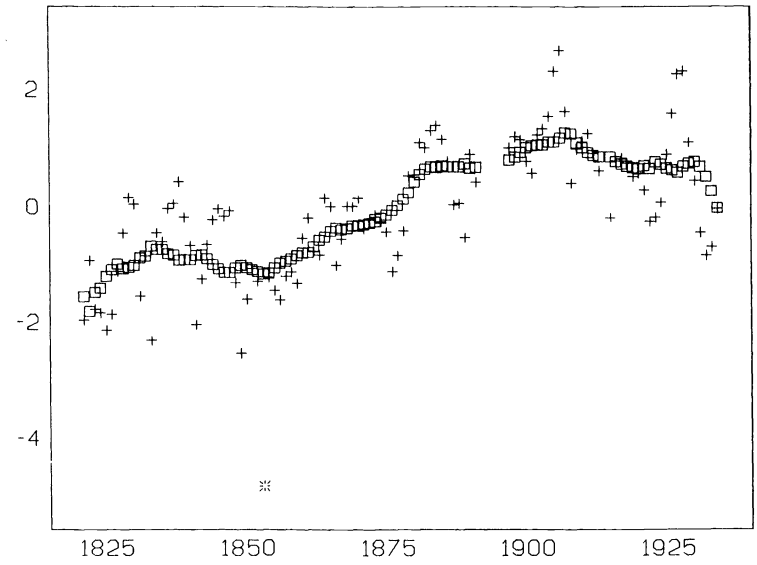


FIG. 24. *Athabasca Basin lynx data, trend term.* Log (*counts*) *vs. years. Pluses = residuals from periodic term; squares = trend.*

The transformation, $g(\cdot)$, can be chosen, interactively, on a graphics system like Orion I, or objectively, using an *ACE* style algorithm (Breiman and Friedman (1982)).

Interactive choice requires $g(\cdot)$ to be a member of a parametric family, for example, the power transformations. When the frequencies are unknown, it should be informative to watch the periodogram (or other estimate of the power spectrum) change as we manually change parameters in $g(\cdot)$. When the frequencies are known, then we might

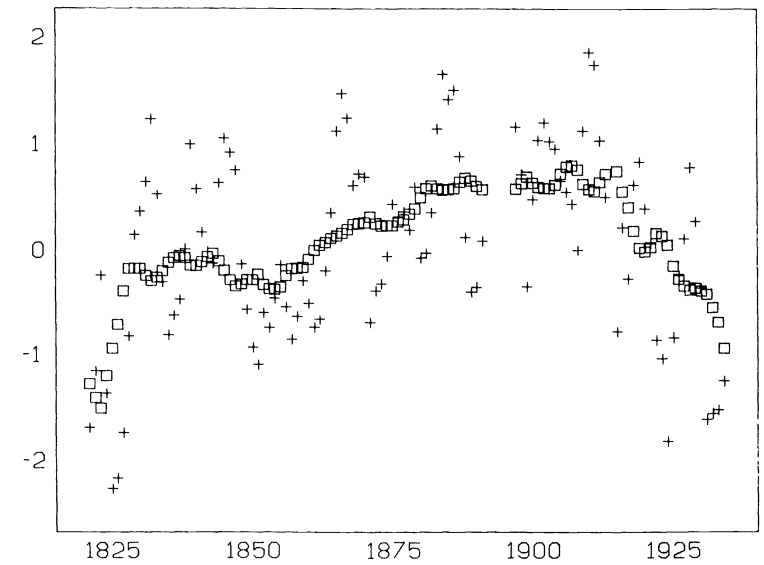


FIG. 25. *West Central lynx data, trend term* Log (*counts*) *vs. years. Pluses = residuals from periodic term; squares = trend.*

FIG. 26. *Upper Saskatchewan lynx data, trend term.* Log (*counts*) *vs. years. Pluses = residuals from periodic term; squares = trend.*

consider watching scatterplots of $g(y_i)$ versus $(\omega_k t_i \bmod 1)$ together with the curve of the smooth $f_k(\omega_k t_i)$ as we adjust $g(\cdot)$.

The *ACE* algorithm (Breiman and Friedman (1982)) provides an objective way to choose nonparametric transformations.

The optimal frequencies can be unstable under transformations of the response. For example, taking the logarithm of the response is equivalent to fitting a multiplicative model. If the data is approximately: $\cos(\omega_1 t) + \cos(\omega_2 t)$, it will also be well approxi-

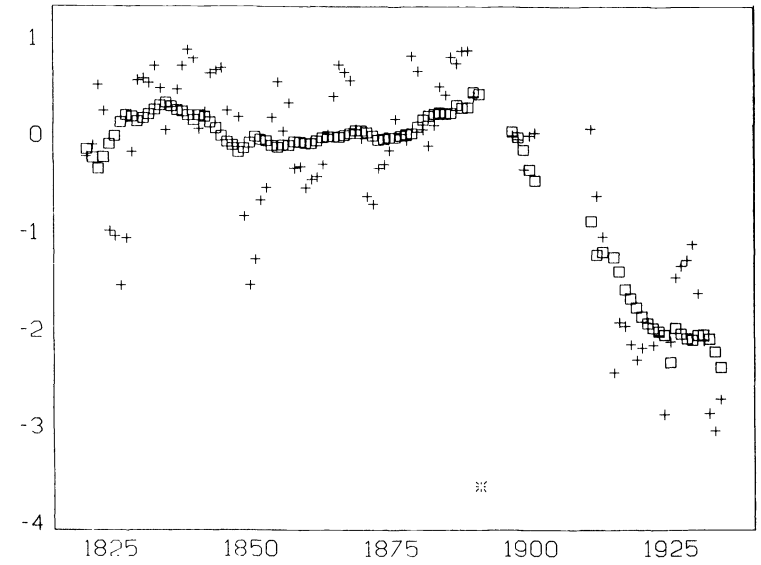


FIG. 27. *Winnipeg Basin lynx data, trend term.* Log (*counts*) *vs. years. Pluses = residuals from periodic term; squares = trend.*

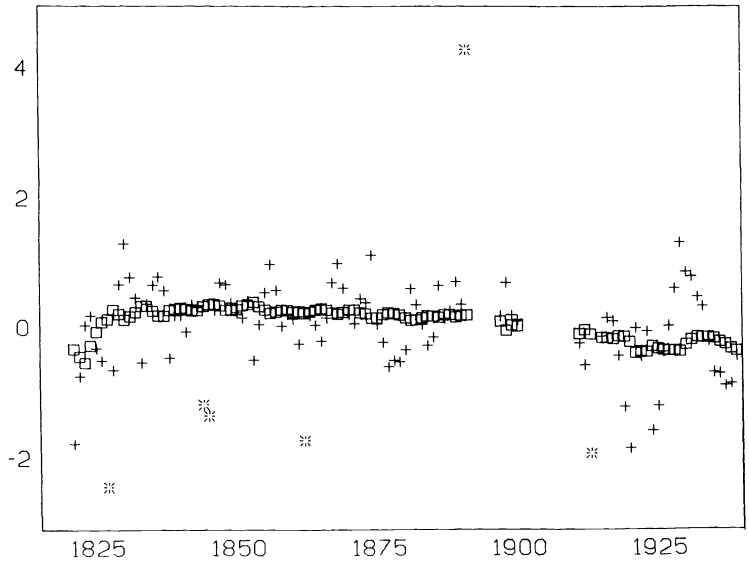


FIG. 28. *North Central lynx data, trend term.* Log (*counts*) *vs. years. Pluses = residuals from periodic term; squares = trend.*

mated by the multiplicative model: $2 \cdot \cos((\omega_1+\omega_2) \cdot t/2) \cdot \cos((\omega_1-\omega_2) \cdot t/2)$ *ACE* should be most useful when the frequencies are known beforehand, as is the case in seasonal adjustment.

9.2. Almost periodic smoothing of time series. For many data sets, a more realistic model than (1) is:

$$y_i \approx f(\omega t_i + \phi(t_i)),$$

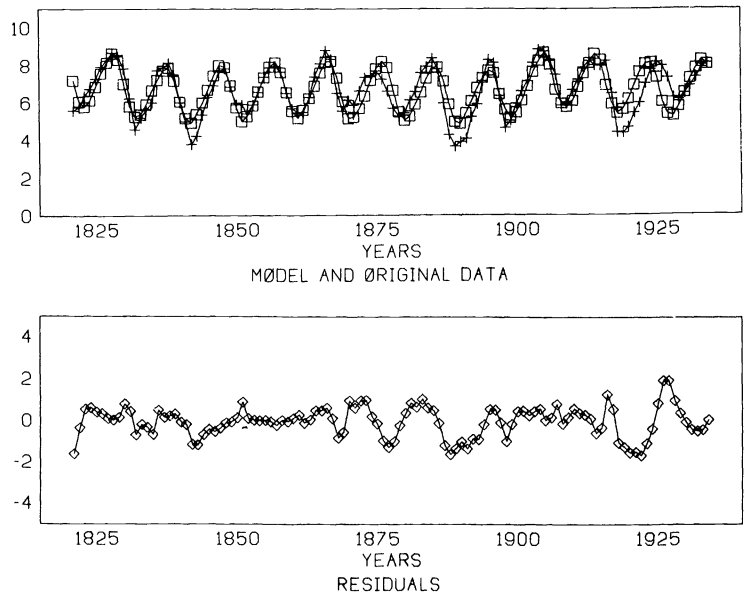


FIG. 29. *MacKenzie River lynx data.* (a) *Original Series* (*pluses*) *with unwrapped model* (*squares*). (b) *Residuals from model.*

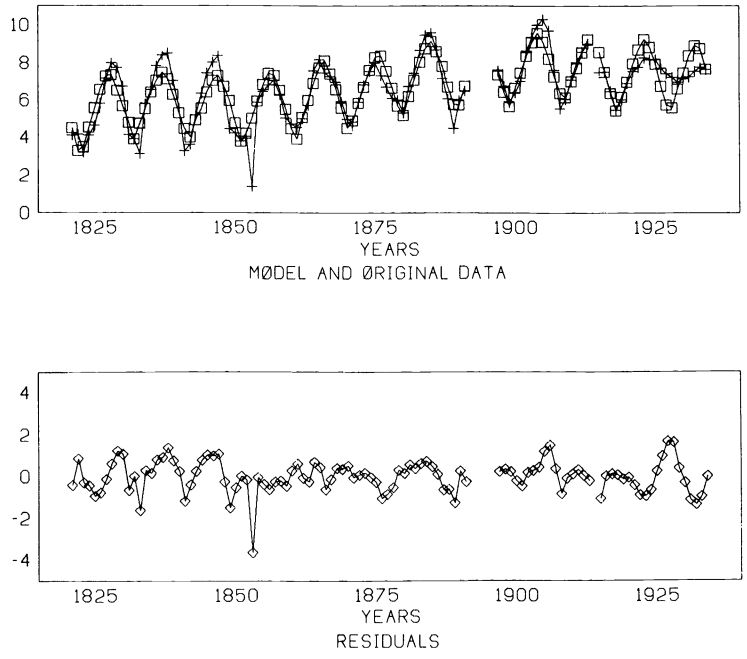


FIG. 30. *Athabasca Basin lynx data.* (a) *Original Series* (*pluses*) *with unwrapied model* (*squares*). (b) *Residuals from model.*

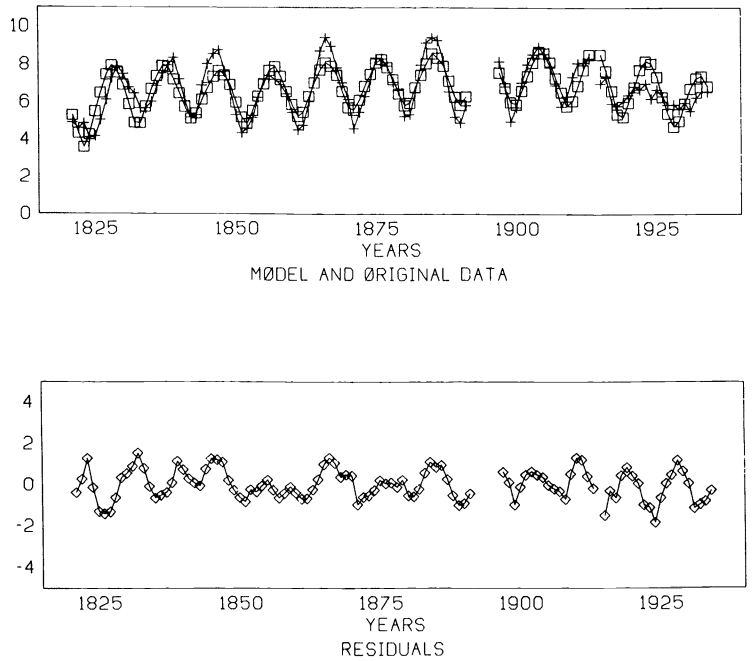


FIG. 31. *West Central lynx data.* (a) *Original Series* (*pluses*) *with unwrapped model* (*squares*). (b) *Residuals from model.*

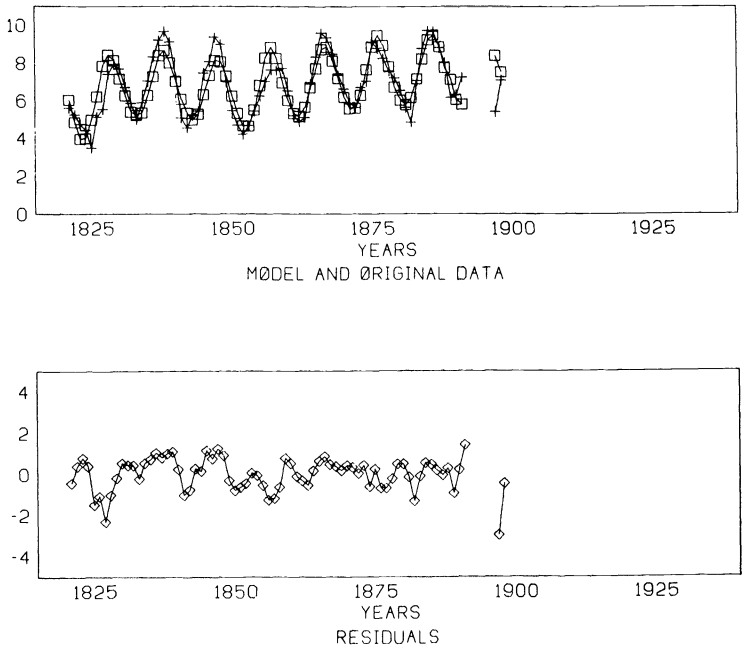


FIG. 32. *Upper Saskatchewan lynx data.* (a) *Original Series* (*pluses*) *with unwrapped model* (*squares*). (b) *Residuals from model.*

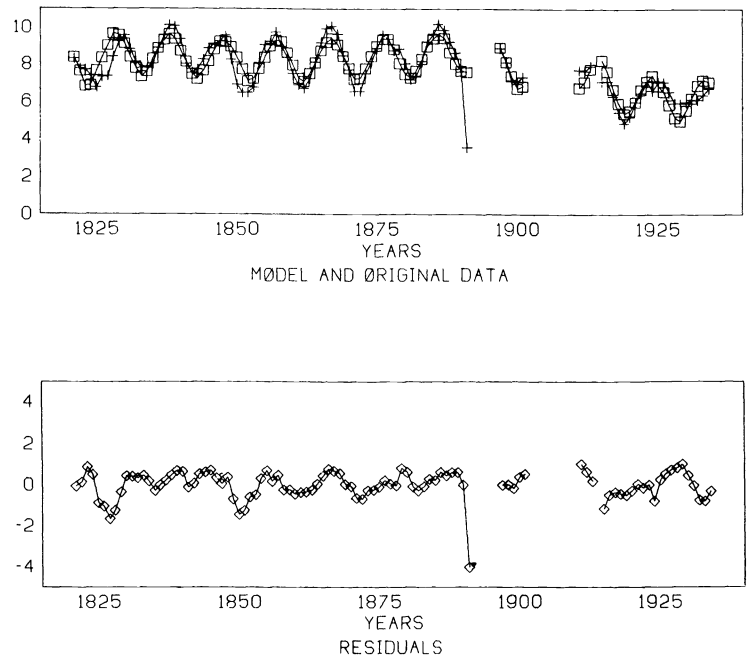


FIG. 33. *Winnipeg Basin lynx data.* (a) *Original Series* (*pluses*) *with unwrapped model* (*squares*). (b) *Residuals from model.*

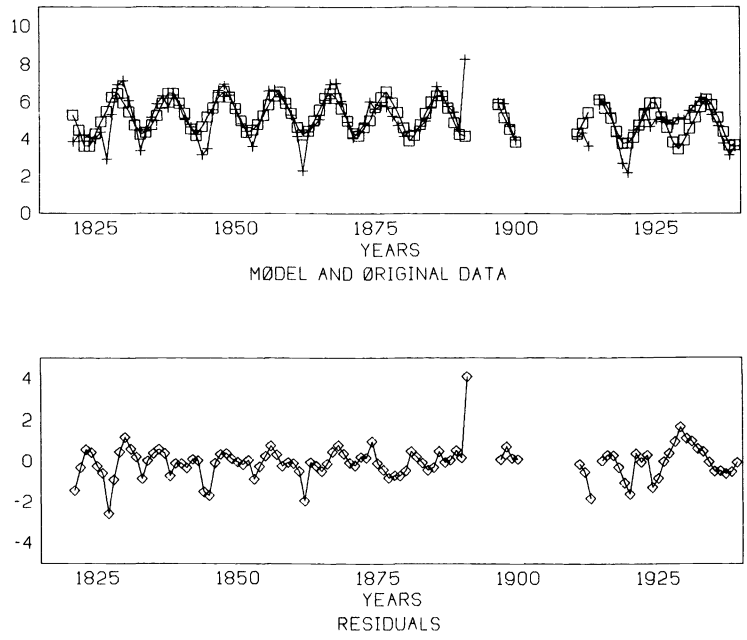


FIG. 34. *North Central lynx data.* (a) *Original Series* (*pluses*) *with unwrapped model* (*squares*) (b) *Residuals from model.*

where $\phi(\cdot)$ is a phase shift that varies slowly over time. Fitting a model of this type is analogous to *complex demodulation* (Bloomfield (1976), Bingham, Godfrey, and Tukey (1967)).

Consider the residuals from the Lynx data (Figs. 29b–34b). The series wanders slightly in and out of phase with the model. This is especially noticeable in several of the series around the time of World War I.

To fit $\phi(t_i)$, we first find an ω and an initial guess for $f(\cdot)$, assuming regular periodic variation as in the previous section. The initial guess for $\phi(t_i)$ is zero for all i. We then adjust $\phi(\cdot)$ and $f(\cdot)$ together, with the following iteration:

Repeat <
  Given $f(\cdot)$, get a rough adjustment for $\phi(t_i)$ to improve the fit to y_i (using, for example, a Taylor's expansion of $f(\cdot)$).
  Smooth $\phi(t_i)$ as a function of t_i to ensure that it varies slowly.
  Given $\phi(t_i)$, find $f(\cdot)$ by smoothing y_i as a function of $(\omega t_i + \phi(t_i)) \bmod 1$.
> Until (*convergence*).

REFERENCES

C. BINGHAM, M. D. GODFREY AND J. W. TUKEY (1967), *Modern techniques of power spectrum estimation*, IEEE Trans. Audio. Electroacoust., AU-15, pp. 56–66.

P. BLOOMFIELD (1976), *Fourier Analysis of Time Series: An Introduction*, John Wiley, New York.

L. BREIMAN AND J. H. FRIEDMAN (1982), *Estimating optimal transformations for multiple regression and correlation*, Dept. Statistics Tech. Rept. Orion 10, Stanford Univ., Stanford, CA.

M. G. BULMER (1974), *A statistical analysis of the ten year cycle in Canada*, J. Anim. Ecol., 43, pp. 701–716.

C. H. D. BUYS-BALLOT (1847), *Les changements périodique de température*, Utrecht.

M. J. CAMBELL AND A. M. WALKER (1977), *A survey of statistical work on the Mackenzie River series of annual Canadian lynx trapping for the years* 1821–1934 *and a new analysis*, JRSS A, 140, pp. 411–431.

W. S. CLEVELAND (1979), *Robust locally weighted regression and smoothing scatterplots*, J. Amer. Statist. Assoc., 74, pp. 828-836.

E. DAMSLETH AND E. SPJØTVOLL (1982), *Estimation of trigonometric components in time series*, J. Amer. Statist. Assoc., 77, pp. 381-387.

C. ELTON AND M. NICHOLSON (1942), *The ten year cycle of numbers of lynx in Canada*, J. Animal Ecol., 1, pp. 215-244.

J. H. FRIEDMAN AND W. STUETZLE (1981a), *Projection pursuit regression*, J. Amer. Statist. Assoc., 76, pp. 817-823.

——— (1981b), *Hardware for kinematic statistical graphics*, Dept. Statistics Tech. Rept. Orion 5, Stanford Univ., Stanford, CA.

——— (1982a), *Projection pursuit methods for data analysis*, in Modern Data Analysis, R. L. Launer and A. F. Siegel, eds.

——— (1982b), *Smoothing of scatterplots*, Dept. Statistics Tech. Rept. Orion 3, Stanford Univ., Stanford, CA.

R. M. HARALICK AND L. WATSON (1979), *A facet model for image data*, Proc. PRIP-79 (Proc. of the IEE Computer Society Conference on Pattern Recognition and Image Processing, 1979), pp. 489-495.

J. C. LERMAN (1982), Dept. Geosciences, Univ. Arizona, Tucson, personal communication.

R. M. MAY (1974), *Stability and Complexity in Model Ecosystems*, Princeton Univ. Press, Princeton, NJ.

J. A. MCDONALD (1982a), *Interactive graphics for data analysis*, Ph.D. thesis, Dept. Statistics, Stanford; available as Dept. Statistcs Tech. Rept. Orion 011, Stanford Univ., Stanford, CA.

——— (1982b), *Exploring data with the Orion I workstation*, a 25 minute, 16 mm sound film, which demonstrates programs described in McDonald (1982a). It is available for loan from: Jerome H. Friedman, Computation Research Group, Bin # 88, SLAC, P.O. Box 4349, Stanford, CA. 94305

——— (1982c), *Projection pursuit regression with the orion I workstation*, a 20 minute, 16 mm color sound film, which demonstrates programs described in McDonald (1982a). It is available for loan from: Jerome H. Friedman, Computation Research Group, Bin # 88, SLAC, P.O. Box 4349, Stanford, CA 94305.

J. A. MCDONALD AND A. OWEN (1983), *Smoothing with running split linear fits*, in preparation.

F. MOSTELLER AND J. W. TUKEY (1977), *Data Analysis and Regression*, Addison-Wesley, Reading, MA.

M. NAGAO AND T. MATSUYAMA (1979), *Edge preserving smoothing*, Computer Graphics and Image Processing, pp. 394-407.

A. SCHER, F. R. D. VELASCO AND A. ROSENFELD (1980), *Some new image smoothing techniques*, IEEE Trans. Systems, Man, and Cybernetics, SMC-10, pp. 153-158.

F. TOMITA AND S. TSUJI (1977), *Extraction of multiple regions by smoothing in selected neighborhoods*, IEEE Trans. Systems, Man, and Cybernetics, SMC-7, pp. 107-109.

E. T. WHITTAKER AND G. ROBINSON (1924), *The Calculus of Observations*, Blackie and Son, London.

SIAM J. SCI. STAT. COMPUT.
Vol. 7, No. 2, April 1986

THE DISTRIBUTION FUNCTION OF POSITIVE DEFINITE QUADRATIC FORMS IN NORMAL RANDOM VARIABLES*

ROBERT H. BROWN†

Abstract. Some existing representations for the cumulative distribution function of positive definite quadratic forms in normal random variables lead to inefficient computational algorithms. These inefficiencies are overcome with the derivation of some alternative recurrence relations. Some additional results concerning the distribution function and some extensions are also provided which make it possible to compute exact significance levels for certain test statistics proposed for the analysis of variance of unbalanced data.

Keywords. quadratic forms, positive definite, laguerre polynominals, gauss hypergeometric function

1. Introduction. Let the random vector Y have an n-variate normal distribution with mean vector θ and positive definite covariance matrix $V = TT'$. If the nonstochastic matrix B is positive definite and such that $TBT' = P'WP$ where P is orthonormal and W is a diagonal matrix of positive constants, then the quadratic form $Q = Y'BY$ can be written

$$\begin{aligned} Q &= Y'BY = (Y-\theta+\theta)'T^{-1}TBT'T'^{-1}(Y-\theta+\theta) \\ &= (Z+\mu)'W(Z+\mu) \\ &= \sum_{i=1}^{n} w_i(z_i+\mu_i)^2 \end{aligned} \tag{1}$$

where $Z = PT'^{-1}(Y-\theta)$, $\mu = PT'^{-1}\theta$. The constants w_i are assumed to be ordered so that $w_1 < w_2 < \ldots < w_n$. The random vector Z has an n-variate normal distribution with mean vector zero and covariance matrix I_n.

The distribution function of the random variable Q has been extensively studied. A comprehensive survey can be found in Johnson and Kotz [1970]. Methods for computing the distribution function have generally taken one of two approaches. Rice [1980], Davies [1980] and Imhof [1961] describe methods based on numerically inverting the characteristic function of Q. These methods are applicable in the more general case where the coefficients of the linear combinations (1) are permitted to assume negative values. Confined to the positive definite case, Sheil and O'Muircheartaigh [1977] and Johnson and Kotz [1968] present algorithms based on particular infinite series representations of the distribution function of Q.

The algorithms based on infinite series representations suffer from several computational disadvantages. The current paper shows how these disadvantages can be easily overcome and presents some new results concerning the distribution function of Q.

2. Infinite series representations. It can be shown (see Johnson and Kotz [1970]) that the probability density function $f(Q;y)$ and the cumulative distribution function $F(Q;y)$ of the positive definite quadratic from Q evaluated at y can be written

*Received by the editors June 26, 1984, and in revised form March 25, 1985. This paper represents a portion of the author's doctoral dissertation which was submitted to the Department of Statistics, University of Kentucky. It was typeset at McNeil Pharmaceutical, Spring House, Pennsylvania, using Compugraphic MCS20/8400 phototypesetter. Final copy was produced on September 25, 1985.

†Manager of Statistics, McNeil Pharmaceutical, Spring House, PA 19477.

$$f(Q;y) = (1/\beta) \sum_{k=0}^{\infty} c_k g(n+2k,\ y/\beta) \tag{2}$$

$$= [\Gamma(n/2)/2] g(n,y/\beta) \sum_{k=0}^{\infty} k! a_k L_k^{(\frac{n}{2}-1)}(y/2\beta)/\Gamma(k+n/2), \tag{3}$$

$$F(Q;y) = \sum_{k=0}^{\infty} c_k G(n+2k,\ y/\beta) \tag{4}$$

$$= G(n,y/\beta) + (y/2\beta)^{n/2} e^{-y/2\beta} \sum_{k=1}^{\infty} (k-1)! a_k L_{k-1}^{(\frac{n}{2})}(y/2\beta)/\Gamma(k+n/2) \tag{5}$$

where $g(m,x)$ and $G(m,x)$ are the density function and cumulative distribution function, respectively, of a central chi-squared random variable with m degrees of freedom and $L_r^{(\alpha)}(x)$ is the generalized Laguerre polynomial.

The coefficients in the above expansions are defined by the following recurrence relations

$$kc_k = \sum_{r=1}^{k} c_{k-r} d_r, \tag{6}$$

$$ka_k = \sum_{r=1}^{k} a_{k-r} b_r \tag{7}$$

where

$$d_r = \tfrac{1}{2} r \sum_{i=1}^{n} \gamma_i^{r-1} \mu_i^2 + \tfrac{1}{2} \sum_{i=1}^{n} (1 - r\mu_i^2) \gamma_i^r \tag{8}$$

with $\gamma_i = (1-\beta/w_i)$,

$$c_0 = e^{-\lambda} \prod_{i=1}^{n} [\beta/w_i]^{1/2},\ \lambda = \tfrac{1}{2} \sum_{i=1}^{n} \mu_i^2$$

and

$$b_r = -\tfrac{1}{2} r \sum_{i=1}^{n} \gamma_i^{r-1} \mu_i^2 + \tfrac{1}{2} \sum_{i=1}^{n} (1 + r\mu_i^2) \gamma_i^r \tag{9}$$

with $\gamma_i = (1-w_i/\beta)$, $a_0 = 1$.

The parameter β is a positive constant chosen to ensure the convergence of whichever series is being used. For some choices of β the series (2)-(5) are uniformly convergent. For computational purposes Kotz, et al [1967a] suggest that $\beta = (w_1 + w_n)/2$ for the Laguerre polynomial expansion and $\beta = 2w_1 w_n/(w_1 + w_n)$ for the central chi-squared expansion are close to optimal choices. The constants n, w_i and μ_i are from (1) the definition of Q as a linear combination of noncentral chi-squareds.

The algorithms of both Sheil and O'Muircheartaigh [1977] and Johnson and Kotz [1968] use the recurrence relations (6)-(9) to compute the coefficients. For computational purposes these equations are unattractive for two related reasons. Using (7) and (9) to compute a_k requires that all previously computed a's and b's are available. This requires work vectors whose lengths depend on the number of partial sums needed for numerical convergence. Since an inner product of the vector of previous a's with the vector of previous b's must be computed, the number of arithmetic operations needed to compute a_k increases as k increases.

The next section will present a simple rearrangement of calculations which avoids both of these difficulties.

3. Computing the coefficients. Examination of (6)-(9) shows that with minor changes in sign and definition of the γ_i the same recurrence relations hold for the coefficients from either expansion. The following derivation is based on the formulae for the Laguerre series coefficients.

From (7) and (9) it can be seen that

$$ka_k = \tfrac{1}{2} \sum_{r=1}^{k} [-r \sum_{i=1}^{n} \gamma_i^{r-1}\mu_i^2 + \sum_{i=1}^{n} (1+r\mu_i^2)\gamma_i^r]a_{k-r}$$

$$= \tfrac{1}{2} \sum_{i=1}^{n} [\sum_{r=1}^{k} \gamma_i^r a_{k-r} - \mu_i^2 \sum_{r=1}^{k} r\gamma_i^{r-1}a_{k-r} + \mu_i^2 \sum_{r=1}^{k} r\gamma_i^r a_{k-r}]$$

$$= \tfrac{1}{2} \sum_{i=1}^{n} (p_{i,k} - f_{i,k} + \gamma_i f_{i,k}) \tag{10}$$

where

$$p_{i,k} = \sum_{r=1}^{k} \gamma_i^r a_{k-r}, \tag{11}$$

$$f_{i,k} = \mu_i^2 \sum_{r=1}^{k} r\gamma_i^{r-1}a_{k-r} \tag{12}$$

Recurrence relations for the $p_{i,k}$ and $f_{i,k}$ can easily be derived. From (11)

$$p_{i,k+1} = \sum_{r=1}^{k+1} \gamma_i^r a_{k+1-r}$$

$$= \sum_{s=0}^{k} \gamma_i^{s+1} a_{k-s}$$

$$= \gamma_i a_k + \gamma_i \sum_{s=1}^{k} \gamma_i^s a_{k-s}$$

$$= \gamma_i(a_k + p_{i,k}) \tag{13}$$

Similarly, from (12)

$$f_{i,k+1} = \mu_i^2 \sum_{r=1}^{k+1} r\gamma_i^{r-1}a_{k+1-r}$$

$$= \mu_i^2 \sum_{s=0}^{k} (s+1)\gamma_i^s a_{k-s}$$

$$= \mu_i^2[a_k + \sum_{s=1}^{k} \gamma_i^s a_{k-s} + \gamma_i \sum_{s=1}^{k} s\gamma_i^{s-1}a_{k-s}]$$

$$= \mu_i^2(a_k + p_{i,k}) + \gamma_i f_{i,k}. \tag{14}$$

Conceptually (13) and (14) are derived by recognizing that $p_{i,k}$ is a k-th degree polynomial in γ_i and then by applying Horner's scheme (Ralston [1965]) to the evaluation of that polynomial.

There are only n of the $p_{i,k}$'s and $f_{i,k}$'s which are updated (i.e. overwritten) recursively using (13) and (14). The coefficient a_{k+1} is then computed using (10). Only the k-th coefficient is saved rather than all previous coefficients. Explicit computation of the b_r's is no longer required. The number of arithmetic operations required to compute the k-th coefficient is now a function of n, the number of parameters, and hence does not increase as k increases.

As stated previously the coefficients of the central chi-squared expansion follow essentially the same recurrence relations. Without presenting the algebraic details, the final results can be stated as follows:

$$kc_k = \tfrac{1}{2} \sum_{i=1}^{n} (p_{i,k} + f_{i,k} - \gamma_i f_{i,k})$$

where $p_{i,k}$ is defined by (11) and therefore follows the recurrence (13) and $f_{i,k}$ is defined by (12) and therefore follows the recurrence (14). However, since the expansion is now in central chi-squared distribution functions rather than Laguerre polynomials, in all these formulae γ_i is now defined by (8) as $\gamma_i = (1-\beta/w_i)$.

The above recurrence relations in combination with well-known results for either the generalized Laguerre polynomials or the central chi-square distribution function provide simple and efficient algorithms for the computation of $F(Q;y)$ or $f(Q;y)$. See Kotz, et al. [1967a&b] for a discussion of truncation error bounds and convergence criteria.

4. Some additional results concerning $F(Q;y)$. For the central chi-squared expansion (2) and (4) Ruben [1963] has shown that if $\beta < w_1$, then

$$c_k = Ae^{-\lambda}E[Q_1^r]/r! \tag{15}$$

where

$$A = \prod_{i=1}^{n} (\beta/w_i)^{1/2},$$

$$\lambda = \sum_{i=1}^{n} \mu_i^2/2,$$

$$Q_1 = \tfrac{1}{2} \sum_{i=1}^{n} (1-\beta/w_i)[z_i+\mu_i\beta^{1/2}/(w_i+\beta)^{1/2}]^2,$$

$z_1, \ldots, z_n$ are iid $N(0,1)$ random variables and $E[Y]$ denotes the expected value of the random variable Y.

A similar result can easily be established for the coefficients from the Laguerre polynomial expansion. To this end it can be seen that

$$\begin{aligned} E[L_k^{(n/2-1)}(Q/2\beta)] &= \int_0^\infty L_k^{(n/2-1)}(y/2\beta)f(Q;y)dy \\ &= \sum_{r=0}^{\infty} \frac{r!a_r}{\Gamma(n/2+r)} \int_0^\infty e^{-t}t^{n/2-1}L_k^{(n/2-1)}(t)\, L_r^{(n/2-1)}(t)\, dt \\ &= a_k \end{aligned} \tag{16}$$

which follows from the orthogonality properties of the Laguerre polynomials. Interchange of summation and integration is justified by the uniform convergence of (3) for appropriate choice of β.

The central chi-squared distribution function $G(p,y)$ with p degrees of freedom follows the well-known recurrence relation

$$G(p+2,y) = G(p,y)-2g(p+2,y)$$

where $g(p+2,y)$ represents the corresponding density function. The above statistical characterizations of the coefficients from (4) and (5) can be used to provide a similar result for the distribution function of a positive definite quadratic form. In particular let the random variable Q be defined by (1) and let $Q' = Q + w\chi_2^2$ where $w_1 \le w \le w_n$ and χ_2^2 is a central chi-square with 2 degrees of freedom distributed independently of Q. If $w = w_r$ for some $r = 1, \ldots, n$ then Q' can be expressed as

$$Q' = \sum_{i=1}^{n} w_i\chi_{1+2\delta_{ir}\lambda_i}^2$$

where $\delta_{ir} = 1$ if $i = r$ and 0 otherwise. The statistic Q' is a linear combination of chi-squared random variables in which each chi-squared has 1 degree of freedom with the exception of the r-th chi-squared which has 3 degrees of freedom. Let c_k and a_k be defined by (6) and (7) as the coefficients in the central chi-square and Laguerre polynomial expansion, respectively, of the cumulative distribution function of Q. Similarly, let c_k' and

a_k' be the coefficients in the corresponding expansions of the distribution function of Q'. The following lemma relates c_k' to c_k and a_k' to a_k.

LEMMA 1. *Let c_k and c_k', a_k and a_k', Q and Q' be as defined above. For the central chi-square expansion assume that β is chosen so that (15) holds. For the Laguerre polynomial expansion, assume that β is chosen so that (16) holds. Then*

$$1.\ c_k' = (\beta/w)c_k + (1-\beta/w)c_{k-1}',$$

$$2.\ a_k' = a_k + (1-\beta/w)a_{k-1}'$$

with $c_{-1}' = a_{-1}' = 0$.

Proof. To prove 1. first note that since $w \geq w_1$ the same choice of β suffices to make (15) hold for both c_k and c_k'. Therefore, from (15)

$$k!c_k' = A' e^{-\lambda} E[(Q_1')^k]$$

where $A' = (\beta/w)\, A$,

$$Q_1' = Q_1 + \tfrac{1}{2}(1-\beta/w)\chi_2^2$$

and A, Q_1 and λ are from (15).
Clearly, when $k=0$, $c_k' = (\beta/w)c_k$. For $k>0$

$$\begin{aligned} k!c_k' &= (\beta/w)Ae^{-\lambda}E\{[Q_1 + \tfrac{1}{2}(1-\beta/w)\chi_2^2]^k\} \\ &= (\beta/w)Ae^{-\lambda}\sum_{r=0}^{k} k!E[Q_1^{k-r}/(k-r)!](\tfrac{1}{2})^r(1-\beta/w)^r E[\chi_2^{2r}/r!] \\ &= k!(\beta/w)\sum_{r=0}^{k} c_{k-r}(1-\beta/w)^r. \end{aligned}$$

From this equation it can be seen that

$$\begin{aligned} c_k' &= (\beta/w)\sum_{s=-1}^{k-1} c_{k-1-s}(1-\beta/w)^{s+1} \\ &= (\beta/w)c_k + (1-\beta/w)(\beta/w)\sum_{s=0}^{k-1} c_{k-1-s}(1-\beta/w)^s \\ &= (\beta/w)c_k + (1-\beta/w)c_{k-1}' \end{aligned}$$

To prove 2. note that since $w < w_n$ the same choice of β suffices to make (16) hold for both a_k and a_k'. Therefore

$$\begin{aligned} a_k' &= E[L_k^{(n/2)}(Q/2\beta + (w/2\beta)\chi_2^2)] \\ &= E[\sum_{r=0}^{k} L_{k-r}^{(n/2-1)}(Q/2\beta)L_r(w\chi_2^2/2\beta)] \end{aligned}$$

from a standard addition formula for the generalized Laguerre polynomials. Then

$$\begin{aligned} a_k' &= \sum_{r=0}^{k} E[L_{k-r}^{(n/2-1)}(Q/2\beta)]E[L_r(w\chi_2^2/2\beta)] \\ &= \sum_{r=0}^{k} a_{k-r}(1-w/\beta)^r \\ &= a_k + (1-w/\beta)\sum_{s=0}^{k-1} a_{k-1-s}(1-w/\beta)^s \\ &= a_k + (1-w/\beta)a_{k-1}'. \end{aligned}$$

Clearly, $a_0' = a_0 = 1$.

Using these results the following theorem can be established.

THEOREM 1. *Let Q and Q' be as defined above. Then*

$$F(Q';y) = F(Q;y) - 2wf(Q';y).$$

Proof. From (4) with $\beta < w_1$

$$\begin{aligned} F(Q';y) &= \sum_{k=0}^{\infty} c_k' G(n+2+2k, y/\beta) \\ &= \sum_{k=0}^{\infty} c_k' [G(n+2k, y/\beta) - 2g(n+2+2k, y/\beta)] \\ &= \sum_{k=0}^{\infty} c_k' G(n+2k, y/\beta) - 2\beta f(Q';y) \\ &= \sum_{k=0}^{\infty} [(\beta/w)c_k + (1-\beta/w)c_{k-1}'] G(n+2k, y/\beta) - 2\beta f(Q';y) \\ &= (\beta/w)F(Q;y) + (1-\beta/w)F(Q';y) - 2\beta f(Q';y). \end{aligned}$$

Collecting terms gives the desired result.

5. A generalization. Let Q be as defined in (1) and let χ_p^2 be an independently distributed central chi-square with p degrees of freedom. $\Pr\{Q \le s\chi_p^2\}$ where s is a positive constant is of some interest and can be easily obtained from the preceding results.

THEOREM 2. *Let Q and χ_p^2 be as defined above. Then*

$$\begin{aligned} F(Q/\chi_p^2; s) &= Pr\{Q \le s\chi_p^2\} = Pr\{\chi_n^2 \le (s/\beta)\chi_p^2\} \\ &\quad + [(n/2)B(n/2, p/2)]^{-1} u^{n/2}(1-u)^{p/2} \sum_{k=1}^{\infty} a_k {}_2F_1[-k+1, (n+p)/2; (n+2)/2; u] \end{aligned}$$

where $B(a,b)$ is the complete beta function, β and a_k are defined by (7) and (9), $u = s/(s+\beta)$ and ${}_2F_1(a,b;c;x)$ is the usual Gauss hypergeometric function.

Proof. $F(Q/\chi_p^2; s) = E_Y[F(Q;sy) | Y=y] = E_Y[G(n; sy/2\beta)]$

$$+ \sum_{k=1}^{\infty} \frac{(k-1)! a_k}{\Gamma(n/2+k)} E_Y[e^{-sy/2\beta}(sy/2\beta)^{n/2} L_{k-1}^{(n/2)}(sy/2\beta)] \tag{17}$$

where Y is a random variable distributed as χ_p^2. It can be seen that

$$\begin{aligned} &E_Y[e^{-sy/2\beta}(sy/2\beta)^{n/2} L_{k-1}^{(n/2)}(sy/2\beta)] \\ &= [\Gamma(p/2)]^{-1}(\beta/s)^{p/2} \int_0^{\infty} e^{-(\beta/s+1)x} x^{(n+p)/2 - 1} L_{k-1}^{(n/2)}(x)\, dx \\ &= [\Gamma(p/2)]^{-1}(\beta/s)^{p/2} \mathcal{L}[x^{(n+p)/2-1} L_{k-1}^{(n/2)}(x); \beta/s+1] \end{aligned}$$

where $\mathcal{L}[f(x);c]$ denotes the Laplace transform of $f(x)$ with parameter c. From tables of the Laplace transform

$$\begin{aligned} &\mathcal{L}[x^{(n+p)/2-1} L_{k-1}^{(n/2)}(x); \beta/s+1] \\ &= \Gamma[(n+p)/2]\, {}_2F_1[-k+1, (n+p)/2; n/2+1; s/(\beta+s)]. \end{aligned}$$

Substituting this expression into (17) gives the desired result.

6. Discussion. The methods used in Section 2 along with the results of Section 5 can be generalized to deal with the distribution function of a nondefinite quadratic form. This will be the subject of a subsequent paper.

The distribution function obtained in Section 5 is of interest in several special contexts. In the special case $w_1 = w_2 = \ldots = w_n$ Theorem 2 gives a representation for the cumulative distribution function of a noncentral F random variable with n and p degrees of freedom and noncentrality parameter $\lambda = \Sigma_{i=1}^{n} \mu_i^2$. Some approaches to dealing with unbalanced data in the analysis of variance of cross-classified data lead to sums of squares of hypotheses which are distributed as linear combinations of chi-squares. Hirotsu [1979] discusses approximations to the distributions of test statistics derived from such approaches.

Theorem 2 provides the basis for a method by which exact (under the usual assumptions) significance levels of these types of procedures can be computed. Since Theorem 2 also gives the noncentral distribution, the power of these methods can also be studied.

REFERENCES

R.B. DAVIES (1980), *The distribution of a linear combination of χ^2 variables,* Appl. Stat., 30, p.p. 323-333.

C. HIROTSU (1979), *An F approximation and its applications,* Biometrika, 66, p.p. 577-584.

J.P. IMHOF (1961), *Computing the distribution of quadratic forms in normal variables,* Biometrika, 48, p.p. 419-426.

N.L. JOHNSON AND S. KOTZ (1968), *Tables of distributions of positive definite quadratic forms in central normal variables,* Sankhya, Ser. B., 30, p.p. 303-314.

N.L. JOHNSON AND S. KOTZ (1970), *Continuous Univariate Distributions, Vol. 2,* Houghton Mifflin Inc., Boston.

S. KOTZ, N.L. JOHNSON AND D.W. BOYD (1967a), *Series representations of distributions of quadratic forms in normal variables. I. Central case,* Ann. Math. Statist., 38, p.p. 823-837.

S. KOTZ, N.L. JOHNSON AND D.W. BOYD (1967b), *Series representations of distributions of quadratic forms in normal variables. II. Noncentral case,* Ann. Math. Statist., 38, p.p. 838-848.

A. RALSTON (1965), *A First Course in Numerical Analysis,* McGraw-Hill, New York.

S.O. RICE (1980), *Distribution of quadratic forms in normal random variables-evaluation by numerical integration,* SIAM J. Sci. Stat. Comput., 4, p.p. 438-448.

H. RUBEN (1963), *A new result on the distribution of quadratic forms,* Ann. Math. Statist., 34, p.p. 1582-1584.

J. SHEIL AND I. O'MUIRCHEARTAIGH (1977), *The distribution of nonnegative quadratic forms in normal variables,* Appl. Stat., 26, p.p. 92-98.

SIAM J. SCI. STAT. COMPUT.
Vol. 7, No. 2, April 1986

024

VARIATION DIMINISHING SPLINES IN SIMULATION*

WILLIAM M. COUGHRAN, JR.† ERIC GROSSE† AND DONALD J. ROSE‡

Abstract. Variation diminishing splines provide an effective tool for modeling active elements in circuit simulation. Using quadratic tensor product splines and maintaining uniform sampling at the boundary by linear extension of the data yields an algorithm that is smooth (unlike simple table lookup), shape preserving (unlike simple interpolation), and efficient (30 microseconds to evaluate on a Cray-1A). The rate of convergence to function and derivative values and to the location of minima is $O(h^2)$.

Key words. monotonicity, convexity, table model, transistor model

AMS(MOS) subject classifications. 65D07, 41A15.

1. Introduction. Few methods are known for the monotone approximation of data in several variables. Tensor product variation diminishing splines provide a simple, efficient, and effective solution for problems where the data need not be exactly interpolated. This paper analyzes this method for the particular case of quadratic splines on uniformly spaced knots and discusses the application to an important modeling problem in electrical circuit simulation.

Tensor product splines are popular tools for unconstrained fitting of data on a rectangular grid. Unfortunately most univariate monotonic spline techniques do not generalize easily. Figure 1 illustrates the difficulty. The univariate algorithm must not only produce monotonic splines, but do so in a monotonic way: if $\{d_i\}$ and $\{d_i^*\}$ are two sets of data such that $d_i \le d_i^*$ for all i then the corresponding fits must satisfy $f(x) \le f^*(x)$ for all x. Fortunately, variation diminishing splines have this property.

For evaluation efficiency, we use quadratic splines on uniform knots. The rate of convergence of the derivative is faster than might be expected at first, and local minima in the data are reproduced in the spline.

In the following sections of this paper, we analyze a particular form of the univariate variation diminishing spline, establish the multivariate generalization, and review previous techniques used for transistor modeling.

2. Uniform variation diminishing splines. First consider the univariate problem. We wish to approximate a smooth monotone function f, given data at uniformly spaced sample points $t_i^* = (i-1)h$ on the interval $[0,1]$ where $h = 1/(n-1)$ and $1 \le i \le n$. Since in our application only a C^1 approximation is needed, we elect to use quadratic splines. Take knots $t_i = (i-2.5)h$ midway between the sample points t_{i-2}^* and t_{i-1}^*. (These are chosen so that $t_i^* = (t_{i+1} + t_{i-1})/2$, as is required for variation diminishing splines.) Using the data as B-spline coefficients gives the variation diminishing spline [18]

$$S(x) = \sum f(t_i^*) B_i(x).$$

(See [6] for the definition of $\{B_i\}$, which are written as $\{B_{i,3,t}\}$ in that reference.) Because of the local support of the B-splines, if $t_i \le x \le t_{i+1}$ then $S(x)$ depends only on

*Received by the editors August 31, 1984, and in revised form March 10, 1985. This paper was typeset at AT&T Bell Laboratories on August 13, 1985.

†AT&T Bell Laboratories, Murray Hill, New Jersey 07974. Electronic mail: research!ehg or ehg@btl.csnet.

‡AT&T Bell Laboratories, Murray Hill, New Jersey 07974. Present address: Computer Science Department, Duke University, Durham, North Carolina 27706.

f at t_{i-2}^*, t_{i-1}^*, and t_i^*. In the trivial case $n=1$, take the constant function $S(x)=f(t_1^*)$.

Note that we depart from the customary definition by not using multiple knots at the endpoints. We thereby obtain B-splines that are all identical up to translation

$$B_i(x)=B_0(x-ih)$$

and avoid introducing an irregular sample point near the boundary. But the definition of $S(x)$ for $x<h/2$ refers to $f(t_0^*)$, an imaginary sample outside $[0,1]$ indicated by the dotted circle in Figure 2. We implicitly estimate this by linear extrapolation from $f(t_1^*)$ and $f(t_2^*)$. This implies that for $x<h/2$ the spline reduces to a linear function. Here and in the following, we only discuss the left boundary and implicitly treat the right boundary symmetrically.

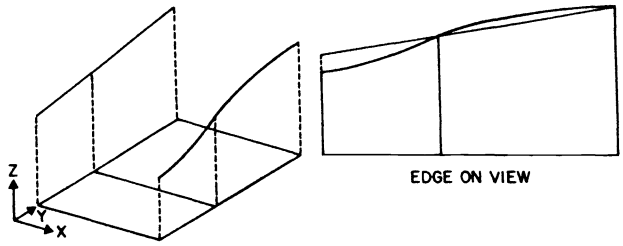


FIG. 1

Monotone interpolation along two grid lines independently may force nonmonotonicity in the orthogonal direction.

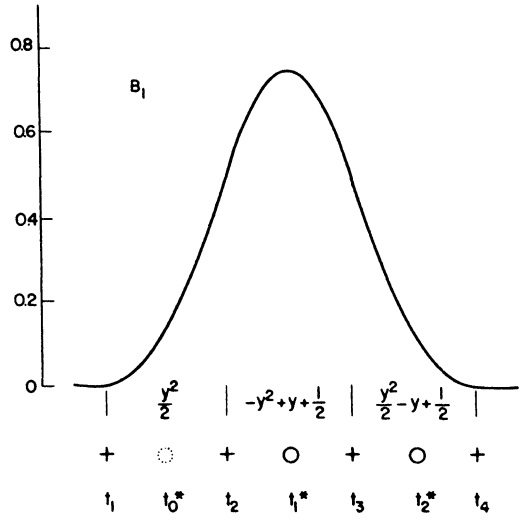


FIG. 2

For knots as indicated by (+), B-splines are translations of the indicated piecewise quadratic, where $y = (x-t_i)/h$ for $t_i \le x \le t_{i+1}$. Function values at the sample points (o) are taken as B-spline coefficients.

Recently we have found that this technique used in computer graphics under the name of "phantom vertices" [4]. Since we were unable to locate any convergence analysis in the literature, we include the following theorem to show that this nonstandard definition retains the properties of variation diminishing quadratic splines defined with the customary multiple knots.

THEOREM 1. *If f'' is Lipschitz continuous then S has the following properties:*

(1) $S \in C^1$;

(2) *if f is linear then $S=f$;*

(3) $S(0)=f(0)$, $S(1)=f(1)$;

(4) *if f is monotone or convex then so is S;*

(5) *if f is quadratic then $S'=f'$;*

(6) $\|f-S\|_{L_\infty[0,1]} = O(h^2)$;

(7) $\|f'-S'\|_{L_\infty[0,1]} = O(h^2)$ *and* $|f'-S'| = O(h)$ *at* 0 *and* 1.

Proof. (1) S is a sum of C^1 functions.

(2) Consider the linear extension f^* of f outside $[0,1]$ and the fit by a variation diminishing spline S^* as customarily defined but with the boundary moved away so that there are equally spaced knots in the neighborhood of 0. It is known that $S^*=f^*$; but the imaginary sample lies on f^*, so $S=S^*$ on $[0,1]$.

(3) Since linear functions are reproduced and since the first three spline coefficients lie on a straight line, S coincides on $[0, h/2]$ with the line through $(0, f(0))$ and $(h, f(h))$.

(4) By the well-known formula for differentiating a spline in B-representation,

$$S' = \sum_i \frac{f(t_i^*)-f(t_{i-1}^*)}{h} B_{i,2},$$

where $\{B_{i,2}\}$ are the piecewise linear B-splines. In other words S' is the piecewise linear interpolant of the first difference of the discrete data, where $(f(t_i^*)-f(t_{i-1}^*))/h$ is associated with t_{i+1}. But if f is monotone, these differences must all have the same sign and hence S must be monotone. If f is convex, these differences are increasing and hence S must be convex.

(5) If f is quadratic then $(f(t_i^*)-f(t_{i-1}^*))/h = f'(t_{i+1})$ and f' is linear. Apply the proof of (4).

(6) Let f^* be a C^1 extension of f to $\mathbb{R}$. The customary variation diminishing spline approximation to f^* is known to be $O(h^2)$ and differs from S only on $[0,h/2]$, where the coefficients of B_0 differ by a quantity that is also $O(h^2)$.

(7) Define

$$S^\Delta = \sum_i f'(t_{i+1}) B_{i,2}$$

and consider

$$|S'(x)-f'(x)| \le |S'(x)-S^\Delta(x)| + |S^\Delta(x)-f'(x)|.$$

If x is in the interior, the first term on the right hand side is $O(h^2)$ because that is the error in a centered difference approximation to f' and because $0 \le B_i \le 1$. The second term is also $O(h^2)$ since it is just the error in local linear interpolation. At the boundary, S' is just a one-sided difference approximation. □

More precise estimates are possible. Theorem 11 of [18] shows that

$$S(x)-f(x) \approx \frac{h^2}{8} f''(x)$$

in the interior. At the boundary, the error introduced by the linear extrapolation is about half that, but of opposite sign.

The restriction to a uniform mesh saves a factor of eight in execution time over efficient general spline codes [6] with $k=3$ and leads to improved convergence of the derivative. Note that higher order splines would not give higher order convergence to f, though they of course would give more continuous derivatives. A more complicated code could handle multiple endpoint knots without much loss of efficiency. A minor advantage of variation diminishing splines over other techniques used previously in simulation modeling is that no preprocessing of the data is required.

As the name implies, variation diminishing splines introduce no minima not already present in the data. For the quadratic spline, it is also possible to show that any discrete local minimum in the data has a corresponding local minimum in the spline and that the difference between the location of the minimum of the spline and that of the function underlying the data is $O(h^2)$.

THEOREM 2. *The point t_i^* is a discrete minimum of the data if and only if the variation diminishing spline S has a strict local minimum $\hat{x}$ in the interval $[t_{i+1}, t_{i+2}]$. Note that S is a quadratic on that interval, and hence such an $\hat{x}$ is unique.*

Proof. As shown in the proof of Theorem 1, S' is the piecewise linear interpolant of the first difference of the discrete data. But the definition of discrete minimum implies that $a_i - a_{i-1}$ is strictly negative and $a_{i+1} - a_i$ is strictly positive, so at some point $\hat{x}$ between t_i and t_{i+1} S' must cross zero, increasing from negative to positive. This implies $\hat{x}$ is a local minimum. □

The method of the proof illustrates why cubic variation diminishing splines could have fewer minima than appear in the data. The derivative would be a quadratic spline with first differences of the data as coefficients. But if one difference is slightly positive with two strongly negative neighbors, the derivative might not cross zero. Cubic interpolatory splines, on the other hand, are infamous for often having more local minima than the data. Polynomial interpolants are even worse.

THEOREM 3. *Assume that f' is Lipschitz continuous and let x^* be a strict local minimum of f. Then for all sufficiently small h the spline S_h has a local minimum $\hat{x}$ with* $|\hat{x} - x^*| = O(h^2)$.

Proof. Define

$$S^{\Delta} = \sum_i f'(t_{i+1}) B_{i,2}$$

and consider

$$|S'(x) - f'(x)| \leq |S'(x) - S^{\Delta}(x)| + |S^{\Delta}(x) - f'(x)|.$$

The first term on the right hand side is $O(h^2)$ because that is the error in a centered difference approximation to f' and because $0 \leq B_i \leq 1$. The second term is also $O(h^2)$ since it is just the error in local linear interpolation.

Since x^* is a strict local minimum, there is an ϵ for which $f''(x^*) > \epsilon > 0$. For sufficiently small h, the previous theorem ensures that S will have a local minimum $\hat{x}$. A Taylor series expansion for f' gives

$$f'(\hat{x}) = f'(x^*) + (\hat{x} - x^*) f''(x^*) + O(h^2).$$

But the left hand side is $O(h^2)$ and $f'(x^*) = 0$, so $\hat{x} - x^*$ must be $O(h^2)$. □

These properties of reproducing local minima have been used by B. Bosacchi of the AT&T Technologies Engineering Research Center in Princeton for the spectral analysis of multilayer thin films.

3. Tensor product generalization. Any linear univariate approximation process can be extended to several variables through the use of tensor products; details of the computational method are described in [14]. For the variation diminishing spline, a two-dimensional tensor variant with the same knots in each variable is given by

$$S(x,y) = \sum_{ij} f(t_i^*, t_j^*) B_i(x) B_j(y).$$

(Of course, in practice we use different numbers of knots in different variables; we have simplified here only to keep down the otherwise bewildering mass of indices.)

COROLLARY 4. *If second derivatives of f are continuous then $\|f-S\| = O(h^2)$, where h is the larger of the sample spacings in x and y.*

Proof. By the triangle inequality,

$$|f(x,y)-S(x,y)| \le |f(x,y)-\sum_j f(x,t_j^*)B_j(y)| + \sum_j |f(x,t_j^*)-\sum_i f(t_i^*,t_j^*)B_i(x)|\, B_j(y).$$

Apply Theorem 1(6) and $\sum B_j(y) = 1$. □

For more general results along this line, see [19].

Define a bivariate function f to be monotone increasing if $x \le x^*$ and $y \le y^*$ implies $f(x,y) \le f(x^*,y^*)$.

COROLLARY 5. *If f is monotone then so is S.*

Proof. By using the monotonicity of univariate variation diminishing splines and the linearity of the definition in the bivariate case, it is easy to see that $S(x,y) \le S(x^*,y) \le S(x^*,y^*)$. □

The extension to three variables is immediate:

$$S(x,y,z) = \sum_{ijk} f(t_i^*, t_j^*, t_k^*) B_i(x) B_j(y) B_k(z).$$

The computational costs of a tensor spline in p dimensions is $O(\prod n_i)$ space if there are n_i sample points for the ith variable and $O(3^p)$ time for one evaluation [14].

4. Experience. Designers of electronic circuits have come to rely heavily on simulation programs. The transistor model used in these simulations is a dominant factor in the speed of the program (and hence the size of the circuits that can be analyzed) and in the accuracy of the computed results. Variation diminishing splines play a useful role here. We will digress briefly to describe the application before discussing the performance of our implementation.

Circuit simulators are based on Kirchhoff's current and voltage laws supplemented by constitutive relations governing the behavior of individual devices, such as transistors. The global circuit equations can be assembled from individual devices using localized Kirchhoff and constitutive relations [11]. In particular, a MOS transistor can be viewed as a device with four terminals: source, gate, drain, and bulk; let u_s, u_g, u_d, and u_b be the corresponding terminal voltages relative to ground. (See Figure 3.) The transistor has three independent currents since Kirchhoff's current law implies that the fourth current is the negative sum of the other three.

If the switching time for a transistor is assumed to be short compared to interesting circuit times then an appropriate transient transistor model is the so-called quasi-static model given by

$$i = \frac{d}{dt} q(v_{ds}, v_{gs}, v_{bs}) + f(v_{ds}, v_{gs}, v_{bs})$$

where i represents the three independent currents, q represents the charges, f represents the steady-state (static) currents, and $v_{ds} \equiv u_d - u_s$, $v_{gs} \equiv u_g - u_s$, $v_{bs} \equiv u_b - u_s$ are the three independent voltages. The voltages $u_{\cdot}$ and the currents i are functions of time t. If there are no bulk leakage currents, the transistor model can be simplified to

$$\begin{pmatrix} i_s \\ i_d \\ i_g \end{pmatrix} = \frac{d}{dt} q(v_{ds}, v_{gs}, v_{bs}) + \begin{pmatrix} -f(v_{ds}, v_{gs}, v_{bs}) \\ f(v_{ds}, v_{gs}, v_{bs}) \\ 0 \end{pmatrix}$$

where now f is scalar. We assume this last form of the transistor model, but the reintroduction of bulk leakage currents only requires approximating two additional functions.

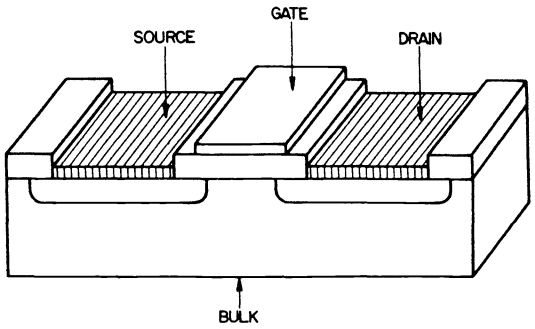


FIG. 3

A MOS transistor.

Sample currents can be measured experimentally or computed by numerical or analytical approximate solution of the partial differential equations describing the behavior of the transistor. Data can be most readily obtained if the sample points lie on a uniform rectangular grid. In the normal operating region, current increases smoothly and monotonically with voltage, which makes the Newton iteration inside the simulation program more robust.

We have implemented the variation diminishing spline algorithm in FORTRAN and found that a single evaluation for $p = 3$ dimensions of S and its three first partial derivatives takes 37 microseconds on a Cray-1A using the CFT 1.10 compiler, including subroutine call overhead. In the transient circuit simulation application, for each set of applied transistor voltages, we need the steady-state current and three charges. By running these four evaluations together, the cost per evaluation drops to 30 microseconds. This is comparable to the cost of compact analytical models. A version of the code is included in release 3 of the PORT library.

Since S is linear near the endpoints, there is a natural C^1 linear extension to $\mathbb{R}$. This is often an excellent approximation in transistor modeling and allows the Newton iteration in the circuit simulator to temporarily step outside the physically realizable region.

Existing experimental instruments and device simulation programs produce data on a uniform grid. If general grids could be used, coordinate transformations such

as square root in the v_{gs} variable for transistor models would reduce somewhat the number of coefficients needed. (We would still use uniformly spaced knots, but in the transformed variable.)

Figure 4 illustrates a typical spline model compared with input data. In this application, the grid consisted of 7 samples in the v_{ds} and v_{gs} directions and 4 in v_{bs}, giving a accuracy of 2%. For $v_{ds}<0$, the symmetric extension $f(-x)=-f(x)$ is used; otherwise, the linear extension described above is applied. This model is for a

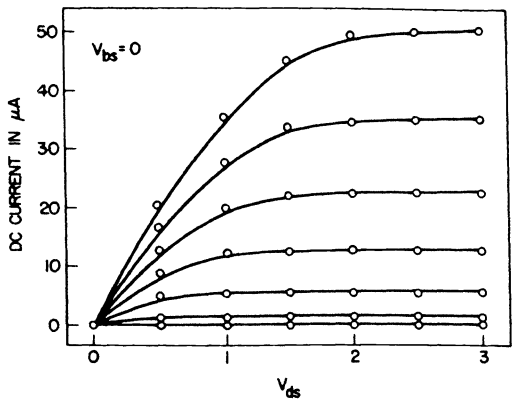


FIG. 4

These slices of a tensor variation diminishing spline and corresponding data values illustrate the qualities needed for simulation models: modest accuracy but guaranteed monotonicity.

transistor of specific length and width, and for a typical circuit simulation perhaps a dozen models would be required. Sometimes the width can be treated as simply a scale parameter and fewer models are needed. The principal remaining difficulty for this application is in the "subthreshold region" of operation of the transistor; in that case, where very little current flows, there is an exponential voltage dependence that is not modeled by the spline. Hence the present model is better suited for digital than for analog circuits. We are attempting to overcome this by a C^1 transformation of the current before fitting by the spline.

In general we have found this approach quite satisfactory and depend exclusively on splines for element models in our circuit simulator *CAzM*[11]. In macromodeling, where the steady-state behavior is preanalyzed for a collection of transistors forming a logic gate, the splines have been especially valuable.

5. Previous models. A brief survey of existing techniques for transistor modeling in circuit simulation may help put our work in perspective.

Most simulators [17] have used analytic models. These are derived from approximate solutions of the partial differential equations describing the behavior of the transistor. See chapter 10 of [24]. The model may be as simple as

$$f(v_{ds},v_{gs}) = \begin{cases} 0 & \text{for } v_{gs} < c_0, \\ c_1 v_{ds}(v_{gs}-c_0) & \text{for } c_0 < v_{gs} < v_{ds}, \\ c_1(v_{gs}-c_0)^2 & \text{for } v_{ds} < v_{gs}, \end{cases}$$

or may require a program of several hundred lines and involve iterative solutions to nonlinear equations. (The best model would solve the partial differential equations for each set of conditions but, even with the fastest methods described in [1, 12], that is vastly too expensive for nontrivial circuits.) A basic advantage of the analytic model is that parameters such as device geometry and temperature appear explicitly and therefore can be varied conveniently. The disadvantages are that new approximations may need to be derived as smaller devices are fabricated, various empirical factors need to be estimated, and as models get more complicated they become rather expensive to compute. A general purpose form is also important for macromodeling, in which several transistors are clumped together and a simple model for, say, a logic gate is built.

These disadvantages have caused increasing attention to be given to table look-up schemes. For an accurate model of the current flowing through a MOSFET, there are at least three important variables, called v_{ds}, v_{gs}, and v_{bs}, representing the difference in potential between the four terminals on the device. The simplest scheme is to tabulate, say, 20^3 values of the current for 20 independent settings of each of the variables. In a compromise to save storage, [9] uses nested tables that yield approximations of the form

$$f(v_{ds}, v_{gs}, v_{bs}) = T_2(v_{ds}, v_{gT}),$$
$$v_{gT} = v_{gs} - T_1(v_{bs}).$$

Nested tables with quadratic interpolation are used by [7], possibly with embellishments to compensate for electrostatic feedback. This idea leads to substantial savings when it is applicable, but in contrast to the three-dimensional tables, there is no guarantee that enough accuracy can be obtained by just increasing the number of sample points.

Currents increase smoothly with voltages on a transistor. If the model does not reflect this, experience shows that the Newton iteration to solve the nonlinear equations characterizing the circuit often fails to converge. In an inadequately tested analytic model, such failures may be seen at the boundary between two regions on which different approximations are used. In the crudest table look-up models, piecewise constant or linear interpolation is used, introducing singularities at each data point. When interpolatory splines are used, monotonic input data can lead to nonmonotonic splines. In the more sophisticated models, such as [21, 22], univariate monotone piecewise cubic interpolation has been used.

Our own method, which we use in the circuit simulation program *CAzM* and which has been adapted in [23] for timing simulation, guarantees smooth monotonic fitting at approximately the same cost to evaluate as the existing compact analytic model [16]. For generality we use a full three-dimensional table, and find that in our applications the 200 to 1000 entries per transistor type do not cause storage difficulties. We use uniformly spaced knots for simplicity and speed.

If nonuniform knots are insisted upon then a piecewise polynomial representation may be preferred. Hence [2] focuses on compressing the piecewise polynomial coefficients, which would otherwise use 64 times as much storage as the B-spline representation. We have not observed the great improvement in accuracy claimed for nonuniform knots.

For univariate monotone fitting, see [13, 20] and papers referenced there. Recently Beatson and Ziegler [5] and Carlson and Fritsch [8] have proposed methods for monotone interpolation on bivariate grids. An extension to three variables may

be possible, but is not immediate. Except for maintaining special properties like $f(0) = 0$, interpolation is of little intrinsic value in our application. Although easy to achieve and yielding the optimal order of convergence in ordinary spline fitting, unnecessary interpolation constraints may hinder progress in monotone fitting. Tensor product variation diminishing splines are popular in computer-aided design, as described in [3], but that has had surprisingly little influence in data fitting. Two Chinese papers, [10] and [15], are the only ones we are aware of using tensor variation diminishing splines, but they consider somewhat different problems.

Acknowledgments. We thank Prasad Subramaniam for suggestions that improved the speed of the code and Sally Liu and Bob Rennick for discussions of modeling near and below threshold.

REFERENCES

[1] R. E. BANK, D. J. ROSE, AND W. FICHTNER, *Numerical methods for semiconductor device simulation*, IEEE Transactions on Electron Devices ED-30, pp. 1031-1041 (1983).

[2] J. BARBY, J. VLACH, AND K. SINGHAL, *Polynomial splines for FET models*, Proceedings of the International Symposium on Circuits and Systems, pp. 206-209 (1983).

[3] R. E. BARNHILL AND R. F. RIESENFELD, *Computer Aided Geometric Design*, Academic Press, New York (1974).

[4] R. H. BARTELS, J. C. BEATTY, AND B. A. BARSKY, *An introduction to the use of splines in computer graphics*, Univ. Waterloo TR CS-83-09, UC Berkeley TR UCB/CSD 83/136 (1985).

[5] R. BEATSON AND Z. ZIEGLER, *Monotonicity preserving surface interpolation*, SIAM Journal on Numerical Analysis 22, pp. 401-411 (1985).

[6] C. DE BOOR, *A Practical Guide to Splines*, Springer-Verlag, Berlin (1978).

[7] J. BURNS, R. NEWTON, AND D. PEDERSON, *Active device table look-up models for circuit simulation*, Proceedings of the International Symposium on Circuits and Systems, pp. 250-253 (1983).

[8] R. E. CARLSON AND F. N. FRITSCH, *Monotone piecewise bicubic interpolation*, SIAM Journal on Numerical Analysis 22, pp. 386-400 (1985).

[9] B. R. CHAWLA, H. K. GUMMEL, AND P. KOZAK, *MOTIS — an MOS timing simulator*, IEEE Transactions on Circuits and Systems CAS-22, pp. 901-910 (1975).

[10] H. L. CHEN, *On surface fitting and shape-preserving approximation with discrete data (Chinese)*, Acta Mathematica Sinica 24, pp. 161-176, MR83f:65015 (1981).

[11] W. M. COUGHRAN, JR., E. H. GROSSE, AND D. J. ROSE, *CAzM: a circuit-analyzer with macromodeling*, IEEE Transactions on Electron Devices ED-30, pp. 1207-1213 (1983).

[12] W. FICHTNER, D. J. ROSE, AND R. E. BANK, *Semiconductor device simulation*, IEEE Transactions on Electron Devices ED-30, pp. 1018-1030 (1983).

[13] F. FRITSCH AND R. CARLSON, *Monotone piecewise cubic interpolation*, SIAM Journal on Numerical Analysis 17, pp. 238-246 (1980).

[14] E. H. GROSSE, *Tensor spline approximation*, Linear Algebra and Its Applications 34, pp. 29-41 (1980).

[15] Y. S. HU, *Some results on the VD approximation of tensor products and convex interpolation (Chinese)*, Acta Mathematica Sinica 24, pp. 857-864, MR84a:41009 (1981).

[16] S. LIU AND L. NAGEL, *Small-signal MOSFET models for analog circuit design*, IEEE Journal of Solid-State Circuits SC-17, pp. 983-998 (1982).

[17] L. W. NAGEL, *SPICE2: a computer program to simulate semiconductor circuits*, University of Calfornia Berkeley Electronics Research Laboratory Memorandum ERL-M520 (1975).

[18] I. J. SCHOENBERG, *On spline functions*, pp. 255-291 in Inequalities, ed. O. Shisha, Academic Press, New York (1967).

[19] M. SCHULTZ, *L^∞-multivariate approximation theory,* SIAM Journal on Numerical Analysis 6, pp. 161-183 (1969).

[20] L. SCHUMAKER, *On shape preserving quadratic spline interpolation,* SIAM Journal on Numerical Analysis 20, pp. 854-864 (1983).

[21] T. SHIMA, H. YAMADA, AND R. DANG, *Table look-up MOSFET modeling system using a 2-D device simulator and monotonic piecewise cubic interpolation,* IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems CAD-2, pp. 121-126 (1983).

[22] T. SHIMA, H. YAMADA, AND R. DANG, *Device/circuit simulator integration technique for MOSFET dircuits,* Proceedings of the International Symposium on Circuits and Systems, pp. 234-237 (1983).

[23] P. SUBRAMANIAM, *Table models for timing simulation,* pp. 310-314 in Proceedings of the Custom Integrated Circuit Conference, Rochester, New York (1984).

[24] S. SZE, *Physics of Semiconductor Devices,* Wiley-Interscience, New York (1981). 2nd edition.

SIAM J. SCI. STAT. COMPUT.
Vol. 7, No. 3, July 1986

THE IMPORTANCE OF SCALING FOR THE HERMITE BICUBIC COLLOCATION EQUATIONS*

WAYNE R. DYKSEN† AND JOHN R. RICE†

Abstract. It is well known that improper scaling of linear equations can result in catastrophic loss of accuracy from Gauss elimination. The scaling process is not well understood and the commonly used "scaling rules" can fail. We study the scaling problem for the linear equations that arise from solving elliptic partial differential equations by collocation using Hermite bicubics. We present an a priori scaling rule that is effective but not foolproof. We conclude that one should use scaled partial pivoting for such equations. We also explore the relationship between the ordering used during Gauss elimination and the underlying geometry of the elliptic problem; we conjecture that this ordering must maintain the geometric integrity of the problem in order to avoid severe round-off problems.

Key words. scaling, Hermite collocation, elliptic problems

AMS(MOS) subject classifications. 65F05, 65F35, 65G05, 65N35

1. Introduction. It is well known that improper scaling of linear equations often results in a catastrophic loss of accuracy from Gauss elimination. Unfortunately, the scaling process is not well understood and the most commonly used "scaling rules" can fail. Textbooks usually choose one of three courses: 1) say that the linear equations should be "properly scaled" and ignore the issue (Dongarra et al. [3]); 2) give some rules for scaling and then warn that they are not infallible (Rice [11]); or 3) present scaled partial pivoting as the proper version of Gauss elimination (Conte and de Boor [2]). A few books combine these (Rice [13]).

We report here on an experimental study of the scaling problem for the linear systems that arise from solving elliptic partial differential equations using Hermite bicubic collocation. An attractive feature of collocation is that it applies easily to general partial differential equations with general boundary conditions. However, the system of linear equations obtained from Hermite bicubic collocation does not possess any special properties such as being positive definite and, as a result, it is most often solved using simple band Gauss elimination. This study demonstrates that it is essential to scale the Hermite bicubic collocation equations; that is, if some type of scaling is not used, then the accumulated effects of round-off dominate the computations. We recommend using both a particular a priori scaling of the equations together with scaled partial pivoting. However, since we cannot formulate a completely reliable a priori scaling rule for these equations which requires less computation than what scaling adds to scaled partial pivoting, we conclude that one should always use scaled partial pivoting. Moreover, we conjecture that the ordering used during Gauss elimination must preserve the underlying geometry of an elliptic problem.

We believe that this conclusion is applicable to other finite element methods. Specifically, collocation using Hermite quintics is an attractive method whose practical value is yet to be explored. It will be a formidable task to determine good a priori scaling factors for this method applied to general problems on general domains. It is likely that the only reliable approach for it is to use software which implements scaled partial pivoting.

* Received by the editors September 6, 1984, and in revised form December 15, 1984. This work was supported in part by the U.S. Department of Energy under contract DE-AC02-81ER10997.

† Department of Computer Sciences, Purdue University, West Lafayette, Indiana 47907.

2. Collocation with Hermite bicubics. We consider a second order, linear elliptic problem on a rectangular domain **R** in the form

$$L[u] = au_{xx} + bu_{xy} + cu_{yy} + du_x + eu_y + fu = g, \qquad (x, y) \in \mathbf{R},$$
$$M[u] = \alpha u + \beta u_x + \gamma u_y = \delta, \qquad (x, y) \in \partial\mathbf{R}$$

where a, b, c, d, e, f, g, α, β, γ and δ are given functions of x and y. We choose a positive integer n and subdivide the domain **R** with a tensor product grid containing n^2 rectangles. We then approximate $u(x, y)$ by

$$U(x, y) = \sum_{i=1}^{N} w_i H_i(x, y) \sim u(x, y)$$

where $N = 4(n+1)^2$ and the $H_i(x, y)$ are the Hermite bicubic basis functions formed as the tensor product of the standard one-dimensional Hermite cubics with the grid lines being the knots.

The N unknowns w_i are determined by choosing N distinct points in **R** and collocating the elliptic problem at these points. In particular, $4n^2$ collocation points are placed at the four Gauss points of each of the n^2 grid rectangles since this gives a fourth order discretization error for smooth problems (Houstis [8], Percell and Wheeler [10]). The remaining $4(2n+1)$ collocation points are the two Gauss points of each boundary grid segment plus one at each of the four corners of **R**. Collocating at these points, we obtain the *Hermite bicubic collocation equations*

$$L[U](x_k, y_k) = g(x_k, y_k), \qquad k = 1, \ldots, 4n^2$$
$$M[U](x_k, y_k) = \delta(x_k, y_k), \qquad k = 4n^2+1, \ldots, 4(n+1)^2.$$

The structure of the coefficient matrix of the resulting linear system is determined by the ordering of the collocation points (the equations or rows) and the basis functions (the unknowns or columns). A common finite element ordering is to order the grid rectangles in the natural way from bottom to top, left to right. The collocation points are then numbered corresponding to their containing grid rectangles (see Fig. 4). The Hermite bicubic basis functions are ordered corresponding to their support in a natural way from bottom to top, left to right. The resulting coefficient matrix is somewhat block bi-diagonal (Dyksen [4], Dyksen and Rice [5]).

3. Numerical experiment. We studied this problem using the ELLPACK system (Rice and Boisvert [14]). Its discretization modules P3C1 COLLOCATION and HERMITE COLLOCATION[1] generate the Hermite bicubic collocation equations; HERMITE COLLOCATION scales the equations associated with the boundary conditions (see § 6) whereas P3C1 COLLOCATION does not. The solution modules LINPACK BAND and BAND GE solve the resulting linear system of equations. LINPACK BAND uses the LINPACK routines SGBFA and SGBSL which do band Gauss elimination with partial pivoting (Dongarra et al. [3]). BAND GE does band Gauss elimination with scaled partial pivoting (Conte and de Boor [2, § 4.3]) using a direct modification of SGBFA and SGBSL. The equations are solved in the order in which they are generated by the discretization modules, namely, the finite element ordering described above.

[1] The module which we refer to here as HERMITE COLLOCATION has subsequently been split into two separate ELLPACK modules, HERMITE COLLOCATION and INTERIOR COLLOCATION, and P3C1 COLLOCATION has been removed from ELLPACK.

We combine these modules to obtain four similar, yet distinct numerical methods. Note that scaling is the only difference between P3C1 COLLOCATION and HERMITE COLLOCATION and between LINPACK BAND and BAND GE. Letting S and U stand for "scaled" and "unscaled", we denote these four methods as follows:

Notation	Numerical method
U/U	P3C1 COLLOCATION/LINPACK BAND
U/S	P3C1 COLLOCATION/BAND GE
S/U	HERMITE COLLOCATION/LINPACK BAND
S/S	HERMITE COLLOCATION/BAND GE

We use a subject population of twenty elliptic problems from the population of Rice et al. [12]; it consists of Problems 2-1, 3-1, 5-1, 6-1, 9-2, 10-2, 10-3, 12-3, 17-2, 20-2, 22-1, 23-6, 33-1, 35-3, 38-1, 40-1, 50-1, 53-3, 54-2 and 59-1. These twenty problems represent a variety of partial differential operators and boundary conditions. Fifteen problems have Dirichlet boundary conditions, five of which are homogeneous. Fifteen of the domains are the unit square.

Each of the four numerical methods described above are applied to each of the subject population problems using the performance evaluation system of Boisvert et al. [1]. We use $n = 4$, 8, 12, 20 and 29 which involves from 100 to 3364 unknowns w_i. The computations are done on a VAX 11/780 computer with floating point accelerator using the UNIX FORTRAN complier f77. Note that this experiment involves computing 400 solutions of elliptic problems.

4. Performance analysis. We now consider the following hypothesis: *Scaling is essential for numerically solving the Hermite bicubic collocation equations.* To establish this hypothesis, we compare these methods pairwise using simple nonparametric analysis as follows:

Comparison	Interpretation
U/S vs U/U	Solve the unscaled equations with scaled partial pivoting (BAND GE) versus partial pivoting (LINPACK BAND)
S/U vs U/U	The scaled versus unscaled equations solved with partial pivoting
S/S vs S/U	Solve the scaled equations with scaled partial pivoting versus partial pivoting
S/S vs U/S	The scaled versus unscaled equations solved with scaled partial pivoting

The two methods of each pair are ranked on each problem using the maximum error at the grid points. For example, Fig. 1 shows performance graphs of $\log(n+1)$ versus the logarithm of the maximum error at the grid points for two problems. We see that method U/S is dramatically more accurate than method U/U; in fact, method U/U gives results which are so contaminated by round-off that they are totally unacceptable. By contrast, we see that methods S/S, S/U and U/S each give similar, accurate results; that is, the effect of either scaling the collocation equations or using scaled partial pivoting to solve them appears to remedy the problem present in method U/U. Note, however, in Fig. 1 that for Problem 23-6 U/S and S/S give significantly better accuracy for one case. This means that our a priori scaling method was not nearly as good in this case as using scaled partial pivoting. These graphs typify the results obtained for the other problems.

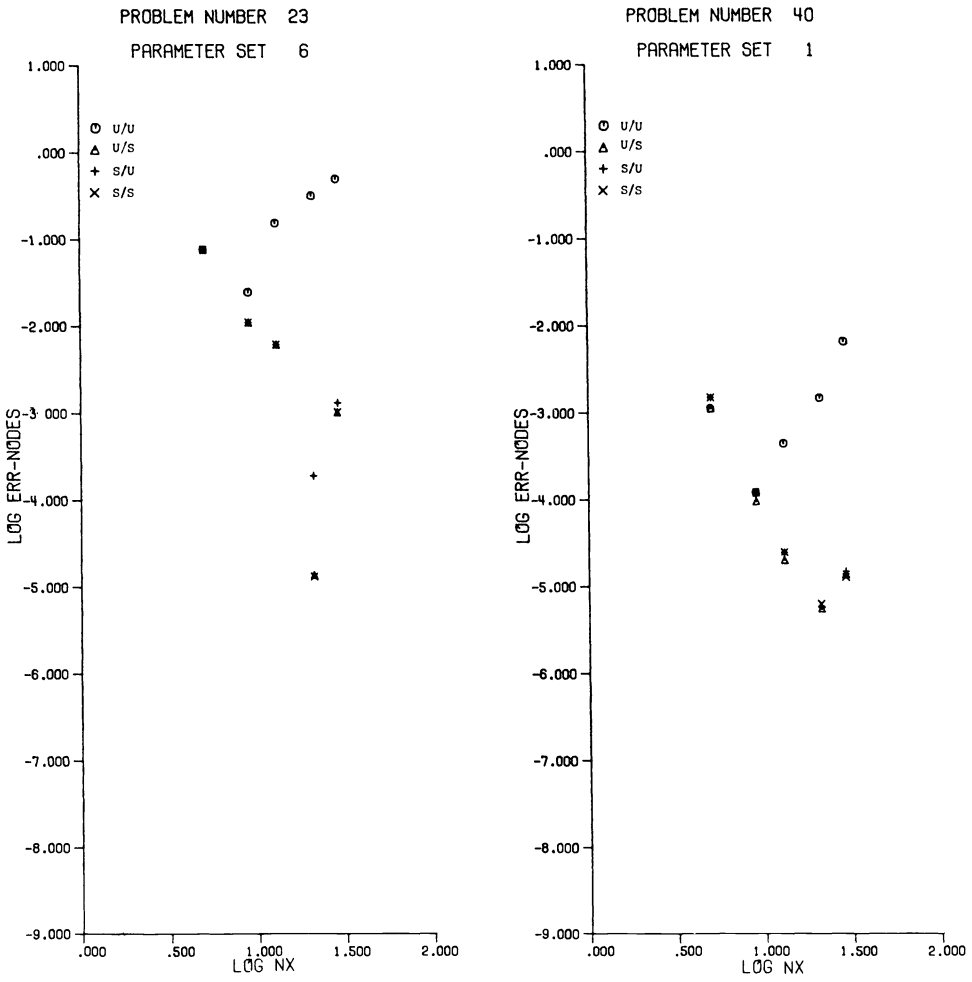


FIG. 1. *Graphs of the logarithm of* $NX = n+1$ *versus the logarithm of the maximum error at the grid points for Problem* 23-6 *which has mixed boundary conditions and for Problem* 40-1 *which has nonhomogeneous Dirichlet boundary conditions.*

We rank each pair of methods on each problem and compute average ranks for four different groups of problems: the ten with nonhomogeneous Dirichlet boundary conditions, the five with homogeneous Dirichlet boundary conditions, the five with mixed boundary conditions, and the entire subject population. An average rank of 1.00 means that the method is always the best whereas 2.00 means that it is always the worst. We obtain confidence levels on the observed differences using the Friedman, Kendall and Babington-Smith test (Hollander and Wolfe [7]). We summarize the results in Tables 1-4.

For example, we see from Table 1 that comparing U/S versus U/U on the entire subject population gives a rank of 1.17 for U/S and a rank of 1.82 for U/U. The difference in rank is significant at the 99% level of confidence.

These experimental results strongly support our initial hypothesis, namely, that scaling is essential for numerically solving the Hermite bicubic collocation equations. We see from Fig. 1 that the results obtained by scaling are significantly more accurate than those obtained by not scaling. Moreover, the data in Tables 1-4 demonstrate that

TABLE 1
Average rank of U/S vs U/U.

Group		Average rank U/S	Average rank U/U	Significance
Dir/Non	(10)	1.00	2.00	99%
Dir/Hom	(5)	1.70	1.30	<80%
Mixed	(5)	1.00	2.00	99%
Combined	(20)	1.17	1.82	99%

TABLE 2
Average rank of S/U vs U/U.

Group		Average rank S/U	Average rank U/U	Significance
Dir/Non	(10)	1.00	2.00	99%
Dir/Hom	(5)	1.60	1.40	<80%
Mixed	(5)	1.00	2.00	97%
Combined	(20)	1.15	1.85	99%

TABLE 3
Average rank of S/S vs S/U.

Group		Average rank S/S	Average rank S/U	Significance
Dir/Non	(10)	1.40	1.60	<80%
Dir/Hom	(5)	1.70	1.30	<80%
Mixed	(5)	1.50	1.50	<80%
Combined	(20)	1.47	1.52	<80%

TABLE 4
Average rank of S/S vs U/S.

Group		Average rank S/S	Average rank U/S	Significance
Dir/Non	(10)	1.50	1.50	<80%
Dir/Hom	(5)	1.60	1.40	<80%
Mixed	(5)	1.20	1.80	80%
Combined	(20)	1.45	1.55	<80%

the observed similarities or differences between methods are themselves statistically significant and not due merely to chance.

Our initial hypothesis can be stated more specifically in terms of the four methods considered here: method S/S is slightly more accurate than methods S/U and U/S which are all very much more accurate than method U/U. We believe that this data supports our hypothesis with a high level of statistical confidence.

Finally, we note from Tables 1 and 2 that problems with homogeneous Dirichlet boundary conditions are a significant special case. In this case, both HERMITE COLLOCATION and P3C1 COLLOCATION eliminate the boundary condition equations from the linear system during the discretization and *before* Gauss elimination. This suggests that the boundary equations might be the key to understanding the severe round-off problems resulting from method U/U.

5. Scaling and the boundary equations. To further study the effects of round-off, we constructed a parameterized elliptic problem, Problem 59, whose solution is a bicubic for which Hermite bicubic collocation gives the exact solution except for round-off. Problem 59-1 is a Poisson problem with nonhomogeneous Dirichlet boundary conditions on the unit square.

Figures 2 and 3 show contour plots of the error for Problem 59-1 using S/U and U/U with $n=8$. Figure 2 shows that if we solve the scaled equations using partial pivoting, then the error is rather randomly distributed and is of the order of machine precision, 10^{-6}. By contrast, we see from Fig. 3 that if the unscaled equations are solved using merely partial pivoting, then the error in the interior of the domain is still on the order of 10^{-6} whereas the error on the boundary is on the order of 10^{-5} and is as large as 10^{-4}. Hence, essentially all of the round-off error occurs on the boundary; this is unexpected since the boundary conditions are Dirichlet and hence should be interpolated exactly. This is further evidence that the boundary equations are the key to understanding the round-off problems.

The relationship between the boundary equations and scaling is geometrical and can be seen by considering the order in which the equations are eliminated during Gauss elimination. Since the equations are associated with the collocation points, we can view the reordering of the equations produced by pivoting as a reordering of the collocation points themselves.

Figure 4 shows a typical example of the order of elimination resulting from solving Problem 59-1 using the unscaled collocation equations. We give the geometric ordering of the collocation equations before Gauss elimination and after Gauss elimination with scaled partial pivoting, partial pivoting and complete pivoting.

Figure 4 shows that the three pivoting strategies differ dramatically in the order in which they eliminate the interior and the boundary equations. Partial pivoting eliminates as many of the interior equations as possible before it must eliminate a boundary equation. Complete pivoting eliminates *all* of the interior equations before eliminating *any* of the boundary equations. Scaled partial pivoting weaves the elimination of the interior and boundary equations together; in fact, the reordering produced by scaled partial pivoting is essentially the Hermite Collorder ordering discussed in Dyksen and Rice [5].

The above phenomena results from the inherent differences in magnitudes of the boundary and interior equations. In a typical elliptic Dirichlet problem, the coefficients of the boundary equations involve values of the basis functions and hence are $O(1)$. The interior equations, however, involve second derivatives of the basis functions and hence are $O(n^2)$. Thus, during simple partial pivoting and complete pivoting, the interior equations are chosen before the boundary equations as often as possible. As a result, the boundary condition information is not used until the last possible moment.

In practice, the lack of scaling using the original ordering results in the two-dimensional analogue of numerically solving an ordinary differential equation from the inside out.

6. Scaling the boundary equations. There are two approaches to scaling the boundary equations. Since the scaling is required only for choosing the pivots, it need not be carried out explicitly, although to do so is a simple way to proceed. Thus, we can scale the boundary equations either *explicitly before* elimination or *implicitly during* elimination.

In order to determine an a priori scaling factor, we consider the Hermite bicubic U in the case in which all of the coefficients w_i are $O(1)$. If the domain is discretized

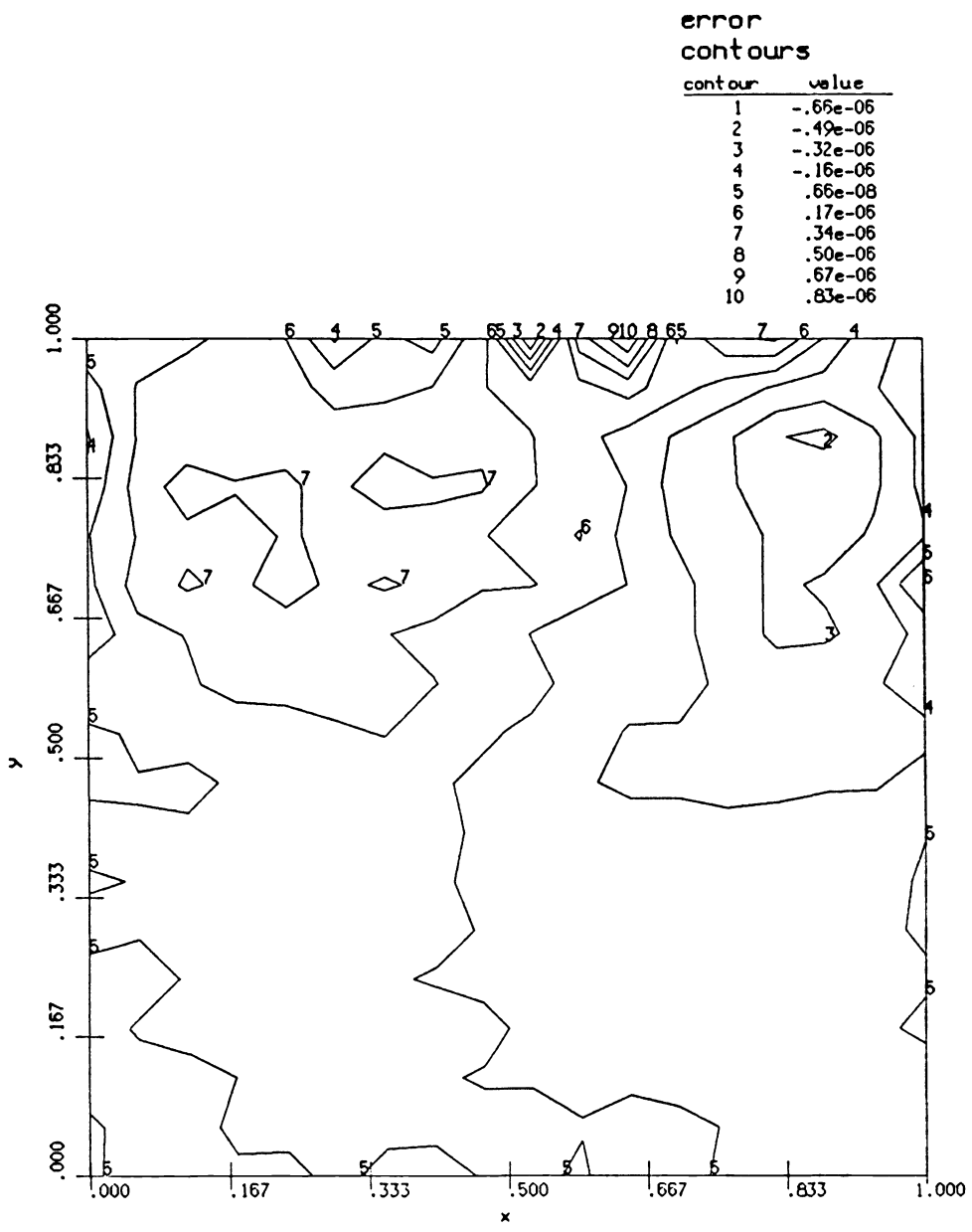


FIG. 2. *Contour plot of the error obtained by solving Problem* 59-1 *using* S/U *with* $n = 8$. *The error is due to round-off and is of the order of* 10^{-6} *which is machine precision.*

with a uniform x and y spacing h_x and h_y, respectively, then a simple computation gives

$$U_{xx} + U_{yy} \sim (1+h_y)(1+1/h_x)/h_x + (1+h_x)(1+1/h_y)/h_y \tag{6.1}$$

and

$$\begin{aligned} U &\sim 1 + h_y + h_x + h_x h_y, \\ U_x &\sim 1/h_x + h_y/h_x + 1 + h_y, \\ U_y &\sim 1/h_y + 1 + h_x/h_y + h_x. \end{aligned} \tag{6.2}$$

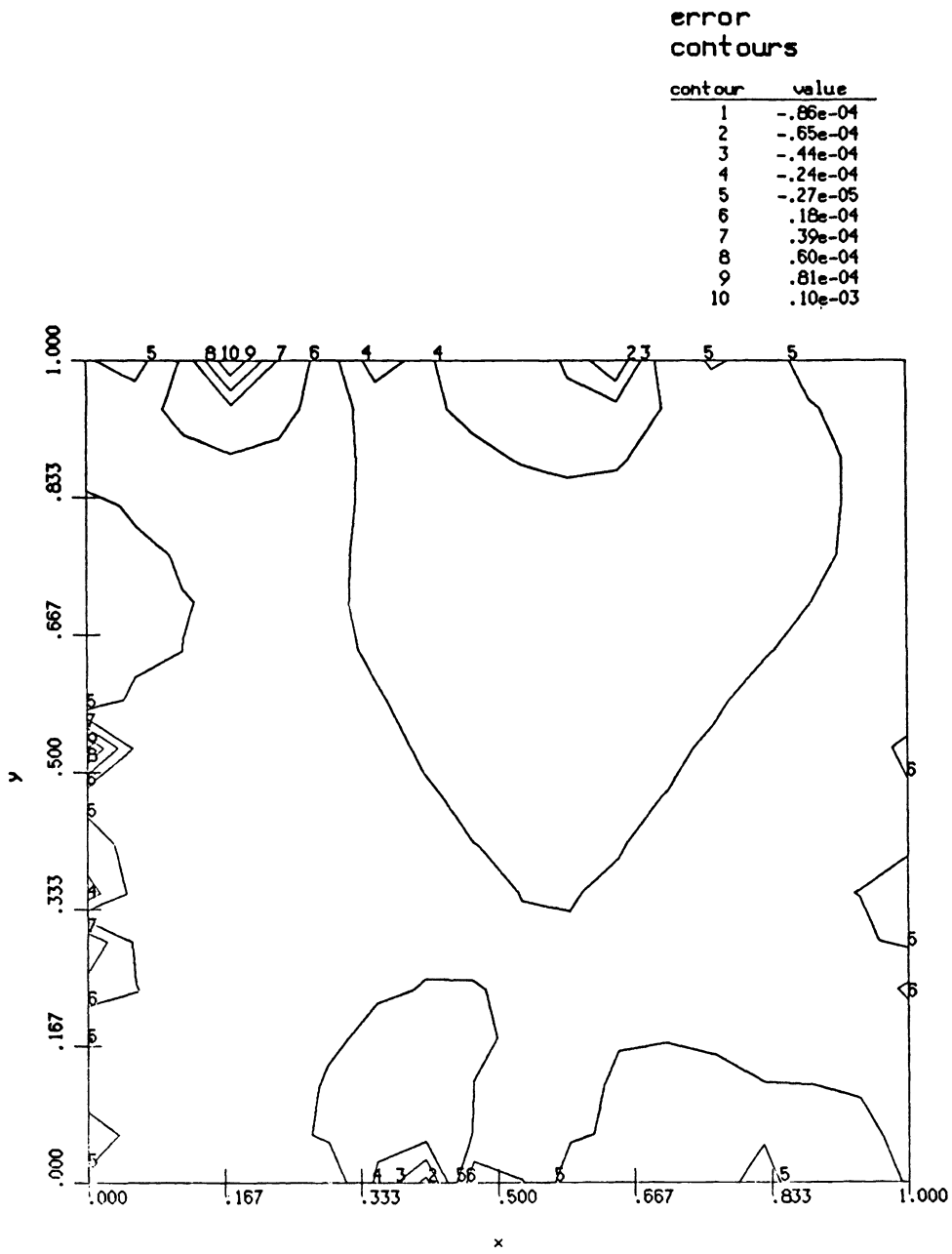


FIG. 3. *Contour plot of the error obtained by solving Problem* 59-1 *using* U/U *with* $n = 8$. *The error in the interior of the domain is on the order of* 10^{-6} *whereas the error on the boundary is on the order of* 10^{-5} *and is as large as* 10^{-4}.

Thus, for the model problem the interior equations look like (6.1) and the boundary equations look like (6.2).

With this in mind, we experimented extensively with many scaling factors, applying them to Problems 2-1, 22-1 and 59-1. We varied n and computed the maximum error at the grid points as well as the condition number of the coefficient matrix using the LINPACK routine SGBCO. For example, Table 5 summarizes the results for Problem

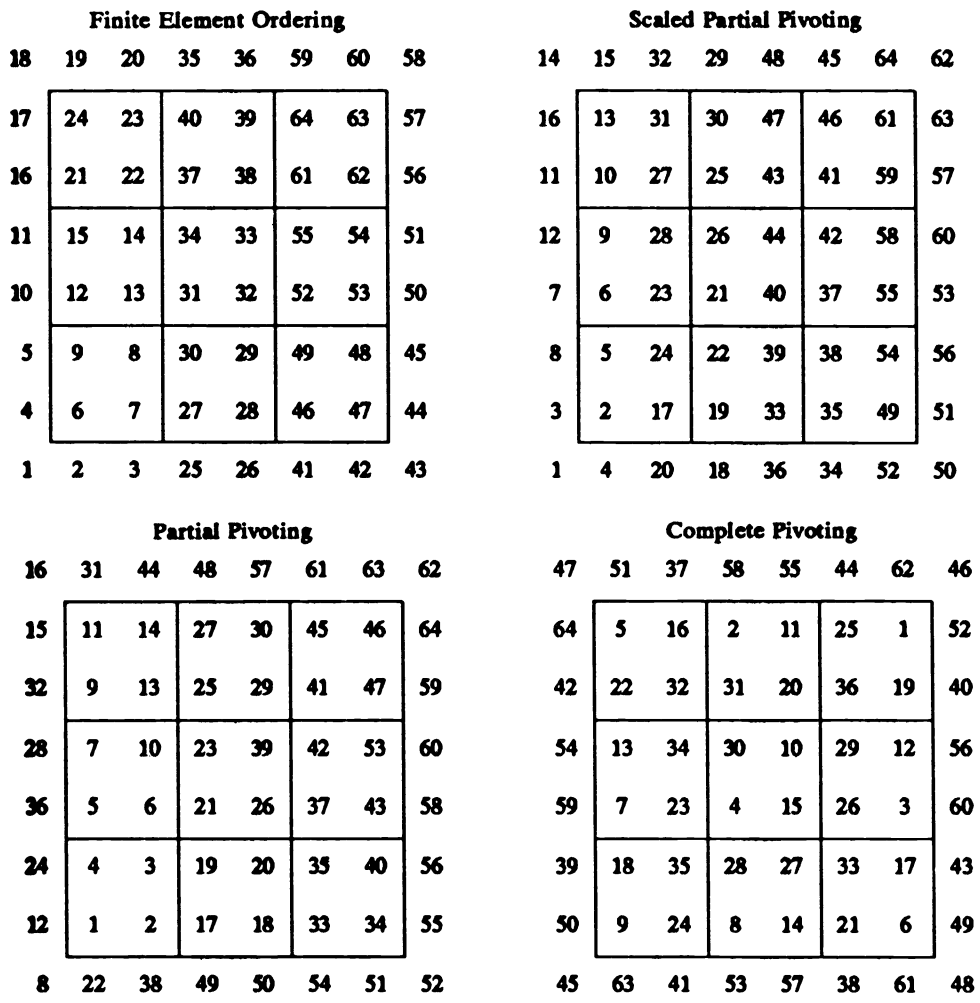


FIG. 4. *The geometric ordering of the collocation equations before Gauss elimination (upper left) and after Gauss elimination using scaled partial pivoting (upper right), partial pivoting (lower left) and complete pivoting (lower right).*

59-1 using the scale factor $1/h_x^2+1/h_y^2$. We see that scaling the boundary equations produces significant changes in both the error and the condition number.

Having experimented with these scaling factors, we propose the L_1 type scaling factor given in (6.1) (Skeel [15], [16], [17]). It has a simple and natural analytical

TABLE 5
The effect of scaling on the condition number and the maximum error for Problem 59-1.

n	Number of unknowns	Condition number scaled	Condition number unscaled	Maximum error scaled	Maximum error unscaled
4	100	$1.4*10^{+4}$	$5.1*10^{+5}$	$9.5*10^{-7}$	$1.1*10^{-5}$
8	324	$5.3*10^{+4}$	$6.9*10^{+6}$	$9.5*10^{-7}$	$8.0*10^{-5}$
16	1156	$1.8*10^{+5}$	$1.0*10^{+8}$	$1.9*10^{-6}$	$7.8*10^{-4}$
22	2116	$3.2*10^{+5}$	$3.5*10^{+8}$	$3.7*10^{-6}$	$1.5*10^{-3}$
28	3364	$4.8*10^{+5}$	$9.1*10^{+8}$	$1.2*10^{-5}$	$2.9*10^{-3}$

basis. Since from (6.2) $U = O(1)$, we multiply a boundary condition equation involving only u by (6.1) to make it the same size as the interior equations. Similarly, since from (6.2) $U_x = O(1/h_x)$ and $U_y = O(1/h_y)$, we scale a boundary equation involving u_x or u_y by the product of h_x or h_y, respectively, and (6.1). This proposed scaling method is in fact used by HERMITE COLLOCATION.

Although the above scaling method works well on a fairly large set of problems, we believe that it is not always practical to scale a priori the boundary equations to make them the same size as the interior equations. The scale factor in (6.1) is derived using a simple model of the coefficients in the elliptic problem. The severe round-off phenomena observed above may occur again for problems in which either the coefficients in the partial differential operator are large or the coefficients in the boundary conditions are small at the collocation points. In such a case one would need to compute either the extreme values of these coefficient functions or perhaps the maximum L_1 norm of the interior equations to scale the boundary equations correctly. As a result, we conjecture that S/S is more reliable than S/U; that is, BAND GE is more reliable than its ancestor LINPACK BAND.

For example, Fig. 5 shows a performance graph for Problem 22-1 which involves an operator with a large coefficient function. In this case, our a priori scaling method is clearly inferior to using scaled partial pivoting (with either the scaled or unscaled collocation equations).

As another example, consider the "scaled" Poisson problem

$$10^k(u_{xx}+u_{yy}) = f, \qquad (x, y) \in \mathbf{R} = [0, 1] \times [0, 1],$$
$$u = g, \qquad (x, y) \in \partial\mathbf{R}$$

where f and g are chosen so that $u = (x^3+(xy)^2+2xy^3+1)/5$. If we vary k and solve this problem using S/S and S/U with $n = 21$ (1764 unknowns), we obtain the results given in Table 6. We again see that scaled partial pivoting is superior to our particular a priori scaling method.

The data in Tables 1–4 do not provide any support for the conjecture that S/S is sometimes more reliable than S/U; the nature of those statistical tests masks this because the advantage of S/S shows up infrequently, only if the discretization error is close to round-off. In particular, there are 23 instances involving 11 problems in which the maximum relative error for either S/S or S/U is less than 10^{-5}. In twelve cases the errors differ by more than 10%, and S/S is more accurate in ten of these cases. For these twelve observations, the Sign test (Hollander and Wolfe, [7]) assures that S/S is more accurate than S/U with 98% confidence (p-value of 0.02).

The comparative work of different scaling methods is easily computed. For a grid of n^2 rectangles, there are $4n^2$ interior equations and $8n+4$ boundary equations. The half bandwidth of the linear system is $4n+7$ and there are at most 16 nonzero entries per interior equation and 8 nonzero entries per boundary equation. The comparative work for three scaling methods is given in Table 7. We see that each method requires much less work than Gauss elimination which is $O((4n+7)^2*4(n+1)^2) = O(n^4)$. Although scaling the boundary equations alone is the least amount of work, it is also probably the least reliable.

The work estimates in Table 7 lead to an important observation. At each stage in scaled partial pivoting, the "scaled" entries below the diagonal are searched for a pivot. In the case of the collocation equations, this involves $4n+7$ multiplies even though there are at most 16 nonzero entries to examine. Clearly it is more efficient for the equations to be scaled during the discretization phase, before these relatively few

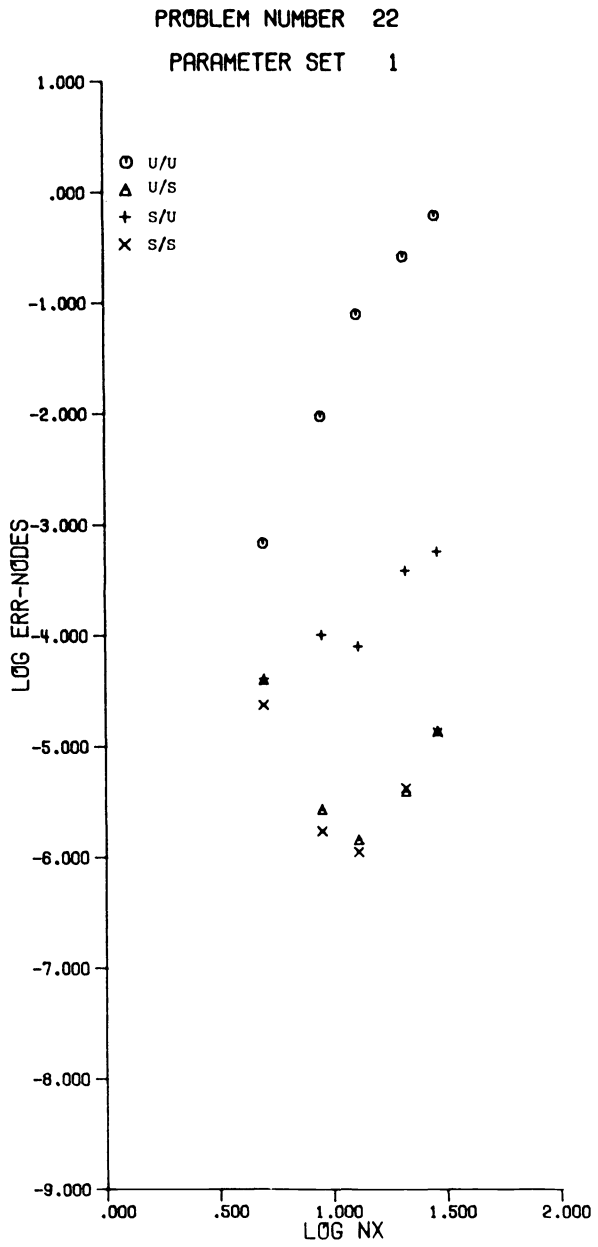


FIG. 5. *Graphs of the logarithm of* $NX = n+1$ *versus the logarithm of the maximum error at the grid points for Problem* 22-1 *which involves an operator of the form* $(wu_x)_x + (wu_y)_y = f$ *where w is large.*

nonzero entries are dispersed throughout the band. The resulting savings is an order of magnitude in the work of scaling. It might be that the overall best choice is to do the scaling of scaled partial pivoting during the discretization and then to use simple partial pivoting during the elimination.

In view of all of the above, we recommend using *both* the a priori scaling method described here along with scaled partial pivoting. Although the a priori scaling method is not foolproof, it is simple to apply. Moreover, neither method of scaling requires

TABLE 6
Maximum errors for the scaled Poisson problem.

k	Maximum error S/S	S/U
0	$3.1*10^{-6}$	$2.6*10^{-6}$
2	$1.4*10^{-6}$	$1.2*10^{-4}$
4	$1.5*10^{-6}$	$1.0*10^{-2}$
6	$6.3*10^{-7}$	$8.5*10^{-1}$
8	$1.7*10^{-6}$	$2.0*10^{+5}$

TABLE 7
Comparative work for three scaling methods.

Scaling method	Work
Only boundary equations during discretization	$O(8*(8n+4)) = O(n)$
All equations during discretization	$O(16*4(n+1)^2) = O(n^2)$
Scaled partial pivoting	$O((4n+7)*4(n+1)^2) = O(n^3)$

any significant extra computation. For example, S/S takes on the average only 5% longer than S/U to solve the collocation equations with $n = 28$ (3364 unknowns).

Finally, we note that it is *not* the case that the Hermite bicubic collocation equations with unscaled boundary equations are inherently ill-scaled. In fact, we have observed that they can be solved accurately *without* scaling and *without* pivoting if one orders the equations and unknowns using the Hermite Collorder ordering given in Dyksen and Rice [5]. Even though this approach seems attractive at first glance, there are a number of reasons why we do not recommend it in practice. First, one would have to implement this ordering which is nontrivial; in many software environments this task would be impractical if not impossible. Secondly, using the Hermite Collorder ordering with no pivoting requires (contrary to intuition) on the average 8% more execution time than using a traditional finite element ordering with scaled partial pivoting (Dyksen and Rice [5, p. 478]). Finally, and most importantly, we conjecture that the phenomena observed in this paper are not particular to Hermite bicubic collocation, but would result with other piecewise polynomial collocation methods. In such settings, both a natural a priori scaling as well as scaled partial pivoting would apply. By contrast, it is not clear that an ordering analogous to Hermite Collorder for which no pivoting is needed can be discovered or even exists. Thus, the striking result that the Hermite bicubic collocation equations can be solved accurately *without* scaling and *without* pivoting is currently only of theoretical interest.

7. Preservation of geometric integrity. The poor scaling of the collocation equations in their original form destroys the relationship between the geometry of the problem and the order of elimination. One hopes that the ellipticity of an elliptic problem should damp out errors, including round-off. However, destroying the geometry of the problem seems to ruin its ellipticity.

As a further example of this phenomenon, we consider the linear equations obtained from Problem 59-1 by using the standard 5-point star discretization modified to include the unscaled Dirichlet boundary equations. As in the case of Hermite bicubic collocation with the Hermite Collorder ordering, these 5-point star equations can be solved to machine precision *without* scaling and *without* pivoting. If the equations are

solved with simple partial pivoting, then round-off dominates the computations as the grid is refined. We see that it is not only obviously inefficient to include the boundary equations haphazardly in the linear system, it is also dangerous. Note that this is done routinely in many finite element programs in structural engineering.

We also generated random row permutations and solved the equations using partial pivoting to see what effect if any this might have on the solution. In summary, we observed that the more the underlying geometry is perturbed, the larger the error becomes. This again suggests that the ordering used during Gauss elimination must maintain the geometric integrity of the elliptic problem. We believe that this is not particular to the 5-point star or Hermite bicubic collocation, and we conjecture that it is true for other numerical methods for elliptic problems.

REFERENCES

[1] R. F. BOISVERT, E. N. HOUSTIS AND J. R. RICE, *A system for performance evaluation of partial differential equation software*, IEEE Trans. Software Engrg., 5 (1979), pp. 418-425.

[2] S. D. CONTE AND C. W. DE BOOR, *Elementary Numerical Analysis*, McGraw-Hill, New York, 1980.

[3] J. J. DONGARRA, C. B. MOLER, J. R. BUNCH AND G. W. STEWART, LINPACK *Users' Guide*, Society for Industrial and Applied Mathematics, 1979.

[4] W. R. DYKSEN, *A new ordering scheme for the Hermite bicubic collocation equations*, Purdue University, Computer Science Department Report CSD-TR 364, May 1, 1981.

[5] W. R. DYKSEN AND J. R. RICE, *A new ordering scheme for the Hermite bicubic collocation equations*, in Elliptic Problem Solvers III, G. Birkhoff and A. Schoenstadt, eds., Academic Press, New York, 1984, pp. 467-480.

[6] W. R. DYKSEN, R. E. LYNCH, J. R. RICE AND E. N. HOUSTIS, *The performance of the collocation and Galerkin methods with Hermite bi-cubics*, SIAM J. Numer. Anal., 21 (1984), pp. 695-715.

[7] M. HOLLANDER AND D. A. WOLFE, *Nonparametric Statistical Methods*, John Wiley, New York, 1975.

[8] E. N. HOUSTIS, *Collocation methods for linear elliptic problems*, BIT, 18 (1978), pp. 301-310.

[9] E. N. HOUSTIS, R. E. LYNCH, J. R. RICE AND T. S. PAPATHEODOROU, *Evaluation of numerical methods for elliptic partial differential equations*, J. Comput. Phys., 27 (1978), pp. 323-350.

[10] P. PERCELL AND M. F. WHEELER, *A C^1 finite collocation method for elliptic problems*, SIAM J. Numer. Anal., 17 (1980), pp. 605-622.

[11] J. R. RICE, *Matrix Computations and Mathematical Software*, McGraw-Hill, New York, 1981.

[12] J. R. RICE, E. N. HOUSTIS AND W. R. DYKSEN, *A population of linear, second order, elliptic partial differential equations on rectangular domains, Parts* 1 *and* 2, Math. Comp., 36 (1981), pp. 475-484.

[13] J. R. RICE, *Numerical Methods: Software and Analysis*, McGraw-Hill, New York, 1985.

[14] J. R. RICE AND R. F. BOISVERT, *Solving Elliptic Problems Using* ELLPACK, Springer-Verlag, New York, 1985.

[15] R. D. SKEEL, *Scaling for numerical stability in Gaussian elimination*, J. Assoc. Comput. Mach., 26 (1979), pp. 494-526.

[16] ———, *Iterative refinement implies numerical stability for Gaussian elimination*, Math. Comp., 35 (1980), pp. 817-832.

[17] ———, *Effect of equilibration on residual size for partial pivoting*, SIAM J. Numer. Anal., 18 (1981), pp. 449-454.

SIAM J. SCI. STAT. COMPUT.
Vol. 7, No. 3, July 1986

NUMERICAL SOLUTION OF NONLINEAR DIFFERENTIAL EQUATIONS WITH ALGEBRAIC CONSTRAINTS II: PRACTICAL IMPLICATIONS*

LINDA PETZOLD† AND PER LÖTSTEDT‡

Abstract. In this paper we investigate the behavior of numerical ODE methods for the solution of systems of differential equations coupled with algebraic constraints. Systems of this form arise frequently in the modelling of problems from physics and engineering; we study some particular examples from fluid dynamics and constrained mechanical systems. We investigate some of the practical difficulties of implementing variable-stepsize backward differentiation formulas for the solution of these equations, showing how to overcome problems of matrix ill-conditioning, and giving convergence tests and error tests which are supported by theory.

Key words. differential-algebraic, constraints, Lagrange multipliers, mechanical systems, high index systems

AMS(MOS) subject classification. 65L05

1. Introduction. In this paper we investigate some of the practical difficulties involved with implementing backward differentiation formulas (BDF) for the solution of differential/algebraic equations (DAE) of the form

$$0 = F_1(x, x', y, t), \qquad 0 = F_2(x, y, t) \tag{1.1}$$

where the initial values of x and y are given at $t = 0$ and $\partial F_1/\partial x'$ is nonsingular. Systems of this form arise frequently in the modelling of engineering problems, for example the simulation of electrical networks and mechanical systems, and the solution of the equations of fluid dynamics. In an earlier paper [1] we showed that for the systems under consideration (certain restrictive assumptions must be placed on (1.1); these assumptions are satisfied in many practical applications) the k-step constant stepsize BDF method converges to order of accuracy $O(h^k)$, where h is the stepsize. Here we are concerned with the practical difficulties such as varying the stepsize and dealing with ill-conditioned matrices which arise in implementing BDF methods for the solution of (1.1). Recently, similar systems of equations have been studied also by Brenan [2].

The idea of using BDF methods for systems of this type was introduced by Gear [3] and consists of replacing x' in (1.1) by a difference approximation, and then solving the resulting equations for approximations to x and y. Let $F = (F_1, F_2)^T$. To solve (1.1) numerically at t_n by the k-step BDF, we replace $x'(t_n)$ by $\rho x_n / h$ where ρ is the difference operator defined by

$$\rho x_n = \sum_{i=0}^{k} \alpha_i x_{n-i}, \tag{1.2}$$

$h = t_n - t_{n-1}$ and α_i are the BDF coefficients, to obtain the system of nonlinear equations

$$F\left(x_n, \frac{\rho x_n}{h}, y_n, t_n\right) = 0. \tag{1.3}$$

* Received by the editors July 19, 1984, and in revised form April 3, 1985.

† Computing Research and Development Division, Lawrence Livermore National Laboratory, Livermore, California 94550. The work of this author was supported by the U.S. Department of Energy Office of Basic Energy Sciences.

‡ Aerospace Division, SAAB-Scania, S-58188 Linköping, Sweden. The work of this author was supported in part by the National Swedish Board for Technical Development when the author was at the Department of Numerical Analysis and Computing Science, Royal Institute of Technology, S-10044 Stockholm, Sweden.

This system has a unique solution [4] if the inverse of the scaled Jacobian matrix

$$hJ_n = \begin{pmatrix} \alpha_0 \dfrac{\partial F_1}{\partial x'} + h\dfrac{\partial F_1}{\partial x} & h\dfrac{\partial F_1}{\partial y} \\ h\dfrac{\partial F_2}{\partial x} & h\dfrac{\partial F_2}{\partial y} \end{pmatrix} \tag{1.4}$$

exists.

The types of equations that we consider here arise commonly in several areas of application, which are discussed in more detail in [5]. For example, the flow of an incompressible, viscous fluid is described by the Navier-Stokes equations

$$\frac{\partial u}{\partial t} + (u \cdot \nabla)u = -\nabla p + \gamma \nabla^2 u, \tag{1.5a}$$

$$\nabla \cdot u = 0 \tag{1.5b}$$

where u is the velocity in two or three dimensions, p is the pressure, and γ is the kinematic viscosity. After spatial discretization of (1.5) with a finite difference or finite element method, the vectors U and P, approximating u and p, satisfy [6]

$$M\dot{U} + (K + N(U))U + CP = f(U, P), \tag{1.6a}$$

$$C^T U = 0 \tag{1.6b}$$

which has the form (1.1). The mass matrix M is the identity matrix (finite differences) or a symmetric positive definite matrix (finite elements). The discretization of the operator ∇ is C and the forcing function f emanates from the boundary conditions.

Another application which fits into this general framework is the simulation of mechanical systems of rigid bodies interconnected directly by joints or via other components such as springs and dampers. The vector q of coordinates of the bodies satisfies the following equations [7]

$$M(q)q'' = f(q, q', t) + G(q)\lambda, \tag{1.7a}$$

$$\Phi(q) = 0. \tag{1.7b}$$

The mass matrix M is nonsingular almost everywhere, λ is the Lagrange multiplier vector and $\partial\Phi/\partial q = G^T$. The algebraic equation (1.7b) often represents geometrical constraints on the system. A simple example of a system such as (1.7) is the physical pendulum. Let L denote the length of the bar, λ is proportional to the force in the bar, and x and y the Cartesian coordinates of the infinitesimal ball of mass one in one end of the bar. Then x, y and λ solve the DAE system

$$\begin{aligned} x'' &= \lambda x, \\ y'' &= \lambda y - g, \\ 0 &= \tfrac{1}{2}(x^2 + y^2 - L^2), \end{aligned} \tag{1.8}$$

where g is the gravity constant.

To state our results, we must first introduce the concept of the *index* of a DAE system. The index is a measure of the singularity of a system. Standard form ODEs, $y' = f(t, y)$, have index zero, the fluid flow system (1.6) has index two, and the constrained mechanical system (1.7) has index three. In general, the higher the index, the more severe the numerical difficulties that we can expect. For the purposes of this paper we will simply define the index as the number of times the constraints of the

system must be differentiated in order to obtain a standard form ODE system. This definition is compatible with previous definitions, for the systems considered here [8].

To illustrate the idea of index, we compute the index of the mechanical system (1.7). If the initial condition $q_0 = q(0)$ is consistent with (1.7b), $\Phi(q_0) = 0$, then the algebraic constraint (1.7b) can be replaced by its differentiated form

$$G^T(q)q' = 0. \tag{1.9}$$

If $\Phi(q_0) = 0$, $q_0' = q'(0)$ and $G^T(q_0)q_0' = 0$ then the condition (1.9) is equivalent to its derivative,

$$\frac{d}{dt}(G^T q') = G^T q'' + G'^T q' = 0. \tag{1.10}$$

We obtain a system of linear equations satisfied by λ by introducing q'' from (1.7a) into the expression above and solving for λ,

$$\lambda = -(G^T M^{-1} G)^{-1}(G^T M^{-1} f + G'^T q'). \tag{1.11}$$

This condition replaces (1.9). Finally, if λ_0, q_0, q_0' satisfy (1.11) at $t = 0$ we can differentiate (1.11) once more and the resulting equation, coupled with (1.7a), form a standard-form ODE system. The index of the original system (1.7) is three, because the constraint was differentiated three times to obtain a standard form ODE system. Similarly, it is easy to verify that the index of the equivalent system (1.7a, 1.9) is two, and that the index of the fluid dynamics system (1.6) is two [5].

In [1], [5], we showed that the k-step constant-stepsize BDF method converges to order $O(h^k)$ for systems of the form (1.1) where the initial values are consistent and the functions are sufficiently smooth, provided the system satisfies

ASSUMPTION 1.1.

(1) The index is less than or equal to one, or

(2) the index is equal to two and $\partial F_2/\partial y \equiv 0$, or

(3) the index is equal to three and the system has the form (1.7).

These conditions are satisfied by the fluid dynamics and mechanical systems mentioned earlier. In this paper we study the practical difficulties which are inherent in implementing a variable-stepsize code based on formulas such as BDF for solving these types of problems.

Often, as in the case of the simulation of mechanical systems of rigid bodies, a DAE system may be written in a different but analytically equivalent way. In some cases this is a good idea, because it may reduce the index and make the problem easier to solve by numerical methods. Therefore, in § 2 we discuss some analytical techniques for rewriting DAE systems in a simpler form, and the reasons why we may or may not want to do this.

The remainder of the paper is concerned with the numerical difficulties associated with solving the systems in their original, high index form. A difficulty that is common to the solution of all high index DAE systems is that the iteration matrix which is used by numerical ODE methods is poorly conditioned when the stepsize is small. This can cause variable-stepsize codes to fail or to give poor diagnostics in case of failure from other causes [9]. In § 3 we give a general technique for scaling the equations and variables in (1.1) that circumvents this difficulty. Scaling can also cause trouble with the convergence and error tests in an automatic code. We study these problems in §§ 4 and 5 and devise convergence tests and error tests that do not suffer from these difficulties, and that are justified by the theory in [1].

2. Alternative forms for DAE systems. In this section we consider some techniques for rewriting a system in an alternative form that may be easier to solve numerically. All of the different forms of the equations that we consider are analytically equivalent in the sense that, given a consistent set of initial conditions, different forms of a system have the same analytical solution. Computationally, however, some forms of the equations may have much different properties than others. We discuss some of the advantages and disadvantages of rewriting a high index DAE system in a different form.

As an example, let us consider some different ways to solve the constrained mechanical system (1.7). First of all, we can attempt to solve the system in its original, index-three form, using an implicit numerical method such as BDF. This technique is actually used in some codes [10], [11] for solving mechanical systems. Solving the problem in this way has the advantages that it is easy to formulate the system (we do not have to differentiate the constraints or rewrite the system in any way), the sparsity of the system is preserved, and the constraints are satisfied exactly on every step. However, there are several difficulties in using a variable-stepsize BDF code for solving systems in this form. First, the iteration matrix that the code uses at each time step is very ill-conditioned for small stepsizes. This can cause severe difficulties with the Newton iteration and with stepsize selection in the code. This difficulty can be partially remedied via the scaling techniques considered in § 3. Secondly, error estimation and stepsize selection algorithms that are normally used in variable-stepsize codes fail for this type of problem (see §§ 4 and 5). One way to overcome this difficulty is to base the error control and stepsize selection strategies only on the vector q, and not on the velocities q' or the Lagrange multipliers λ. We can use the theory developed in [1] to show that the constant-stepsize BDF converges for problems written in this form, but this says nothing about what will happen when there are errors, for example, in the initial velocities. We have found from experiment that a variable-stepsize code which bases its stepsize selection and order control strategies only on q can obtain completely wrong answers when the initial velocities fail to satisfy the condition that the derivative of the constraint should be zero. Thus, we must be very careful that the initial conditions are consistent in the sense that not only the constraint, but also the derivative of the constraint, is zero at the initial time. Unfortunately, all of the techniques for solving mechanical systems that we discuss in this section experience some type of serious difficulty when the initial conditions to the original problem are not chosen to be consistent. For this reason, we do not reject this method (of solving the original index-three problem with variable-stepsize BDF) entirely, but we do doubt its reliability when the initial conditions are inconsistent, or in situations where there are steep gradients or discontinuities in the velocities. It may be possible to reliably solve systems in the index-three form with methods other than BDF, such as extrapolation [8] or defect correction, by controlling errors on the velocities as well as the positions, but we will not take up this subject further in this paper.

A second way of solving (1.7) is to differentiate the constraint and solve the system (1.7a), (1.9). This approach produces a poorly conditioned iteration matrix (though not as poorly conditioned as solving the system in its original form), but we can again eliminate most of these difficulties by scaling the problem as described in § 3. Stepsize selection and error control strategies for variable-stepsize BDF codes fail for these problems, unless we somehow exclude the Lagrange multipliers λ from the error control decisions. In contrast to the error control strategies proposed in the previous paragraph, there is some justification (see § 5) for excluding λ from the error control decisions. In this approach, it is a simple task to verify that the initial conditions $q'(0)$ on the velocity satisfy the algebraic constraint (1.9). If $G^T(q(0))q'(0) \neq 0$ then $q'(0)$

can be corrected by computing an impulse Λ in the system at $t=0$ such that

$$q'_+ - q'(0) = G\Lambda, \qquad G^T q'_+ = 0.$$

However, now we must be careful that the constraint itself is satisfied at the initial time. An initial error in the constraint will contaminate the solution in the whole time interval of interest. There are, though, reasons to believe [5] that any system which does not satisfy the constraint initially is not a good physical model. The main difficulty with solving (1.7a), (1.9) is that of "drifting off" the original constraint. (Note that this is not a problem for solving the Navier-Stokes equations (1.6), which are essentially the same form as (1.7a), (1.9) because there the constraint that we are using is the original constraint of the system.) This formulation of the problem does not force the constraint to be satisfied on every step, and there may be a tendency for the amount by which the constraint is not satisfied to increase from step to step. By using small stepsizes (in an automatic code, by keeping the error tolerances fairly stringent), we can keep small these errors in the amount by which the constraint is not satisfied. Whether this is a serious problem or not depends on the application, although clearly it could be troublesome if the solution is desired over a long interval in time.

A third strategy, which is used in some codes for solving mechanical systems [10], is to eliminate the Lagrange multipliers analytically by means of methods in analytical mechanics to obtain a standard form ODE system. The system of ODEs is assembled from a data structure describing the mechanical system. If the resulting problem is not stiff, this approach has the advantage that the system can be solved by an explicit numerical method. The number of unknowns after this type of transformation usually is smaller, but the sparsity of the system has decreased, which is an important consideration if the problem happens to be stiff. Again, we must be very careful that the initial conditions satisfy that both the constraint and the derivative of the constraint are zero, or we will obviously obtain a solution which is nonsense. A constraint corresponding to an eliminated Lagrange multiplier is automatically satisfied in the chosen representation of the mechanical system. Consider the pendulum (1.8) as an example. Let

$$x = L \sin \varphi, \qquad y = -L \cos \varphi.$$

Then the algebraic constraint of constant length is fulfilled and the well-known ODE is

$$\ddot{\varphi} - \gamma \sin \varphi = 0, \qquad \gamma = \frac{g}{L}.$$

Various combinations of the above strategies can be employed. Baumgarte [12] discusses a technique for circumventing this problem of "drifting off" the constraints $\Phi(q)$ by adding to the original equations an equation consisting of a linear combination of Φ, $d\Phi/dt$ and $d^2\Phi/dt^2$. The linear combination is chosen so that the resulting system damps errors in satisfying the constraint equation. This approach is similar, but not identical, to penalty function methods (Lötstedt [13], Sani et al. [14]). Depending on the choice of the parameters in the linear combination, we may see any of the difficulties discussed earlier. This technique introduces extraneous eigenvalues into the system, which may or may not cause difficulties. Finally, the penalty techniques have the disadvantage that if the initial conditions are not posed correctly, they introduce a nonphysical transient into the problem [14].

While all of these techniques have their shortcomings, we have attempted here to put them on a firmer foundation, and to suggest some ways to implement them more effectively. All of these techniques experience serious difficulties when the initial conditions are not consistent.

Before leaving the subject of alternate forms for DAE systems, there is one more aspect of this problem that we wish to consider. Sometimes there is a choice of which variables to use for solving a problem. For example, in the system

$$\begin{aligned} u' &= v, \\ v' &= f(u, v, t) + G(u)\lambda, \\ G^T v &= 0 \end{aligned} \tag{2.1}$$

we could have replaced v in the constraint by u' to obtain

$$\begin{aligned} u' &= v, \\ v' &= f(u, v, t) + G(u)\lambda, \\ G^T u' &= 0. \end{aligned} \tag{2.2}$$

Are there any advantages in writing the system in one form over the other? Using BDF with Newton iteration for linear problems in exact arithmetic, the two forms of the equations give identical solutions. (This is because Newton's method is exact in one iteration for linear systems, and because the equations which result from discretizing both systems by BDF are identical—this last fact is obviously true for nonlinear systems too.) For nonlinear systems in exact arithmetic we know of no reasons why Newton's method would be more likely to converge for one form of the equations than the other. Both forms of the system may lead to very poorly conditioned iteration matrices. In the next section we will suggest scaling techniques to overcome this difficulty. These techniques are directly applicable to problems of the form (2.1) but not to (2.2) (though it is often possible to devise ways to scale the analytically equivalent systems such as (2.2)). This may be a reason to prefer (2.1). Everything that we discuss in this paper, with the exception of the scaling techniques introduced in § 3, is applicable to systems such as (2.2) where these obvious substitutions have been made. Thus, if it is more convenient to solve one of these alternative forms of a system, then there is some justification for doing so.

3. Conditioning of matrices arising in the solution of DAEs. In this section we study the solution of DAEs of the form (1.1). Conditioning is a problem for DAE systems, and especially for high index systems, because the condition number of the iteration matrix for a system with index of m is $O(h^{-m})$ ([5, Thm. 4.1]). We describe schemes for scaling systems (1.1) so that the iteration matrices are no longer singular as $h \to 0$, and we discuss how these scaling techniques can be conveniently implemented into existing DAE software.

The system of linear equations to be solved in each Newton iteration step is

$$Az = b.$$

If we use Gaussian elimination with partial pivoting we know that the computed solution $z + \Delta z$ satisfies

$$(A + \Delta A)(z + \Delta z) = b, \tag{3.1}$$

where

$$\|\Delta A\|_\infty \leqq r\mathbf{u}\|A\|_\infty. \tag{3.2}$$

In (3.2) r is a moderate number and $\mathbf{u}$ is the machine unit. The accuracy of this computation is improved if we introduce a row scaling of A by premultiplying by a

diagonal matrix D. Then by (3.2)

$$
\begin{aligned}
|\Delta z_i| &\leqq |(A^{-1}D^{-1}D\Delta A(z+\Delta z))_i| \\
&\leqq \sum_j |(A^{-1}D^{-1})_{ij}||(D\Delta A(z+\Delta z))_j| \\
&\leqq \sum_j |(A^{-1}D^{-1})_{ij}| \, \|D\Delta A\|_\infty \|z+\Delta z\|_\infty \\
&\leqq r\mathbf{u} \sum_j |(A^{-1}D^{-1})_{ij}| \, \|DA\|_\infty \|z+\Delta z\|_\infty .
\end{aligned} \tag{3.3}
$$

We consider three cases: index one, index two, and index three systems of the form

$$
\begin{aligned}
x' - f(x, y, t) &= 0, \\
g(x, y, t) &= 0.
\end{aligned} \tag{3.4}
$$

(These ideas extend easily to the slightly more general form (1.1).) For these problems, the iteration matrix is written as

$$
hJ_n = \begin{bmatrix} \alpha_0 I - h\dfrac{\partial f}{\partial x} & -h\dfrac{\partial f}{\partial y} \\ h\dfrac{\partial g}{\partial x} & h\dfrac{\partial g}{\partial y} \end{bmatrix}. \tag{3.5}
$$

Case I. When the index is one, we have that $\partial g/\partial y$ is nonsingular, so hJ_n is nonsingular as $h \to 0$ if we scale the rows corresponding to the algebraic constraint by $1/h$. Since we are not scaling variables, but only equations, the effect of this scaling should be to improve the accuracy of the solution of the linear system, for *all* variables.

Case II. For this case, we will assume that the index is two, and that $\partial g/\partial y \equiv 0$. By explicitly computing $(hJ_n)^{-1}$ we find that the orders of the blocks of the inverse are

$$
\begin{pmatrix} 1 & 1/h \\ 1/h & 1/h^2 \end{pmatrix}, \tag{3.6}
$$

where the elements in the first row correspond to x and those in the second row to y. If we scale the bottom rows of hJ_n (corresponding to the "algebraic" constraints) by $1/h$, then the scaled matrix can be written as

$$
hJ_n' = \begin{pmatrix} \alpha_0 I - h\dfrac{\partial f}{\partial x} & -h\dfrac{\partial f}{\partial y} \\ \dfrac{\partial g}{\partial x} & 0 \end{pmatrix}. \tag{3.7}
$$

It follows from (3.3) that roundoff errors proportional to $\mathbf{u}/h$ and $\mathbf{u}/h^2$ are introduced in x and y, respectively, while solving the unscaled linear system. With the suggested scaling the roundoff errors are of $O(\mathbf{u})$ in x and $O(\mathbf{u}/h)$ in y. As $h \to 0$ these errors can begin to dominate the solution y. This is likely to cause difficulties for the error estimates and convergence tests in an automatic code. These difficulties, along with what can be done to minimize their effects, will be described in greater detail later. For now, we merely note that the effect of the proposed scaling is to control the size of the roundoff errors in x which are introduced in solving the linear system. At the same time, the "algebraic" variables y may contain errors proportional to $\mathbf{u}/h$. However, since the values of y do not affect the state of the system directly (that is, how the system will respond at future times), we may be willing to tolerate much larger errors

in y than in x. In any case, this scaling is a significant improvement over the original scaling (3.5). For the scaled system, the errors are considerably diminished, and the largest errors are confined to the variables which are in some sense the least important. Painter [15] describes difficulties due to ill-conditioning for solving incompressible Navier-Stokes equations of the form (1.6), and employs essentially the same scaling that we have suggested here to solve the problem. These difficulties are most severe when an automatic code is using a very small stepsize, as in starting a problem or passing over a discontinuity in some derivative.

We further note that if we are using Gaussian elimination with partial pivoting, we do not need any column scaling. The solution will be the same without this scaling, because it does not affect the choice of pivots [16]. What the analysis shows is that the errors which are due to ill-conditioning are concentrated in the "algebraic" variables of the system and not in the "differential" variables. Thus, we must be particularly careful about using the "algebraic" variables in other tests in the code which might be sensitive to the errors in these variables.

Case III. For this case, we will work with systems of the form

$$\begin{aligned} u' &= v, \\ v' &= f(u, v, t) + G(u)\lambda, \\ \Phi(u) &= 0, \end{aligned} \tag{3.8}$$

i.e., mechanical systems where we have assumed, without loss of generality for the purposes of this discussion, that $M = I$. For these systems, the iteration matrix is written as

$$hJ_n = \begin{pmatrix} \alpha_0 I & -hI & 0 \\ hX & \alpha_0 I + hY & -hG \\ hG^T & 0 & 0 \end{pmatrix} \tag{3.9}$$

where

$$X = -\frac{\partial f}{\partial u} - \frac{\partial G}{\partial u}\lambda,$$

and

$$Y = -\frac{\partial f}{\partial v}.$$

Let $(hJ_n)^{-1}$ be partitioned into nine blocks such that the three block rows correspond to the variables u, v and λ and the three block columns correspond to the differential equations and the algebraic constraint in (3.8). Then the leading terms in $\gamma = h/\alpha_0$ in the blocks of $(hJ_n)^{-1}$ are

$$\frac{1}{\alpha_0}\begin{pmatrix} I-P & \gamma(I-P)S^{-1} & \gamma^{-1}S^{-1}GA \\ -\gamma^{-1}P & (I-P)S^{-1} & \gamma^{-2}S^{-1}GA \\ -\gamma^{-2}AG^T & -\gamma^{-1}AG^TS^{-1} & \gamma^{-3}A \end{pmatrix}. \tag{3.10}$$

In (3.10) the notation is $S = I + \gamma Y + \gamma^2 X$, $A = (G^T S^{-1} G)^{-1}$ and $P = S^{-1}GAG^T$. If we perform row equilibrium, i.e. scale the last row in (3.9) by $1/h$, then (3.3) and (3.10) give that the roundoff errors in u, v and λ are proportional to $\mathbf{u}$, $\mathbf{u}/h$ and $\mathbf{u}/h^2$, respectively. Note that if we scale the second row in (3.9) by h then the ith row block in each column of $(hJ_n)^{-1}D^{-1}$ is of $O(\gamma^{1-i})$. The errors with either of these scalings are much smaller than if we had solved the system with the original unscaled matrix (3.9), which had condition number $O(1/h^3)$. We may not be interested in the values

of λ, because these Lagrange multipliers have no direct effect on the state of the system. The situation is not quite so simple with respect to the velocities v, however. The components of v in certain directions *do* in part determine the state of the system. In the two-dimensional pendulum example (1.8) such a direction is perpendicular to the bar. However, the $O(\mathbf{u}/h)$ errors that we can expect in v using this scaling are still considerably smaller than the $O(\mathbf{u}/h)$, $O(\mathbf{u}/h^2)$ and $O(\mathbf{u}/h^3)$ errors that we could expect in u, v and λ in solving the original unscaled system. The analysis shows that the errors which are due to ill-conditioning are concentrated in the variables v and λ, and not in u. Thus, we must be careful about using the "algebraic" variables v and λ in other tests in the code.

One further question that remains is how to implement row scaling in a general DAE code. There is a very nice solution to this problem. In a general purpose DAE code [18] (for solving systems of the form $F(t, y, y') = 0$), there is a subroutine which the user writes for computing the residual $F(t, y, y') = \Delta$, given (t, y, y'). The user can scale Δ inside this subroutine, and according to our guidelines, if we pass the stepsize h in the argument list to this subroutine. This way, the scaling costs virtually nothing. An alternative idea is to provide an option to automatically do row equilibration, or to use linear system solvers which perform row scaling, as suggested in Shampine [17].

4. Tests for terminating the corrector iteration. In the last section we saw that for high index systems, even with scaling, there are relatively large errors in some of the variables. For the most part, these variables do not determine the state of the system, so these errors are in some sense tolerable. However, from the point of view of an automatic code where we must have some criterion for deciding when to terminate the corrector iteration, the errors in these variables are still a source of difficulties. Our objective in this section is to show that, from the point of view of propagation of errors in the state variables of a system, it is sufficient to terminate the Newton iteration based on the errors in the *scaled* variables, where the variables are scaled by powers of the stepsize as in the previous section. This eliminates troubles due to ill conditioning in the corrector iteration part of a code.

We will examine the propagation of errors caused by the interruption of the Newton iterations for a BDF method, for systems of the form (1.1) of index one, two and three.

For these purposes, let (x_n, y_n) be the computed solution (where the corrector iteration has not necessarily been solved exactly), and let $(\tilde{x}_n, \tilde{y}_n)$ be the true solution to the difference equation. Then $\tilde{x}_n = x_n + \delta_n^x$, $\tilde{y}_n = y_n + \delta_n^y$, where δ_n^x and δ_n^y are the errors in x_n and y_n due to terminating the Newton iteration early. Let $e_n^x = x_n - x(t_n)$, $e_n^y = y_n - y(t_n)$ be the global errors and let τ_n be the local truncation error. Then we have

$$
\begin{aligned}
0 &= F(\tilde{x}_n, \rho\tilde{x}_n/h, \tilde{y}_n, t_n) \\
&= F(x_n + \delta_n^x, \rho(x_n + \delta_n^x)/h, y_n + \delta_n^y, t_n) \\
&= F(x(t_n) + e_n^x + \delta_n^x, \rho(x(t_n) + e_n^x + \delta_n^x)/h, y(t_n) + e_n^y + \delta_n^y, t_n) \\
&= F(x(t_n), x'(t_n), y(t_n), t_n) + \frac{\partial F}{\partial x}(e_n^x + \delta_n^x) \qquad (4.1) \\
&\quad + \frac{\partial F}{\partial x'}(\rho(e_n^x + \delta_n^x)/h + \tau_n) \\
&\quad + \frac{\partial F}{\partial y}(e_n^y + \delta_n^y) + (\text{higher order terms}).
\end{aligned}
$$

For notational convenience, introduce $F_{1x} = \partial F_1/\partial x$, $F'_{1x} = \partial F_1/\partial x'$, $A_1 = \alpha_0 F'_{1x} + hF_{1x}$, $A_2 = \partial F_1/\partial y$, $A_3 = \partial F_2/\partial x$, $A_4 = \partial F_2/\partial y$. Then the errors must satisfy

$$\begin{pmatrix} A_1 & hA_2 \\ hA_3 & hA_4 \end{pmatrix} \begin{pmatrix} e_n^x + \delta_n^x \\ e_n^y + \delta_n^y \end{pmatrix} = \begin{pmatrix} -F'_{1x}(c_n + h\tau_n) \\ 0 \end{pmatrix} + (\text{higher order terms}) \tag{4.2}$$

where $c_n = \sum_{i=1}^{k} \alpha_i (e_{n-i}^x + \delta_{n-i}^x)$. Only higher order terms of δ_n^x and δ_n^y are present in the right-hand side of (4.2). From (4.2) we find that

$$\begin{pmatrix} e_n^x \\ e_n^y \end{pmatrix} = \begin{pmatrix} -\delta_n^x \\ -\delta_n^y \end{pmatrix} + \begin{pmatrix} A_1 & hA_2 \\ hA_3 & hA_4 \end{pmatrix}^{-1} \left[\begin{pmatrix} \zeta \\ 0 \end{pmatrix} + (\text{higher order terms}) \right], \tag{4.3}$$

where ζ is the right-hand side of (4.2a). For the purposes of only controlling the errors in x and y at t_n, the same criterion can be used on both δ_n^x and δ_n^y. In the analysis in [1] of the accumulated error in x and y as time progresses we derived the requirements on the residual η from the Newton iteration to obtain $\|e_n^x\| = O(h^k)$ and $\|e_n^y\| = O(h^k)$ with the kth order BDF. It follows from (4.1) that

$$0 = hF(x_n, (\alpha_0 x_n + c_n)/h, y_n, t_n) + \begin{pmatrix} A_1 & hA_2 \\ hA_3 & hA_4 \end{pmatrix} \begin{pmatrix} \delta_n^x \\ \delta_n^y \end{pmatrix} + (\text{higher order terms}).$$

Hence,

$$\begin{pmatrix} A_1 & hA_2 \\ hA_3 & hA_4 \end{pmatrix} \begin{pmatrix} \delta_n^x \\ \delta_n^y \end{pmatrix} \approx h\eta = -h \begin{pmatrix} F_{1n} \\ F_{2n} \end{pmatrix}, \tag{4.4}$$

where F_{1n} and F_{2n} are evaluated at x_n, y_n and t_n. In the index one case η must be of $O(h^k)$ according to the results in [1]. The iteration should be terminated when δ_n^x and $h\delta_n^y$ are of $O(h^{k+1})$ in (4.4). Let $\eta^T = (\eta_1^T, \eta_2^T)$. If $A_4 \equiv 0$ and the index is two, then η_1 and η_2 must be of $O(h^k)$ and $O(h^{k+1})$. This is achieved by letting δ_n^x and $h\delta_n^y$ be proportional to h^{k+1} in (4.4). Equation (4.4) is satisfied with $x^T = (u^T, v^T)$ and $y = \lambda$ by the index three system (3.8). Here, we require $\|\eta_1\| = O(h^{k+1})$ and $\|\eta_2\| = O(h^{k+2})$ in [1] which is obtained by taking $\|\delta_n^x\| = O(h^{k+1})$ and $\|h\delta_n^y\| = O(h^{k+1})$. These conclusions are independent of the row scaling.

In general, variables should never be totally excluded from the test for convergence of the corrector iteration. Even if a variable occurs linearly, as for example in Brown and Gear [19], it is a mistake to exclude it from the convergence test in an automatic code. The reason for this is that codes are so often using Jacobians which were calculated on previous steps and possibly even based on different stepsizes that in general we cannot count on the Newton iteration to converge in one step.

5. Error tests. In this section we will examine the reasons why some variables should be excluded from the error test in an automatic code.

It follows from (4.1) and (4.2) that the errors in the "algebraic" variable y on previous time steps, e_i^y, $i < n$, do not directly influence the errors in any of the variables at the current time t_n for systems of index one and two satisfying Assumption 1.1 since they do not appear in (4.1). Therefore, we can consider deleting these variables from the error estimate. This could be advantageous for the smooth operation of a code. Consider first the case of solving index one systems. Suppose, for example, that on one step we make a fairly large mistake by terminating the Newton iteration before it has really converged, and that on this step the value of y that should have been computed has a large error in it, but that based on the incorrect value it passes the error test anyway. If we base the error test on y, then on the next step this could cause a big problem, because the new value of y does not approach the old (incorrect) value of y, so that their difference, and hence the error estimate, does not approach zero as

$h \to 0$. Because of this, we feel that in this case it is probably wise to leave y out of the error control decisions. The main drawback in this strategy is that if we would like to know the values of y at interpolated points (between mesh points) then the stepsize should be based in part on the values of y. For index two systems, the stepsize control strategy in an automatic BDF code will fail, for reasons explained in Petzold [9], unless the "algebraic" variables y are excluded from the error test. For Navier-Stokes systems, this means that the pressure should be excluded from the error test, and for constrained mechanical systems (1.7a), (1.9) where the constraint has been differentiated once, it means that the Lagrange multipliers should be excluded from the error test.

Another possible error estimate is based on the observation that the part of the error in x and y due to the truncation error $h\tau_n$ in (4.2) in the present step is

$$e_{\tau_n} = \begin{pmatrix} e^x_{\tau_n} \\ e^y_{\tau_n} \end{pmatrix} = -(hJ_n)^{-1} \begin{pmatrix} F'_{1x} h\tau_n \\ 0 \end{pmatrix}, \tag{5.1}$$

where higher order terms have been omitted. This is the contribution to the global error that is directly influenced by a change of the stepsize. An estimate of e_{τ_n} can be obtained by multiplying the usual error estimate by the matrix

$$(hJ_n)^{-1} \begin{pmatrix} F'_{1x} & 0 \\ 0 & 0 \end{pmatrix}.$$

An approximate factorization of hJ_n is available from the Newton iteration. Then an estimate of e_{τ_n} can be computed according to (5.1). The step size is chosen such that e_{τ_n} is less than a given error tolerance. This is a generalization of the estimate given by Sincovec et al. [20] for constant-coefficient DAE systems (see also Petzold [9]). If $A_4 = 0$ and the index is two, we find from (3.6) that $e^x_{\tau_n} \sim h\tau_n \sim h^{k+1}$ as we would expect but that $e^y_{\tau_n} \sim \tau_n \sim h^k$.

For the index three mechanical systems, the situation is almost as simple. According to (3.10) the leading term in the contribution from the discretization error $h\tau_n$ to the errors e^u_n and e^v_n is

$$\begin{pmatrix} e^u_{\tau_n} \\ e^v_{\tau_n} \end{pmatrix} = \frac{1}{\alpha_0} \begin{pmatrix} I - P & \gamma(I-P)S^{-1} \\ -P/\gamma & (I-P)S^{-1} \end{pmatrix} \begin{pmatrix} h\tau^u_n \\ h\tau^v_n \end{pmatrix}. \tag{5.2}$$

The next term of higher order in the lower left-hand block is of $O(\gamma)$. The lower left part of the first matrix above is of $O(h^{-1})$. In general we have $P\tau^u_n \neq 0$ and the order of the asymptotic behavior of $e^v_{\tau_n}$ is one less than what we can expect for differentiated variables from index one or two systems. What we would ideally like to control is $e^u_{\tau_n}$ and the truncation error in the components of the velocity vector v which are in allowable directions (which would not cause u to violate the constraint). Stepsize control strategies used in BDF codes will fail [9] if we base the strategy on the values of v and/or λ. One way to solve this problem using a general purpose BDF code is to base the error control and stepsize selection strategies solely on u. This is actually done in a production code [10], [11] for solving mechanical systems. However, we question whether this is always a reliable procedure.

Another possible error estimate for the index three mechanical system is based on the observation from (5.2) that we can obtain the leading terms in $e^u_{\tau_n}$ and $e^v_{\tau_n}$ by solving

$$(hJ_n) \begin{pmatrix} e^u_{\tau_n} \\ e^v_{\tau_n} \\ z \end{pmatrix} = \begin{pmatrix} h\tau^u_n \\ h\tau^v_n \\ 0 \end{pmatrix}$$

where the value of z does not matter to us. Now if we are only interested in the components of v in the nullspace $N(G^T)$ of G^T then such a component is

$$v^* = (I - P)v, \tag{5.3}$$

where

$$P = S^{-1}G(G^TS^{-1}G)^{-1}G^T,$$

as in (3.10). The matrix P is a projector [21] since $P^2 = P$. Let $\|\cdot\|$ denote the spectral matrix norm. For h sufficiently small

$$S^{-1} = I + hS_1, \qquad \|S_1\| = O(1),$$

and from the definition of P in (5.3)

$$P = P_0 + hP_1, \qquad \|P_1\| = O(1),$$

where $P_0 = G(G^TG)^{-1}G^T = P_0^T$, an orthogonal projector [21]. Therefore,

$$\|P\| = 1 + O(h), \qquad \|I - P\| = 1 + O(h),$$

and $\|v^*\|$ is bounded by $\|v\|$,

$$\|v^*\| \leqq \|v\|(1 + O(h)). \tag{5.4}$$

From the pendulum example in (1.8) and the definition of S and P we find that with $u^T = (x, y)$

$$\|S - I\| = O(h^2), \quad P_0 = u(u^Tu)^{-1}u^T, \quad \|P - P_0\| = O(h^2).$$

The motion of the pendulum ball is constrained in the direction $u = P_0u$ and free in the perpendicular direction $w^T = (y, -x)$ for which $P_0w = 0$.

The contribution to the error in v^* from the truncation error is

$$e_{\tau_n}^{v^*} = (I - P)e_{\tau_n}^{v}. \tag{5.5}$$

It follows from (3.10) that the leading term in $e_{\tau_n}^{v^*}$ is the solution to

$$(hJ_n)\begin{pmatrix} e_{\tau_n}^{v^*} \\ z_1 \\ z_2 \end{pmatrix} = \begin{pmatrix} e_{\tau_n}^{v} \\ 0 \\ 0 \end{pmatrix}.$$

The reasons why the particular projector in (5.3) is chosen are first that P is close to an orthogonal projector so that a relation such as (5.4) is satisfied and second the determination of $e_{\tau_n}^{v^*}$ is not too complicated. Thus we can obtain $e_{\tau_n}^{u}$ and $e_{\tau_n}^{v^*}$ with two extra back substitutions using the approximate factorization of hJ_n which was computed during the Newton iteration. It follows from (5.2) and (5.5) that the terms of lowest order in $e_{\tau_n}^{v^*}$ are

$$(I - P)(-hP\tau_n^u/\gamma + h(I - P)S^{-1}\tau_n^v) = h(I - P)\tau_n^v + O(h^{k+2}).$$

The projector in front of the truncation errors is bounded independently of h. Hence, $e_{\tau_n}^{v^*}$ is proportional to h^{k+1}. This is in contrast to the error $e_{\tau_n}^{v}$ which is $O(h^k)$ due to the error component $Pe_{\tau_n}^{v} \sim P\tau_n^u$ in Pv. If the approximate factorization of hJ_n is the exact factorization at t_i, $i < n$, then v^* corresponding to the computed $e_{\tau_n}^{v^*}$ will not satisfy $G_n^Tv^* = 0$ exactly, but rather

$$0 = G_i^Tv^* = G_n^Tv^* + hr, \qquad \|r\| = O(1),$$

where $r = -(n-i)(G_n')^T v^* + \cdots$. In sum, we have derived an estimate based on the size of $(e^u_{\tau_n}, e^{v^*}_{\tau_n})$. It is somewhat complicated, but it may be more reliable than basing the estimate solely on τ^u_n.

There are several issues to consider when implementing these alternative error tests (where some variables are excluded from the error tests) in software for the solution of differential/algebraic systems. First, we note that there are two main tests which control the operation of a code—namely, the test to decide when to terminate the Newton iteration, and the error test to accept or reject the current step and to control the stepsize. If we exclude some variables from the error test but base the convergence test on the values of all the variables (or scaled variables), then the weighted max norm seems to be preferable to some other norms. Why? Consider for example the RMS norm, which is used in several popular ODE codes. Since this norm looks at all the variables for the convergence test, but only some of them for the error test, and it is weighted by the number of variables, then something like the following can happen. The code can pass the convergence test very easily, possibly because some very small components are included in the norm for the convergence test, but not really have converged to a great enough accuracy in the variables which are included in the error test. This can cause the code not to run smoothly, or to be unreliable. With the max norm, this kind of incompatibility in norms cannot occur. The second observation is that it is relatively easy and cheap to implement the error test where some variables are excluded in a code which allows the user to write a subroutine to define his own norm. We can pass a flag to the norm routine which tells it whether it wants a norm for the convergence test or for the error test, and then the norm can be computed based on the appropriate variables.

Acknowledgment. We wish to thank Bob Skeel for his comments on an earlier version of this paper, and in particular for suggesting (3.3) as a basis for the roundoff error analysis.

REFERENCES

[1] P. LÖTSTEDT AND L. R. PETZOLD, *Numerical solution of nonlinear differential equations with algebraic constraints* I: *Convergence results for backward differentiation formulas*, Math. Comp., to appear.

[2] K. E. BRENAN, *Stability and convergence of difference approximations for higher index differential-algebraic systems with applications in trajectory control*, Ph.D. Thesis, 1983, Univ. California, Los Angeles.

[3] C. W. GEAR, *Simultaneous numerical solution of differential/algebraic equations*, IEEE Trans. Circuit Theory, CT-18 (1971), pp. 89-95.

[4] J. DIEUDONNÉ, *Foundations of Modern Analysis*, Academic Press, New York, 1969.

[5] P. LÖTSTEDT AND L. R. PETZOLD, *Numerical solution of nonlinear differential equations with algebraic constraints*, SAND83-8877, Sandia National Laboratories, Livermore, CA, 1983.

[6] P. M. GRESHO, R. L. LEE AND R. L. SANI, *On the time-dependent solution of the incompressible Navier-Stokes equations in two and three dimensions*, in Recent Advances in Numerical Methods in Fluids, Pineridge, Swansea, 1980.

[7] J. WITTENBURG, *Dynamics of Systems of Rigid Bodies*, Teubner, Stuttgart, 1977.

[8] C. W. GEAR AND L. R. PETZOLD, ODE *methods for the solution of differential/algebraic systems*, SIAM J. Numer. Anal., 21 (1984), pp. 367-384.

[9] L. PETZOLD, *Differential/algebraic equations are not* ODEs, this Journal, 3 (1982), pp. 367-384.

[10] D. BENSON, personal communication, Lawrence Livermore National Laboratory, Livermore, CA.

[11] N. ORLANDEA, D. A. CALAHAN AND M. A. CHACE, *A sparsity-oriented approach to the dynamic analysis and design of mechanical systems—Part* 1 *and Part* 2, Trans. ASME, J. Engineering for Industry, Ser. B, 99 (1977), pp. 773-784.

[12] J. BAUMGARTE, *Stabilization of constraints and integrals of motion in dynamical systems*, Comput. Meth. Appl. Mech. Engrg., 1 (1972), pp. 1-16.

[13] P. LÖTSTEDT, *On a penalty function method for the simulation of mechanical systems subject to constraints*, TRITA-NA-7919, Royal Institute of Technology, Stockholm, Sweden, 1979.

[14] R. L. SANI, B. E. EATON, P. M. GRESHO, R. L. LE AND S. T. CHAN, *On the solution of the time-dependent incompressible Navier–Stokes equations via a Penalty Galerkin finite element method*, UCRL-85354, Lawrence Livermore National Laboratory, Livermore, CA, 1981.

[15] J. F. PAINTER, *Solving the Navier–Stokes equations with* LSODI *and the method of lines*, Report UCID-19262, Lawrence Livermore National Laboratory, Livermore, CA, 1981.

[16] A. VAN DER SLUIS, *Condition, equilibration and pivoting in linear algebraic systems*, Numer. Math., 15 (1970), pp. 74–86.

[17] L. F. SHAMPINE, *Conditioning of matrices arising in the solution of stiff* ODEs, SAND82-0906, Sandia National Laboratories, Albuquerque, NM, 1982.

[18] L. R. PETZOLD, *A description of* DASSL: *A differential/algebraic system solver*, SAND82-8637, Sandia National Laboratories, Livermore, CA, 1982.

[19] R. L. BROWN AND C. W. GEAR, *Documentation for* DFASUB—*A program for the solution of simultaneous implicit differential and nonlinear equations*, UIUCDCS-R-73-575, Univ. Illinois, Urbana-Champaign, 1973.

[20] R. F. SINCOVEC, B. DEMBART, M. A. EPTON, A. M. ERISMAN, J. W. MANKE AND E. L. YIP, *Solvability of large-scale descriptor systems*, Report, Boeing Computer Services Company, Seattle, WA, 1979.

[21] A. BEN-ISRAEL AND T. N. E. GREVILLE, *Generalized Inverses: Theory and Applications*, John Wiley, New York–London, 1974.

SIAM J. SCI. STAT. COMPUT.
Vol. 7, No. 3, July 1986

MAINTAINING SOLUTION INVARIANTS IN THE NUMERICAL SOLUTION OF ODEs*

C. W. GEAR†

Abstract. Techniques for maintaining the numerical consistency of equality and inequality invariants known to be obeyed by the true solutions of ODEs are discussed. Shampine [11] has examined the problem and made some recommendations for one-step methods. A general framework is used to justify a particular form of his technique for both one-step and multistep methods in the case of equality invariants. This framework can also be used for inequality invariants, but is not satisfactory for multistep methods. However, in a large class of inequality invariants, typified by those arising in chemical kinetic problems, it is shown that the invariant can be satisfied automatically by the numerical method.

Key words. conservation laws, invariants, inequalities, chemical kinetics, Lagrange multipliers, index-2 systems

AMS(MOS) subject classifications. 34A50, 65L05

1. Introduction. Many systems of ordinary differential equations which are solved numerically have known invariants, that is, relations which hold along any solution trajectory. These may be equalities such as the invariance of the total energy, momentum, mass, etc., and as such are integrals of the system, or they may be inequalities of the form $b(y, t) > 0$ or $b(y, t) \geqq 0$ which are known from physical principles and can, in principle, be demonstrated from the equations. Often it is desirable to conserve these invariants numerically because the solution may be quite sensitive to small changes in some of these quantities and may even become unstable if some of the inequalities are violated. We refer the reader to Shampine [11] for a discussion of these reasons and a review of other literature.

The equations take the form

$$y' = f(y, t), \quad y(0) = y_0, \tag{1.1}$$

where, along any particular solution trajectory, $y(t)$, we have

$$g(y(t), t) = g(y_0, t_0), \tag{1.2}$$

$$b(y(t), t) \geqq 0 \text{ if } b(y_0, t_0) \geqq 0. \tag{1.3}$$

(The notation $y \geqq 0$ when y is a vector means that all components of y satisfy the inequality.) Equation (1.1) is sufficient to uniquely determine a solution $y(t)$ which automatically satisfies eqs. (1.2) and (1.3). The problem is to ensure that the latter two equations are satisfied by the numerical solution.

As examples of equality invariants we consider the D3 system of equations from the stiff test set [5] and the equations for a pendulum. The D3 system is

$$y_1' = y_3 - 100y_1y_2, \tag{1.4a}$$

$$y_2' = y_3 + 2y_4 - 100y_1y_2 - 2\times10^4y_2^2, \tag{1.4b}$$

$$y_3' = -y_3 + 100y_1y_2, \tag{1.4c}$$

$$y_4' = -y_4 + 10^4y_2^2. \tag{1.4d}$$

* Received by the editors November 26, 1984, and in revised form June 30, 1985. This work was supported in part by the U.S. Department of Energy under grant DEAC0276ERO2383, and in part by the Applied Mathematical Sciences subprogram of the Office of Energy Research, U.S. Department of Energy, under contract W-31-109-Eng-38 while the author was on leave at Argonne National Laboratory.

† Department of Computer Science, University of Illinois at Urbana-Champaign, Urbana, Illinois 61801.

Equations (1.4) describe a chemical kinetic problem in which y_1 and y_2 react to form y_3, and y_2 reacts with itself to form y_4. It can be seen that the mass-balance invariants of this system are

$$k_1 = y_1 + y_3 \tag{1.5a}$$

and

$$k_2 = y_2 + y_3 + 2y_4. \tag{1.5b}$$

Note that these are linear, the weights are non-negative, and that every variable appears in at least one invariant. For systems that satisfy invariants meeting these criteria, we can conclude that they satisfy an invariant

$$k = \sum_1^s w_i y_i = w^T y \tag{1.5c}$$

where $w = \{w_i\}$ are a set of strictly positive weights for the s components of the system, and w^T is the transpose of w.

A simple pendulum satisfies

$$\theta' = \omega, \tag{1.6a}$$

$$\omega' = -g \sin(\theta)/L \tag{1.6b}$$

where θ is the angle with the vertical, L the pendulum length, and g the acceleration due to gravity. Equations (1.6) have a constant energy, so we have the invariant

$$k = \omega^2 - 2g \cos(\theta)/L \tag{1.7}$$

This is a nonlinear invariant.

As an example of an inequality invariant, we also use eqs. (1.4). Because the variables are the concentrations of radicals in a chemistry model, they should be non-negative. Indeed, if the equations permitted the concentrations to assume negative values from a consistent initial condition with $y_0 \geqq 0$, the mathematical model would be incorrect. In fact, it is simple to prove that these and similar equations do satisfy $y(t) \geqq 0$ for all $t > 0$ if $y_0 \geqq 0$. Also, note that this lower bound and eqs. (1.5) imply an upper bound for all variables, namely $y_i \leqq k/w_i$.

Shampine [11] discusses one-step methods for the numerical integration of eq. (1.1) and suggests that, after each step, the numerical solution can be perturbed by the minimum amount to satisfy eq. (1.2). He also argues that perturbing the solution at the end of a step to satisfy the inequality invariants (1.3) actually decreases the local error, at least in the case of simple invariants such as $y \geqq 0$. His treatment is restricted to one-step methods because of the potential impact of the perturbations used to satisfy the invariants on the error estimating formulas used in multistep methods. Since these formulas usually are equivalent to high-order divided differences of the computer values, arbitrary perturbations can lead to unreasonably large error estimates which will cause inefficiency by both reducing the stepsize and reducing the order.

This paper takes a different approach to the problem of equality invariants, producing instead a modified set of equations which automatically satisfy the invariants. Computationally, the technique can be implemented with exactly the same code as would be used by Shampine in one of his suggestions, but the approach provides a theory which justifies its use in multistep methods.

The same approach can be used for inequality invariants, but its application can lead to discontinuous derivatives, which impact a multistep method severely. However,

in a number of classes of problems, of which virtually all chemical kinetic problems are examples, it is possible to control the code so that the inequality invariants are satisfied by the numerical solution automatically. The class of problems are those that can be written in the form

$$y' = -Dy + q, \tag{1.8}$$

where the diagonal matrix D and the vector q are non-negative functions of y and t on the region $y \geqq 0$, and q is bounded. For such problems it is clear that $y \geqq 0$ is an invariant of the solution. We will present a technique in § 5 which will cause it to be an invariant of the numerical solution also. We will see that, although $q(y)$ is an unbounded function of y in chemical kinetic applications, the boundedness of y, which implies a bound on $q(y)$, is sufficient to allow the results to be applied.

2. Linear equality invariants. It is convenient to classify the equality invariants into two classes, linear and nonlinear. Many authors have pointed out that the majority of numerical methods preserve linear invariants. This follows because the linear invariants represent linear dependencies between the right-hand sides of eq. (1.1). Thus, an invariant of the form eq. (1.5c) implies that

$$w^T y' = 0, \tag{2.1}$$

which, using eq. (1.1) means that

$$w^T f(y, t) = 0 \tag{2.2}$$

holds everywhere. If the numerical method has the form

$$y_{n+1} = y_n + \phi_n(y_n, t_n, h) \tag{2.3}$$

where y_n is the numerical approximation to $y(t_n)$ at t_n, and ϕ also satisfies $w^T\phi = 0$ everywhere, then y_{n+1} satisfies the invariant if y_n does. The fact that $w^T\phi = 0$ follows from eq. (2.2), the form of the method, and induction in most cases. Because linear invariants are automatically satisfied (apart from roundoff errors[1]), and because it is inexpensive to use them to reduce the order of the system by elimination of variables, this is usually the best approach to take. Edsberg [4] points out that numerical accuracy in the matrix work required in stiff systems is also an important reason for eliminating variables using linear invariants. This is discussed in Shampine [11]. (It should be noted that the choice of variable to be eliminated may seriously affect roundoff error because of the relative sizes of the variables. In many chemical kinetic problems, variables may differ by many orders of magnitude.)

We concur that it is generally better to eliminate variables. However, if they are eliminated, it must be remembered that the corresponding inequality invariants are not eliminated. Thus, in the example given in eqs. (1.4), we could use (1.5a) and (1.5b) to replace y_1 and y_4 with $k_1 - y_3$ and $(k_2 - y_2 + y_3)/2$ respectively, thus reducing the order of the system to two. However, the four inequality invariants, which originally were $y \geqq 0$ when y was a four-dimensional vector, remain as the two invariants $y \geqq 0$

[1] If implicit methods are used, the way in which the nonlinear equations are solved could affect the linear invariants. If iterative methods based on Newton are used, and if the approximation for the matrix $\partial f/\partial y$ satisfies $w^T(\partial f/\partial y) = 0$, the approximate solution to the nonlinear equations also satisifes the linear invariant within roundoff error. This follows from the fact that successive corrections Δy to the iterated solution satisfy an equation of the form $[I - \gamma(\partial f/\partial y)]\Delta y = f$, where f is related to f and satisfies eq. (2.2). Premultiplying this by w^T yields $w^T\Delta y = 0$. Since the initial iterate is obtained from an explicit integration method and satisfies the invariant in exact arithmetic, all subsequent iterates also satisfy it.

(since y is now a two-dimensional vector) and the two upper bounds

$$y_3 \leqq k_1, \tag{2.4a}$$

$$y_2 + y_3 \leqq k_2. \tag{2.4b}$$

If the numerical values of the variables are allowed to become large enough to violate these bounds, the solution could become unstable. Section 4 discusses a way to satisfy inequality invariants of this sort. There we will treat them as if all are of the form $y \geqq 0$, but the techniques extend to ones of the form of (2.4).

If it is not feasible to eliminate the linear equality invariants by order reduction, they can either be ignored, since they will be satisfied within roundoff error, or can be handled by the technique given in the next section for nonlinear equality invariants so that even roundoff error does not accumulate. However, if they are not eliminated but ignored, we may have to take them into account when satisfying other invariants, so that modifications made to satisfy the other invariants do not violate the ignored linear invariants.

3. Nonlinear equality invariants. As Shampine [11] points out, it is not usually practical to use nonlinear equality invariants to eliminate variables. He proposes projecting the solution of the differential equation (1.1) back onto the invariant after every step using a minimum norm perturbation to select the projection direction. If the stepsize control restricts the local error in each step to be less than the tolerance, it is not possible for the numerical solution to move further from the invariant than the tolerance in one step. Hence, a perturbation to the solution at the end of a step to satisy the invariant will not cause an error larger than the tolerance. (This argument is also used to prove convergence as the stepsize is reduced to zero, but it is applicable only to one-step methods.) However, for the above argument to be valid, the norms used to compute the minimum norm projection and to control the local error in the code should be the same. Therefore, he suggests that the user may want to change the error control in the code to use an L_2 norm (if it does not already) because it is easier to compute the minimum projection in the L_2 norm than in others.

We will give a modified system of equations with the same solution as the original system, which can be solved by performing a minimum L_2 norm projection onto the invariant after each step, regardless of the norm used in the code. This justifies the use of the process both in one-step and multistep codes.

Because eq. (1.2) is an invariant for the system, we have

$$\frac{\partial g}{\partial y} y' + \frac{\partial g}{\partial t} = 0 \tag{3.1}$$

along any solution trajectory, which implies the identity

$$Gf + g_t = 0 \tag{3.2}$$

for all y and t, where $G = \partial g/\partial y$ and $g_t = \partial g/\partial t$. To avoid excessive notation, we have not shown the explicit dependency of g and G on y, and t, and will not do so unless it is needed for clarity.

Let us consider a modification of eq. (1.1):

$$z' = f(z, t) + G^T\lambda. \tag{3.3}$$

We couple this with the invariant (1.2) applied as a constraint to z:

$$g(z, t) = g(z_0, t_0). \tag{3.4}$$

The variable λ is a type of Lagrange multiplier. If the number of constraints in eq. (3.4) is m, eqs. (3.3) and (3.4) form a system of $s + m$ equations for the s variables z and the m variables λ. These equations are differential-algebraic equations [7] of index 2 provided that the constraints (3.4) are linearly independent. The equations have the same solution as the original system, as can be seen by differentiating eq. (3.4) to get

$$Gz' + g_t = 0. \tag{3.5}$$

Substituting eq. (3.3) into this we get

$$Gf + g_t + GG^T\lambda = 0. \tag{3.6}$$

Using the identity (3.2) in this, we have

$$GG^T\lambda = 0. \tag{3.7}$$

If the m constraints (3.4) are linearly independent, we have $m \leqq s$ and G has full rank. Hence, GG^T is nonsingular, so eq. (3.7) implies that $\lambda = 0$ and a solution of eqs. (3.3) and (3.4) yields a $z(t)$ that is a solution of eq. (1.1). Conversely, any solution of (1.1) satisfies eq. (3.3) with $\lambda = 0$.

The relation between eqs. (3.3) and (3.4) on the one hand and eqs. (1.1) and (1.2) on the other can be viewed in several ways. λ can be viewed as a Lagrange multiplier which permits the invariants to be satisfied as constraints. Because they are automatically satisfied along the solution, the Lagrange multipliers are automatically zero. Another view is to start with the approach of Baumgarte ([1], [2], and [3]) in which a multiple of the amount by which the invariant (1.2) is not satisfied is added to the ODE (1.1) in order to force its solution to return to the invariant. Stability considerations show that G^TDg should be added, where D is a positive, diagonal matrix. He recommends choosing a specific functional of g for D which results in a predetermined exponential decay rate of the residual g. The approach taken here is to choose $\lambda = Dg$ continuously to force the solution onto the invariant at all times. In the discrete integration, values of λ are chosen separately at each time step. They can be thought of as impulses applied at each step to keep the solution on the invariant.

The reader may wonder if we have not made a simple problem more complex by increasing the number of variables and turning it into a differential/algebraic equation (DAE). Fortunately, this system of DAEs has a particularly simple form, and it has been shown in [6] that variable-stepsize, variable-order BDF methods are convergent for index 2 problems. It is also shown in § 4 that any multistep method convergent for ordinary differential equations is convergent for this particular index 2 problem when applied in the appropriate way. While similar results have not been developed for Runge–Kutta methods, it appears that implicit and explicit Runge–Kutta methods can also be used for these problems. (See Petzold [10] for a discussion of implicit Runge–Kutta methods applied to index 1 problems.)

If the BDF method were to be applied to (3.3) and (3.4), an implicit system of $s + m$ nonlinear equations would have to be solved at each step. If the ODE is not stiff, this is clearly not an efficient scheme if a Newton-like method is used for the solution of these equations. Instead, a predictor-corrector, nonstiff method such as an Adams method can be used for eq. (3.3) while a Newton method can be used for eq. (3.4); that is, at each corrector iteration we compute

$$y^{q+1} = y^q + h\beta_0(f(y^q, t) - y'^{q-1} + G^T\lambda^q), \tag{3.8}$$

$$y'^q = f(y^q, t) + G^T\lambda^q, \tag{3.9}$$

where λ^q is chosen so that the $(q+1)$st corrector iterate y^{q+1} satisfies eq. (3.4). This technique can be applied to any implicit multistep method. If only one corrector iteration is used, it is equivalent to performing a final perturbation to the result of a step in the direction G^T to satisfy eq. (3.4). If G is constant, this is exactly the minimum L_2 norm change needed to satisfy eq. (3.4). In practice, the local error must be small enough that G does not change appreciably during this projection step so that it is adequate to compute G immediately after the corrector step. (If G did change significantly over the projection, the differential equation almost certainly is in a region of potentially rapid change, the error estimation procedures are suspect, and the stepsize should be reduced.) For the same reason, one step of a Newton method is often adequate for computing the value of λ in (3.8) such that y satisfies (3.4). In this case, we simply solve the linear system

$$GG^T\lambda = -g(y, t),$$

where y is the value obtained from eq. (3.8) when λ is zero.

The technique could be applied at each stage of an explicit Runge-Kutta method, although it is questionable if the increased accuracy would justify the amount of additional work over doing it only to the final result of each step unless the problem was very sensitive to small deviations of the invariants (in which case it might well be stiff). If the problem is stiff, a Newton-like iteration must be used for the corrector, and the projection onto the invariant manifold can be folded into that iteration.

Since the theory tells us that the solution for λ should be zero, the local error in λ is precisely its value. The error control should measure the size of λ in whatever norm is used in the code along with the estimates of the local errors in the other components.

As Shampine [11] points out, the corrections to the solution to satisfy the nonlinear invariants could destroy the linear invariants if they have not been removed by reduction of order. While it is possible to solve this problem by treating the linear invariants along with the nonlinear invariants, some of this work can be avoided by modifying the direction in which the solution is projected onto the nonlinear invariants so that it lies in the space spanned by the linear invariants. The linear invariants have the form

$$Wy = k, \tag{3.10}$$

where W is a constant matrix. If these are treated simultaneously with the nonlinear invariants, we will have to add the additional term $W^T\kappa$ to eq. (3.3) so that κ and λ can be chosen to satisfy both (3.4) and (3.10). However, we have already noted that eq. (3.10) is satisfied automatically by the computed solution before it is projected into the nonlinear invariant manifold. Hence, any changes made to y, say Δy, should satisfy

$$W\Delta y = 0. \tag{3.11}$$

The change to y has the form

$$\Delta y = (W^T\kappa + G^T\lambda). \tag{3.12}$$

It is sufficient to satisfy the nonlinear invariant (3.4) and (3.11) simultaneously. Let us write the matrix $B = [W^T, G^T]$, as $\tilde{B}R$ where R is an upper triangular matrix (computed by the Gram-Schmidt process) so that the columns of $\tilde{B} = [\tilde{W}^T, \tilde{G}^T]$ are orthogonal, and set $[\tilde{\kappa}^T, \tilde{\lambda}^T]^T = R[\kappa^T, \lambda^T]^T$. Using these relations in eq. (3.12) we get

$$\Delta y = (\tilde{W}^T\tilde{\kappa} + \tilde{G}^T\tilde{\lambda}). \tag{3.13}$$

Applying eq. (3.11) to this, and noting that $\tilde{G}$ is orthogonal to $\tilde{W}$ and hence to W, we find that $\tilde{\kappa} = 0$ and

$$\Delta y = \tilde{G}^T \tilde{\lambda}. \tag{3.14}$$

Therefore, we can compute the correction, Δy, that satisfies the linear invariants by orthogonalizing the rows of G against the rows of W to get $\tilde{G}$ and then projecting the solution into the nonlinear invariant manifold (3.4) using a projection in the space spanned by the rows of $\tilde{G}$. Since W is a constant matrix, its rows can be orthonormalized prior to the integration to get $\tilde{W}$, and G can be orthonormalized against $\tilde{W}$ at each step.

4. Convergence of multistep methods for special index 2 problems. A multistep method is not convergent for index 2 problems unless it satisfies many restrictions, which, for example, rule out Adams methods (see März [9] for a treatment of multistep methods on index 1 problems). However, it is possible to use any stable implicit multistep method for eqs. (3.3) and (3.4) provided the solution for λ is zero. They can be applied in the form

$$\sum_{i=0}^{k} (\alpha_i z_{n-i} + h\beta_i f(z_{n-i}, t_{n-i})) + h\beta_0 G_n^T \lambda_n, \tag{4.1}$$

$$g(z_n, t_n) = 0, \tag{4.2}$$

where λ_n is chosen so that eq. (4.2) is satisfied. The important difference between this technique and the application of multistep methods to the general index 2 problem is that in the application of the multistep method at the nth step, z'_{n-i} is replaced with $f(z_{n-i}, t_{n-i})$ for $i \geqq 1$, but with $f(z_n, t_n) + G_n^T \lambda_n$ when $i = 0$ to get eq. (4.1). This does not affect the order of the method because we are using the true solution for past values of λ rather than computed values.

THEOREM 1. *Suppose that a variable-order, variable-step, multistep method is scaled so that* $\beta_0 = 1$, *its truncation error is defined as*

$$\tau_n = \frac{1}{h_n} \sum_{i=0}^{k} (\alpha_i z(t_{n-i}) + h_n \beta_i z'(t_{n-i})), \tag{4.3}$$

the maximum roundoff or other errors in solving eqs. (4.1) *and* (4.2) *are* η, *and the initial error in* z *is* e_0. *If*

(*a*) *the method is convergent for ordinary differential equations,*
(*b*) $\tau_n = O(h)$ *where* $h = \max(h_n)$,
(*c*) $\eta = O(h^2)$,
(*d*) $e_0 = O(h)$,
(*e*) *the* β *coefficients (after the scaling so that* $\beta_0 = 1$*) are bounded,*
(*f*) $f(z,t)$ *is Lipschitz continuous,*
(*g*) $[GG^T]^{-1}$ *exists and is bounded,*

then the solution of eqs. (4.1) *and* (4.2) *is an* $O(\tau + \eta/h + e_0)$ *approximation to the solution of eqs.* (3.3) *and* (3.4).

Proof. The proof of Theorem 2.2 in [6] can be used with minor modifications (after adjusting for the change of notation—x and y in the reference have been replaced by z and λ here). When the equivalent of eq. (2.20a) of [6] is developed, it has the form

$$\frac{1}{h_n} \rho_n e^z + \sigma_n f_z e^z + G_z^T \lambda_n e_n^z + G_n^T e_n^\lambda = \frac{\eta_n}{h_n} + \tau_n \tag{4.4}$$

where σ_n is the difference operator based on the β coefficients. $\tilde{G}_{11}$ of reference [6] has been replaced by $\sigma_n f_z + G_z^T \lambda_n$. It is necessary to bound this quantity in order to apply the previous proof. The first term of the expression for $\tilde{G}_{11}$ is bounded by the hypotheses. Since the true solution for λ is zero, $\lambda_n = e_n^\lambda$. Any bound on e^λ will allow the proof to continue and show that e^λ is $O(h)$, so the contraction mapping theorem can be used to complete the result.

5. Inequality invariants. If we have the inequality invariants (1.3) and wish to enforce them on the numerical solution, we can use a similar approach. In this case we modify eq. (1.1) only if $b(y, t)$ is negative. The resulting equation is

$$y' = f(y, t) + G^T\{\min(b_i(y, t), 0)\lambda_i\}. \tag{5.1}$$

The value of λ_i is irrelevant if $b_i(y, t)$ is non-negative. We can think of these as Lagrange forces applied only when the inequality is about to be violated. (For a problem in which inequality constraints rather than invariants are enforced, see Lötstedt [8]). If it is also necessary to preserve equality invariants, this can be combined with the techniques of the previous sections.

Although this method provides a theoretical basis for projecting the computed solution back into the invariant spaces, it seems possible that the computed values of λ may be discontinuous, and that this could affect the error control in a multistep method (but not in a one-step method if the error control is locally computed in each step). In the equations of chemical kinetics and related equations, we often wish to use multistep methods because the equations usually are stiff with real eigenvalues and the BDF methods have proved to be the most efficient methods for that case. For this reason, we are going to suggest an alternate scheme that guarantees positive solutions for the special class of problems (1.8). Suppose we apply a multistep method to (1.8). We get

$$y_n = h\beta_0(-Dy_n + q) + \Sigma, \tag{5.2}$$

where $D = D(y_n, t_n)$, $q = q(y_n, t_n)$, and Σ is the sum of prior values and derivatives. This is a nonlinear system to be solved for y_n. First note that $\gamma = h\beta_0$ is positive. We will prove that eq. (5.2) has a positive solution provided Σ is positive. We will delay the proof of this result until we have examined its consequences. The result means that we can force the nonlinear equation solver to locate a positive solution for eq. (5.2), provided we can guarantee that Σ is positive, Σ is a polynomial prediction of the value of $y_n - h\beta_0 y_n'$. For small enough h it will be positive. For example, for the backward Euler method, $\Sigma = y_{n-1}$ so it is always positive. Therefore, for this class of problems, we recommend testing Σ to see if it is positive at the start of each step, and if not, reducing the stepsize appropriately. Then, the users' favorite iteration can be used to compute $\Delta y^{(m)}$. To keep the solution positive, the new iterate should be set to $y^{(m+1)} = \max(y^{(m)} + \Delta y^{(m)}, 0)$, but the convergence test should be done on $\Delta y^{(m)}$.[2] Unfortunately, it does not seem possible to show that forcing the iterates to stay in the positive quadrant forces convergence to a positive solution with the hypotheses given, so it may be necessary to reject steps if they get stuck at the boundary of the positive quadrant and reduce the stepsize for a retry, or to move away from the boundary to look for the solution.

It remains to show

THEOREM 2. *If*

D is a diagonal matrix,

[2] I am grateful to Linda Petzold for this suggestion.

D and q are non-negative, continuous functions of y_n,
q is bounded,
Σ *is non-negative,*
$\gamma = h\beta_0$ *is positive,*
then eq. (5.2) *has a positive solution* y_n.

Proof. Consider eq. (5.2) as a set of single equations. As a function of y_{ni} (the second subscript referring to the ith component of y_n), the ith equation can be written as

$$(1 + \gamma D_i)y_{ni} - \Sigma_i - q_{ni} = 0. \tag{5.3}$$

The left-hand side is negative when $y_{ni} = 0$ and positive when

$$y_{ni} = (\Sigma_i + \max_y q_{ni}).$$

This is sufficient to guarantee a positive solution of eq. (5.2). by

LEMMA. *If the nonlinear equations* $f(x) = 0$ *are such that f is a continuous function of x from the closed s-dimensional unit hypercube* $0 \leqq x \leqq 1$ *into* $\boldsymbol{R}^s$, *and if, for all i,* $f_i(x) < 0$ *when* $x_i = 0$ *and* $f_i(x) > 0$ *when* $x_i = 1$, *then the equation has a solution in the unit hypercube, H.*

Proof. We will construct a function $F(x) = x + Df(x)$, where D is a nonsingular diagonal matrix (unrelated to the matrix in Theorem 2), such that F is a continuous map of the H into itself. Under the metric

$$\rho(x, y) = \max_i |x_i - y_i|,$$

H is a ball and the Brouwer fixed-point theorem is applicable to $F(x) = x$, thus proving the existence of at least one zero of f in H. D is constructed in two stages. First let D_1 contain as its ith diagonal element the inverse of the maximum of $|f_i(x)|$ over H. Thus, $g(x) = D_1 f(x)$ satisfies $|g(x)| \leqq 1$ and also satisfies all of the hypotheses of f. Define

$$\delta_i = \min_x (x_i, 1 - x_i \; g_i(x) = 0).$$

Because H is closed, $\delta_i > 0$ (it is the minimum distance from the i-dimension faces of H to a zero of f or g). Now define the ith diagonal of D_2 to be $-\delta_i$ and let $D = D_1 D_2$. Note that the ith component of $F(x)$ is $x_i - \delta_i g_i(x)$, which lies in [0,1] if x is in H. Hence $F(x)$ maps H into itself and the lemma is proven.

In the case of chemical kinetic equations, we do not immediately satisfy the hypothesis that q is bounded because q is normally a polynomial equation of the unknowns with positive coefficients (see eqs. (1.4), for example, where q consists of exactly those terms with positive coefficients and $-Dy_n$ consists of exactly those terms with negative coefficients). However, we can still arrive at the result. We have already noted that the linear invariant (1.5c) coupled with the nonnegativity of the variables implies an upper bound on the variables. This is only a bound on the solution of the ODE, not necessarily on the numerical approximation. However, we will show that there is a numerical solution of eq. (5.2) that satisfies this upper bound also. Let $q_{\max}$ be the vector of the largest values that can be assumed by the components of q over the region of possible values of y. Replace q in eq. (5.2) with $\tilde{q}$, where $\tilde{q}$ is equal to q in those components which are less than the maximum, and equal to $q_{\max}$ elsewhere. The modified problem satisfies the hypotheses of Theorem 2, therefore it has a positive solution. Since this solution satisfies the linear invariant (1.5c), its components must

satisfy the same bounds as the true solution. Hence, at the computed solution y_n, $\tilde{q}(y_n) = q(y_n)$, so that y_n is a positive numerical solution of the original eq. (5.2).

Acknowledgments. I am grateful to the referees for many helpful suggestions.

REFERENCES

[1] J. BAUMGARTE, *Stabilization of constraints and integrals of motion in dynamical systems*, Comput. Meth. in Appl. Mech. Engrg., 1 (1972), pp. 1-16.

[2] J. BAUMGARTE, *Asymptotische Stabilisierung von Integralen bei gewohnlichen Differentialgleichungen I. Ordnung*, Z. Angew. Math. Mech., 53 (1973), pp. 701-704.

[3] J. BAUMGARTE AND E. STIEFEL, *Examples of transformations improving the numerical accuracy of the integration of differential equations*, in Lecture Notes in Mathematics 362, Springer-Verlag, New York, 1974, pp. 207-236.

[4] L. EDSBERG, *Integration package for chemical kinetics*, in Stiff Differential Systems, R. A. Willoughby, ed., Plenum Press, New York, 1973.

[5] W. H. ENRIGHT, T. E. HULL AND B. LINDBERG, *Comparing numerical methods for stiff systems of ODEs*, BIT 15 (1975), pp. 10-49.

[6] C. W. GEAR, G. K. GUPTA AND B. LEIMKUHLER, *Automatic integration of Euler-Lagrange equations with constraints*, J. Comput. Appl. Math., 12 & 13 (1985), pp. 77-90.

[7] C. W. GEAR, AND L. R. PETZOLD, *ODE methods for the solution of differential/algebraic equations*, SIAM J. Numer. Anal., 21 (1985), pp. 716-728.

[8] P. LÖTSTEDT, *Numerical simulation of time-dependent contact and friction problems in rigid body mechanics*, this Journal, 5 (1984), pp. 370-393.

[9] R. MÄRZ, *Multistep methods for initial value problems in implicit differential equations*, Humboldt-Universität zu Berlin, preprint 22, 1981.

[10] L. R. PETZOLD, *Order results for implicit Runge-Kutta methods applied to differential/algebraic systems*, Sandia Report SAND84-8838, Sandia National Laboratories, Livermore, CA, Aug. 1984.

[11] L. F. SHAMPINE, *Conservation laws and the numerical solution of ODEs*, Sandia Albuquerque Report SAND84-1241, June 1984.

SIAM J. SCI. STAT. COMPUT.
Vol. 7, No. 3, July 1986

004

AN UPWIND SECOND-ORDER SCHEME FOR COMPRESSIBLE DUCT FLOWS*

MATANIA BEN-ARTZI† AND JOSEPH FALCOVITZ‡

Abstract. An upwind second-order scheme is proposed for time-dependent, inviscid, compressible fluid flow in one space dimension and variable cross-section. The basic ingredient of the method is an analytic solution of the corresponding GRP (Generalized Riemann Problem), leading to explicit expressions for the fluxes. Sonic lines are given a special treatment. The only accessory technique used is a simple monotonicity algorithm. Results of two test problems are shown: (a) the cylindrical converging shock problem, where boundary conditions at $r=0$ are shown to be natural extensions of the analytic method; (b) a quasi 1-D nozzle problem.

Key words. compressible duct flow, second-order scheme, Godunov scheme, high resolution, Lagrangian, Eulerian

AMS(MOS) subject classifications. 65M05, 76L05

1. Introduction. Consider the Euler equations that model the time-dependent flow of an inviscid, compressible fluid through a duct of smoothly varying cross-section. Denoting by $A(r)$ the area of the cross-section at r, these equations are,

$$A\frac{\partial}{\partial t}U+\frac{\partial}{\partial r}[AF(U)]+A\frac{\partial}{\partial r}G(U)=0,$$

$$U=\begin{pmatrix}\rho\\ \rho u\\ \rho E\end{pmatrix},\quad F(U)=\begin{pmatrix}\rho u\\ \rho u^2\\ (\rho E+p)u\end{pmatrix},\quad G(U)=\begin{pmatrix}0\\ p\\ 0\end{pmatrix}, \tag{1.1}$$

where ρ, p, u are, respectively, density, pressure and velocity, $E=e+\frac{1}{2}u^2$ is the total specific energy (with e being the internal specific energy), and an equation-of-state of the form $p=p(e,\rho)$ is assumed.

In this paper we derive an upwind second-order scheme for the time integration of (1.1), imposing no assumptions on the form of $A(r)$. Thus, our generic name "duct flows" covers indeed all 1-D flows, including, for example, cylindrical and spherical flows with radial symmetry. The present paper is an extension of our previous work [2] which treated the case $A\equiv 1$ (slab symmetry).

To give the basic idea, suppose that we use equally spaced grid-points $r_i=i\Delta r$ along the r-axis and equal time intervals of size Δt. By "cell i" we shall refer to the interval extending between the "cell boundaries" $r_{i\pm 1/2}=(i\pm\frac{1}{2})\Delta r$. We let Q_i^n denote the average value of the quantity Q over cell i at time $n\Delta t$. Similarly, we designate by $Q_{i+1/2}^{n+1/2}$ the value of Q at the cell boundary $r_{i+1/2}$, averaged over the time interval $\cdot\,(n\Delta t, (n+1)\Delta t)$. Generally speaking, a difference scheme for (1.1) is given by,

$$\begin{aligned}U_i^{n+1}-U_i^n=&-\frac{\Delta t}{\Delta V_i}[A(r_{i+1/2})F(U)_{i+1/2}^{n+1/2}-A(r_{i-1/2})F(U)_{i-1/2}^{n+1/2}]\\ &-\frac{\Delta t}{\Delta r}[G(U)_{i+1/2}^{n+1/2}-G(U)_{i-1/2}^{n+1/2}],\end{aligned} \tag{1.2}$$

* Received by the editors March 13, 1984, and in revised form January 18, 1985.

† Department of Mathematics, Technion-Israel Institute of Technology, Haifa 32000 Israel. Present address, Department of Mathematics, University of California, Berkeley, California 94720.

‡ Computation Division, Rafael Ballistic Center, P.O. Box 2250, Haifa 31020 Israel.

where $\Delta V_i = \int_{r_{i-1/2}}^{r_{i+1/2}} A(r)\,dr$. In this scheme one must still give an appropriate interpretation to the "flux" values $F(U)_{i+1/2}^{n+1/2}$, $G(U)_{i+1/2}^{n+1/2}$. A fundamental way of doing it, and at the same time obtaining the correct amount of "up-winding", was proposed by Godunov [7] and may be described as follows. Assume that at time t_n the point $r_{i+1/2}$ separates the two *uniform* states U_i^n, U_{i+1}^n. This now constitutes a simplified initial value problem for (1.1), namely, the *Riemann Problem* (RP). The subsequent solution for $t > t_n$ consists of several centered waves emerging from the initial discontinuity and separating regions of smooth flow. Let $R_E(r/t;\, U_-, U_+)$ be the solution of the Riemann problem in the planar case ($A \equiv 1$), subject to the initial condition $U(r, 0) = U_\pm$ according to $\pm r > 0$. In writing r/t we use the well-known "self-similarity" of the solution in this case. The Godunov scheme is now obtained by setting,

$$
\begin{aligned}
&U_{i+1/2}^n = R_E(0;\, U_i^n, U_{i+1}^n),\\
&F(U)_{i+1/2}^{n+1/2} = F(U_{i+1/2}^n), \qquad G(U)_{i+1/2}^{n+1/2} = G(U_{i+1/2}^n).
\end{aligned}
\tag{1.3}
$$

As is well known, inserting (1.3) in (1.2) results in a first-order scheme. In order to upgrade this order of accuracy and achieve better resolution of discontinuities we assume that the values of U in cell i at time $t = t_n$ are *linearly* distributed and retain the notation U_i^n for the average values. Let $(\Delta U)_i^n$ be the variation of U over cell i. Thus $U_+ = U_{i+1}^n - \frac{1}{2}(\Delta U)_{i+1}^n$ and $U_- = U_i^n + \frac{1}{2}(\Delta U)_i^n$ are the limiting values (from right and left respectively) of U at the discontinuity $r = r_{i+1/2}$. Assume now for simplicity that

$$
r_{i+1/2} = 0, \qquad t_n = 0.
$$

The two linearly distributed states on both sides of the discontinuity are given by

$$
U_+(r) = U_+ + \frac{r}{\Delta r} \cdot (\Delta U)_{i+1}^n, \qquad r > 0,
$$

$$
U_-(r) = U_- + \frac{r}{\Delta r} \cdot (\Delta U)_i^n, \qquad r < 0.
$$

Due to the nonuniformity of the initial data, as well as the variable cross-section, the flow in the neighborhood of the singularity does not possess the "self-similarity" features of the Riemann problem. Thus, characteristic curves and discontinuity trajectories are no longer straight lines, and they do not carry constant values of the flow variables. Hence, in order to evaluate $U_{i+1/2}^{n+1/2}$ with second-order accuracy we need not only the "limiting value" $U_{i+1/2}^n$ (reflecting the immediate values of the variables after the resolution of the discontinuity) but also its rate of change in time. We are now led naturally to the following statement of the *Generalized Riemann Problem* (GRP).

Given two linearly distributed states $U_+(r)$, $U_-(r)$, let

$$
U(r, 0) = \begin{cases} U_+(r), & r > 0, \\ U_-(r), & r < 0 \end{cases}
$$

be the initial condition for (1.1). *Let $R(r, t;\, U_\pm(r))$ be the solution for $t > 0$. Find*

$$
\begin{aligned}
&R(0, 0;\, U_\pm(r)) = \lim_{t \to 0+} R(0, t;\, U_\pm(r)),\\
&\frac{\partial R}{\partial t}(0, 0;\, U_\pm(r)) = \lim_{t \to 0+} \frac{\partial R}{\partial t}(0, t;\, U_\pm(r)).
\end{aligned}
\tag{1.4}
$$

Turning back to the scheme (1.2), the GRP solution may now be used to obtain a second-order algorithm which replaces (1.3) as follows:

$$
\begin{aligned}
&U^n_{i+1/2} = R(0,0;\, U_\pm(r)),\\
&\left[\frac{\partial}{\partial t}U\right]^n_{i+1/2} = \frac{\partial R}{\partial t}(0,0;\, U_\pm(r)),\\
&\left[\frac{\partial}{\partial t}F(U)\right]^n_{i+1/2} = F'(U^n_{i+1/2})\left[\frac{\partial}{\partial t}U\right]^n_{i+1/2},\\
&\left[\frac{\partial}{\partial t}G(U)\right]^n_{i+1/2} = G'(U^n_{i+1/2})\left[\frac{\partial}{\partial t}U\right]^n_{i+1/2},\\
&F(U)^{n+1/2}_{i+1/2} = F(U^n_{i+1/2}) + \frac{\Delta t}{2}\left[\frac{\partial}{\partial t}F(U)\right]^n_{i+1/2},\\
&G(U)^{n+1/2}_{i+1/2} = G(U^n_{i+1/2}) + \frac{\Delta t}{2}\left[\frac{\partial}{\partial t}G(U)\right]^n_{i+1/2}.
\end{aligned}
\tag{1.5}
$$

We have used the notation $F'(U)$, $G'(U)$ for the Jacobian matrices of F, G, respectively, with respect to U.

The main building block in our scheme is an analytic derivation of the solution to the GRP, *as formulated above.*

Once this solution is established, the procedure (1.2), (1.5) is set up, yielding a second-order accurate upwind scheme for (1.1). Let us observe some features of this scheme.

(a) A Riemann problem is solved once per cell per time-step. It determines completely the initial wave structure which serves as a basis for all subsequent calculations.

(b) A "plug-in" procedure, using the explicit analytic solution, is used in order to compute time-derivatives of flow quantities. Cell averages are then updated according to (1.2), (1.5), and the new slopes are obtained by a straightforward differencing of cell-boundary values at t_{n+1}.

(c) The only accessory technique used is a simple monotonicity algorithm. In particular, no other dissipative mechanism is incorporated.

Further details of the numerical scheme are practically identical to those used in the planar case [2] and will not be repeated here. Thus, we concentrate in the present paper on the analytic solution of the GRP and in doing so, we impose the following basic hypothesis.

Given the initial data for the GRP, define the "*associated RP*" as the Riemann problem for the *planar* case ($A \equiv 1$), with uniform initial states $U_\pm = \lim_{r\to 0\pm} U_\pm(r)$. We then assume that locally the wave pattern for the GRP is the same as that of the associated RP, meaning that if the RP solution involves a shock travelling to the left, then so does the GRP solution, etc. Furthermore, the two solutions converge to the same values at the singularity. Using the notation R_E, R introduced above for the solutions of these problems, the last statement implies

$$
\lim_{t\to 0+} R(r,t;\, U_\pm(r)) = R_E(p;\, U_-,\, U_+) \quad \text{along } \frac{r}{t} = p = \text{const.} \tag{1.6}
$$

Observe, however, that some features of the solution to the associated RP are lost in the case of the GRP, even locally. Thus, the solution is no longer self-similar, characteristic curves are not straight lines and centered rarefaction waves are not necessarily isentropic. In particular, this applies also to the Riemann problem (i.e., uniform states on both sides) with variable cross-section ($A(r) \neq$ const.).

Our approach to the solution of the GRP is based on a simple "propagation of singularities" argument, i.e., the fact that along a characteristic curve the jump in the highest-order transversal derivative satisfies an ordinary differential equation [6, § V.1]. To be more specific, suppose that we deal with a Γ^--rarefaction wave, one that consists of Γ^--characteristics (slope $u-c$) fanning out from the singularity. The Γ^+ characteristics cross the wave transversally and the discontinuity itself may be regarded as a degenerate Γ^+ curve, denoted as Γ_0^+. It turns out that the derivative of $R(r, t; U_+(r))$ in the Γ^- direction satisfies a simple ODE "along" Γ_0^+. The equation reflects both the nonuniformity of the state ahead of the rarefaction (i.e. $U_-(r)$) and the geometrical factor introduced by the variation of $A(r)$. While our final goal is an "Eulerian" solution (in terms of r, t), we shall find it convenient to carry out the first steps in terms of a "local" Lagrangian coordinate ξ. Such a representation, of course, guarantees that the contact discontinuity lies along $\xi = 0$ while the $\Gamma^\pm$-waves propagate in $\pm\xi$-directions. The difficult problem posed by a "sonic" grid-line (i.e., when $r = 0$ is contained in a rarefaction fan) is bypassed and relegated to a later stage.

Let us review briefly the plan of the paper. Basic notation and assumptions are listed in § 2, along with the local Lagrangian formulation. In § 3 we analyze the structure of a centered rarefaction wave and give our main analytical result of this work. This result is combined in § 4 with the Rankine–Hugoniot relations to give the derivatives of flow quantities along the contact discontinuity. At this stage we have all the information needed in order to establish a Lagrangian scheme, analogous to (1.2). Indeed, such a scheme would be very close to the one proposed by van Leer [12] and was actually worked out in [2]. However, we shall not pursue it further here. In § 5 we switch back to the full Eulerian treatment and discuss the "nonsonic" case, which follows easily from the Lagrangian case. The "sonic" case is taken up in § 6. The grid-line is now contained in a (curvilinear) rarefaction fan and must be determined in terms of characteristic coordinates. To compute the limits (1.4) in this case we need a careful inspection of the geometry of the characteristic curves near the singularity.

Finally, we illustrate the method in § 7 by two numerical examples which are very different in nature: (a) a converging cylindrical shock; (b) a steady state solution to a duct flow problem. In the context of the first example we must discuss, of course, the boundary conditions at $r = 0$. It is shown that only a simple adjustment is needed in order to implement the analytic solution at the singularity. We emphasize that both problems were solved using exactly the same scheme as described above.

2. Notation and Lagrangian formulation. Here and in the sequel we refer to the Generalized Riemann Problem (GRP) as stated in the previous section (see (1.4)).

In addition to the basic flow variables appearing in (1.1), we shall also make extensive use of the speed of sound c and the "Lagrangian" speed of sound $g = \rho c$. We shall carry out the analysis in terms of a general equation of state. The results are particularly simple when applied to the γ-law equation,

$$(2.1)_\gamma \qquad p = (\gamma - 1)\rho e, \qquad \gamma > 1.$$

We shall always give the explicit form of the formulae for this special case and add the label "γ" to the numbers of such equations (as has already been done in $(2.1)_\gamma$).

Initially $U(r,0)=U_{\pm}(r)$ is piecewise linear with a jump discontinuity at $r=0$. We let $U(r,t)=R(r,t;U_{\pm}(r))$ be the solution of (1.1). Using the initial density distribution we define the Lagrangian "mass" coordinate by

$$d\xi = A\rho\, dr, \qquad \xi(0)=0. \tag{2.2}$$

In terms of ξ, t, the equations (1.1) take the form

$$\begin{gathered}\frac{\partial}{\partial t}V+\frac{\partial}{\partial \xi}(A\Phi(V))+A\frac{\partial}{\partial \xi}\Psi(V)=0,\\ V=\begin{pmatrix}\tau\\ u\\ E\end{pmatrix},\quad \Phi(V)=\begin{pmatrix}-u\\ 0\\ pu\end{pmatrix},\quad \Psi(V)=\begin{pmatrix}0\\ p\\ 0\end{pmatrix},\quad \tau=\frac{1}{\rho}.\end{gathered} \tag{2.3}$$

We assume that initially $V(\xi,0)=V_{\pm}(\xi)$ is piecewise linear with a jump at $\xi=0$. This is justified by the fact (to be proved later) that $\partial U(0,0)/\partial t$ (in the sense of (1.4)) depends only upon the *limiting* values (*as* $r\to\pm 0$) of $\partial U(r,0)/\partial r$ and $U(r,0)$. Hence we may replace r-derivatives by ξ-derivatives using (2.2). Letting $V_{\pm}=\lim_{\xi\to\pm 0}V_{\pm}(\xi)$ we denote by $R_L(p;V_-,V_+)$, $p=\xi/t$, the (Lagrangian) solution to the "associated RP". Clearly, the line $\xi=0$ represents the contact discontinuity.

We shall employ the following notation conventions.

Subscripts "r, l" denote limiting values as $\xi\to 0+$, $0-$, respectively.

An asterisk (*) is used for values at $t=0+$ along $\xi=0$ (along with "r, l" for discontinuous quantities).

We shall use the subscript "0" for the limiting value as $t\to 0+$ along $r=0$ (i.e., in the Eulerian framework).

Further details are given in Table 1, where Q stands for any one of the flow variables (see also Fig. 1).

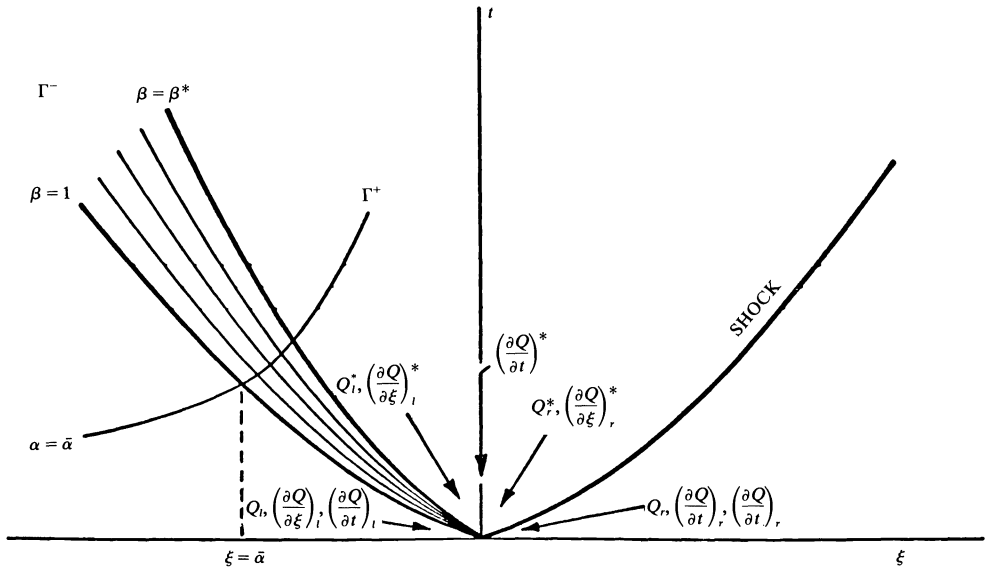


FIG. 1 *Wave pattern and related quantities, in conjunction with Table* 1, § 2.

TABLE 1
Notation.

Symbol	Definition
Q_r, Q_l	$\lim Q(\xi, 0)$ as $\xi \to 0+, 0-$.
$\left(\frac{\partial Q}{\partial \xi}\right)_r, \left(\frac{\partial Q}{\partial \xi}\right)_l$	Constant slopes for $\xi > 0$, $\xi < 0$.
$R_L(p; V_-, V_+)$	Lagrangian solution of the "associated RP", $p = \xi/t$.
$R_E(p; U_-, U_+)$	Eulerian solution of the "associated RP", $p = r/t$.
V^*	$= R_L(0; V_l, V_r)$.
Q_r^*, Q_l^*	Right and left values for Q discontinuous across $\xi = 0$. ($Q = \rho, c, g = \rho c$.)
$\left(\frac{\partial Q}{\partial t}\right)^*$	$= \lim_{t \to 0+} \frac{\partial}{\partial t} Q(\xi, t)$ at $\xi = 0$.
$\left(\frac{\partial Q}{\partial t}\right)_r^*, \left(\frac{\partial Q}{\partial t}\right)_l^*$	Right and left values of $\left(\frac{\partial Q}{\partial t}\right)^*$ for discontinuous Q.
$\left(\frac{\partial Q}{\partial \xi}\right)_r^*, \left(\frac{\partial Q}{\partial \xi}\right)_l^*$	$= \lim_{t \to 0+} \lim_{\xi \to 0+} \frac{\partial}{\partial \xi} Q(\xi, t), \lim_{t \to 0+} \lim_{\xi \to 0-} \frac{\partial}{\partial \xi} Q(\xi, t)$.
$\left(\frac{\partial Q}{\partial t}\right)_r, \left(\frac{\partial Q}{\partial t}\right)_l$	$= \lim_{\xi \to 0+} \frac{\partial}{\partial t} Q(\xi, t)\Big\|_{t=0}, \lim_{\xi \to 0-} \frac{\partial}{\partial t} Q(\xi, t)\Big\|_{t=0}$.
U_0	$= R_E(0; U_l, U_r)$
$\left(\frac{\partial Q}{\partial t}\right)_0$	$= \lim_{t \to 0+} \frac{\partial}{\partial t} Q(r, t)$ at $r = 0$.

Remark 2.1. Note that if two limiting processes are implied, they must be carried out in the indicated order. For example, $(\partial Q/\partial \xi)_r^*$ means that first the ξ-derivative is evaluated at $\xi = 0+$ and its limit is then taken as $t \to 0+$. On the other hand, $(\partial Q/\partial t)_r$ is computed by first taking the t-derivative at $t = 0$ and then letting $\xi \to 0+$.

Also note the meaning of the various groups of quantities:

Q_r, Q_l, $(\partial Q/\partial \xi)_r$, $(\partial Q/\partial \xi)_l$ are the given initial data;

Q_0, Q^*, Q_r^*, Q_l^* result from the solution of the associated RP;

$(\partial Q/\partial t)_l$, $(\partial Q/\partial t)_r$ are time derivatives evaluated ahead of the waves and can therefore be expressed in terms of initial spatial derivatives in view of (1.1), (2.3);

$(\partial Q/\partial t)_0$, $(\partial Q/\partial t)^*$, $(\partial Q/\partial \xi)_r^*$, $(\partial Q/\partial \xi)_l^*$ result from the solution of the GRP.

3. Resolution of a centered rarefaction wave (CRW). It is a well-known fact [5] that a jump discontinuity in the initial data (for compressible time-dependent flow) is resolved in terms of three types of waves, namely, contact discontinuities, shocks and centered rarefaction waves. The first two are trajectories (in the (r, t) plane) along which certain "jump conditions" are satisfied. In the context of the GRP these conditions allow for a straightforward treatment of these singularities, as we show in the next section. However, the centered rarefaction wave (CRW) is more difficult to handle. It consists of a family of characteristic curves fanning out from the singularity and covering a "triangular" area in the (r, t) plane. Due to the nonuniformity of the data, those characteristic curves are not straight lines and do not carry constant values of the flow variables. Thus, the features which facilitate the solution of the Riemann

problem in the uniform case are lost and we are forced to take a closer look at the behavior of flow variables near the singularity, in the region covered by the CRW.

This section is devoted to the study of the CRW, which constitutes the main analytical tool used in this work. The basic idea is to use characteristic coordinates throughout the CRW, so that the singularity is "blown up" into a "full segment" in the characteristic plane, where the CRW is now represented by a rectangular zone. The base of this rectangle corresponds to the singularity. The flow variables are smooth in the rectangle. Their values at the base are related to a Riemann solution while their first derivatives normal to the base correspond to directional derivatives in the (r, t) plane. It turns out that these derivatives satisfy simple differential equations along the base. Thus, knowing the derivatives at one endpoint enables us to determine their values at the other (a "propagation of singularities" argument).

To fix the ideas we shall assume henceforth that the wave configuration is as displayed in Fig. 1. (Recall that the nature of the wave pattern is already determined by the associated RP.) In this and the next section we work solely with the Lagrangian formulation (2.2), (2.3).

Let $\Gamma^{\pm}$ be the characteristic families of (2.3), corresponding to the eigenvalues $\pm gA = \pm A\rho c$. At the singularity, the slopes of Γ^- curves extend from $-g_l A(0)$ at the head to $-g_l^* A(0)$ at the tail of the rarefaction. Let us use characteristic coordinates (α, β) such that $\alpha = \text{const.}$ (resp. $\beta = \text{const.}$) corresponds to a Γ^+ (resp. Γ^-) curve and such that,

$$\begin{aligned} &\beta = \text{normalized slope of } \Gamma^- \text{ at the origin,}\\ &\beta = 1 \text{ at the head characteristic,}\\ &\alpha = \text{value of } \xi \text{ at intersection point of } \Gamma^+\\ &\qquad \text{with the curve } \beta = 1. \end{aligned} \tag{3.1}$$

In particular, the definition (3.1) implies,

$$\xi(\alpha, 1) = \alpha, \qquad g(0, \beta) = g_l \beta. \tag{3.2}$$

Thus, the CRW is parametrized by $(\alpha, \beta) \in [-\alpha_0, 0] \times [\beta^*, 1]$, where $\alpha_0 > 0$ is fixed and $\beta^* = g_l^* / g_l \leqq 1$. Any variable Q defined in this domain is represented as $Q(\alpha, \beta)$ (and this includes the coordinates ξ, t themselves). It follows readily from the definition of α, β that the characteristic curves satisfy the equations,

$$\begin{aligned} \frac{\partial \xi}{\partial \alpha} &= -gA \frac{\partial t}{\partial \alpha},\\ \frac{\partial \xi}{\partial \beta} &= gA \frac{\partial t}{\partial \beta}. \end{aligned} \tag{3.3}$$

Observe that here $A = A(r) = A(r(\xi, t)) = A(\alpha, \beta)$. In what follows we set $A'(r) = dA/dr$. The characteristic relations along $\Gamma^{\pm}$ are

$$\begin{aligned} &\frac{\partial p}{\partial \alpha} - g \frac{\partial u}{\partial \alpha} + \frac{ucgA'}{A} \frac{\partial t}{\partial \alpha} = 0,\\ &\frac{\partial p}{\partial \beta} + g \frac{\partial u}{\partial \beta} + \frac{ucgA'}{A} \frac{\partial t}{\partial \beta} = 0. \end{aligned} \tag{3.4}$$

Our objective in this section is to derive analytic expressions for all directional derivatives $\partial Q/\partial\alpha(0,\beta)$ for the flow variables at the singularity ($\beta^* \leqq \beta \leqq 1$). As mentioned in § 1, our basic hypothesis here is that all values $Q(0,\beta)$ are determined by $R_L(p; V_-, V_+)$, where $p=-g_l\beta$. Furthermore, we assume that $Q(\alpha,\beta)$ is smooth up to the singularity $\alpha=0$.

We start by deriving an asymptotic expression for the solution of (3.3).

LEMMA 3.1. *The solution of* (3.3), *along with the boundary conditions* $\xi(\alpha,1)=\alpha$ *and* $\xi(0,\beta)=t(0,\beta)=0$, *is given by*

$$\begin{aligned} \xi(\alpha,\beta) &= \alpha\beta^{1/2}+\varepsilon(\alpha,\beta)\alpha^2, \\ t(\alpha,\beta) &= -k\alpha\beta^{-1/2}+\eta(\alpha,\beta)\alpha^2, \qquad k=(g_lA(0))^{-1}, \end{aligned} \tag{3.5}$$

where $\varepsilon(\alpha,\beta)$, $\eta(\alpha,\beta)$ *are smooth functions (see Appendix for further details on* ε, η).

Proof. Differentiate the first equation of (3.3) with respect to β, the second with respect to α and note that, at $\alpha=0$, $\partial A/\partial\beta=\partial t/\partial\beta=0$. Solving for $\partial t/\partial\alpha$ we obtain

$$2g(0,\beta)\frac{\partial}{\partial\beta}\left(\frac{\partial t}{\partial\alpha}(0,\beta)\right)+\frac{\partial g}{\partial\beta}(0,\beta)\cdot\frac{\partial t}{\partial\alpha}(0,\beta)=0.$$

From (3.2) and (3.3) we have that $\partial t/\partial\alpha(0,1)=-k$ and $\partial g/\partial\beta(0,\beta)=g_l$. Hence, $\partial t/\partial\alpha\,(0,\beta)=-k\beta^{-1/2}$, $\partial\xi/\partial\alpha\,(0,\beta)=\beta^{1/2}$, from which (3.5) follows readily. Q.E.D.

The characteristic relations (3.4) will now be used to determine $\partial u/\partial\alpha\,(0,\beta)$. As we shall see later, this leads easily to the determination of $\partial V/\partial\alpha\,(0,\beta)$ (V as in (2.3)). The following theorem is therefore the fundamental analytical result of this paper.

THEOREM 3.2. *Let* $a(\beta)=\partial u/\partial\alpha\,(0,\beta)$, $\beta^*\leqq\beta\leqq 1$. *Then* $a(\beta)$ *satisfies a differential equation of the form,*

$$a'(\beta)+H(\beta)+T(\beta)\frac{\partial t}{\partial\alpha}(0,\beta)=0, \tag{3.6}$$

where $H(\beta)$, $T(\beta)$ *can be determined explicitly from the equation of state and the initial conditions ahead of the rarefaction (i.e.,* V_l, $(\partial V/\partial\xi)_l$). *Furthermore,* $H(\beta)$, $T(\beta)$ *reflect, respectively, the thermodynamic and geometrical nonuniformity ahead of the rarefaction. In particular,*

$$H(\beta)\equiv 0 \quad \text{if}\left(\frac{\partial S}{\partial\xi}\right)_l=0 \quad (S-\text{entropy}),$$

$$T(\beta)\equiv 0 \quad \text{if } A'(0)=0.$$

While the expression for $H(\beta)$ *is rather complicated and will be given in the process of the proof, we have, with* $\lambda=A'(0)/A(0)$,

$$T(\beta)=-\frac{\lambda}{2}\frac{\partial}{\partial\beta}[u(0,\beta)c(0,\beta)]. \tag{3.7}$$

Equation (3.6) *is supplemented by the initial condition,*

$$a(1)=\left(\frac{\partial u}{\partial\xi}\right)_l+g_l^{-1}\left(\frac{\partial p}{\partial\xi}\right)_l. \tag{3.8}$$

For a γ-law equation of state, (3.6) *yields*

$$
\begin{aligned}
a(\beta) = a(1) &+ \frac{2}{g_l(3\gamma-1)}\left[c_l\left(\frac{\partial g}{\partial \xi}\right)_l - \frac{\gamma+1}{2}\left(\frac{\partial p}{\partial \xi}\right)_l\right]\cdot(\beta^{(3\gamma-1)/2(\gamma+1)}-1) \\
&- \frac{\lambda}{\gamma-3}A(0)^{-1}\rho_l^{-1}\left[(\gamma-1)u_l + 2c_l\right]\cdot(\beta^{(\gamma-3)/2(\gamma+1)}-1) \\
&+ \frac{4\lambda c_l}{3\gamma-5}A(0)^{-1}\rho_l^{-1}\cdot(\beta^{(3\gamma-5)/2(\gamma+1)}-1).
\end{aligned} \tag{3.9}$_\gamma$
$$

(Equation (3.9)$_\gamma$ *must be modified for* $\gamma = \frac{3}{5}, 3$. *See* (3.21)$_\gamma$, (3.22)$_\gamma$.)

Proof. We eliminate p from (3.4) by differentiating the first equation with respect to β, the second with respect to α, subtracting and evaluating at $\alpha = 0$ (where, of course, $\partial t/\partial\beta = 0$). This leads to

$$
\begin{aligned}
&2g(0,\beta)a'(\beta) + \frac{\partial g}{\partial \alpha}(0,\beta)\cdot\frac{\partial u}{\partial \beta}(0,\beta) + a(\beta)\frac{\partial g}{\partial \beta}(0,\beta) \\
&-\lambda\frac{\partial t}{\partial \alpha}(0,\beta)\cdot\frac{\partial}{\partial \beta}(u(0,\beta)c(0,\beta)g(0,\beta)) = 0.
\end{aligned} \tag{3.10}
$$

Since the CRW is nonisentropic, it is impossible to compute g_α merely from p_α. So to eliminate g_α we proceed as follows. Given a state (g_0, p_0) let

$$
g = G(g_0, p_0, p) \tag{3.11}
$$

represent the location of all states (g, p) which have the same entropy as (g_0, p_0). For $\alpha < 0$, let $\xi(\alpha, \beta)$ be as in (3.5) and let

$$
p_0(\alpha,\beta) = p_l + \left(\frac{\partial p}{\partial \xi}\right)_l\cdot\xi(\alpha,\beta), \qquad g_0(\alpha,\beta) = g_l + \left(\frac{\partial g}{\partial \xi}\right)_l\cdot\xi(\alpha,\beta) \tag{3.12}
$$

be the initial values at $\xi(\alpha, \beta)$. Since the flow is isentropic along the streamline $\xi = \text{const.}$, we conclude from (3.11), (3.12), that,

$$
g(\alpha,\beta) = G(g_0(\alpha,\beta),\, p_0(\alpha,\beta),\, p(\alpha,\beta)). \tag{3.13}
$$

Differentiating this equation with respect to α and setting $\alpha = 0$ we have

$$
\begin{aligned}
\frac{\partial g}{\partial \alpha}(0,\beta) &= \left[G_{g_0}(g_l, p_l, p(0,\beta))\left(\frac{\partial g}{\partial \xi}\right)_l + G_{p_0}(g_l, p_l, p(0,\beta))\left(\frac{\partial p}{\partial \xi}\right)_l\right]\cdot\frac{\partial \xi}{\partial \alpha}(0,\beta) \\
&\quad + G_p(g_l, p_l, p(0,\beta))\cdot\frac{\partial p}{\partial \alpha}(0,\beta) \\
&= I(\beta)\cdot\beta^{1/2} + G_p(g_l, p_l, p(0,\beta)) \\
&\quad \cdot\left(g(0,\beta)a(\beta) - \lambda u(0,\beta)c(0,\beta)g(0,\beta)\frac{\partial t}{\partial \alpha}(0,\beta)\right),
\end{aligned} \tag{3.14}
$$

where $I(\beta)$ is the expression in square brackets and we have made use of (3.4), (3.5)

in the evaluation of $\partial p/\partial\alpha$ and $\partial\xi/\partial\alpha$. Inserting (3.14) in (3.10) results in

$$\begin{aligned} &2g(0,\beta)a'(\beta)+E(\beta)a(\beta)\\ &\quad+\frac{\partial u}{\partial\beta}(0,\beta)\cdot\Big[I(\beta)\beta^{1/2}-\lambda G_p(g_l,p_l,p(0,\beta))u(0,\beta)\\ &\qquad\cdot c(0,\beta)g(0,\beta)\frac{\partial t}{\partial\alpha}(0,\beta)\Big]\\ &\quad-\lambda\frac{\partial}{\partial\beta}(u(0,\beta)c(0,\beta)g(0,\beta))\frac{\partial t}{\partial\alpha}(0,\beta)=0, \end{aligned} \tag{3.15}$$

where

$$E(\beta)=\frac{\partial g}{\partial\beta}(0,\beta)+G_p(g_l,p_l,p(0,\beta))g(0,\beta)\frac{\partial u}{\partial\beta}(0,\beta).$$

But $E(\beta)\equiv 0$. Indeed, from the second characteristic relation (3.4), evaluated at $\alpha=0$, we infer that

$$E(\beta)=\frac{\partial g}{\partial\beta}(0,\beta)-\frac{\partial}{\partial\beta}G(g_l,p_l,p(0,\beta))=\frac{\partial}{\partial\beta}(g(0,\beta)-g(0,\beta))=0.$$

We have used here the fact that the limiting values $Q(0,\beta)$ are those of the associated RP, hence isentropic with (g_l,p_l).

Turning now to the terms involving $\lambda\partial t/\partial\alpha\ (0,\beta)$ in (3.15) we use once again the second relation in (3.4) to get

$$\frac{\partial u}{\partial\beta}(0,\beta)\cdot G_p(g_l,p_l,p(0,\beta))u(0,\beta)c(0,\beta)g(0,\beta)=-u(0,\beta)c(0,\beta)\frac{\partial g}{\partial\beta}(0,\beta).$$

Inserting the last equality in (3.15) we get (3.6) with $T(\beta)$ as in (3.7) and (since $g(0,\beta)=g_l\beta$)

$$H(\beta)=\frac{1}{2}g_l^{-1}\beta^{-1/2}I(\beta)\frac{\partial u}{\partial\beta}(0,\beta). \tag{3.16}$$

To show that $H(\beta)$ vanishes identically if $(\partial S/\partial\xi)_l=0$, we observe that under this condition, keeping β fixed,

$$G(g_0(\alpha,\beta),\ p_0(\alpha,\beta),p(0,\beta))-g(0,\beta)=O(\alpha^2),$$

since the entropy at $(g_0(\alpha,\beta),\ p_0(\alpha,\beta))$ deviates only by $O(\alpha^2)$ from that of (g_l,p_l). Differentiation with respect to α leads to $I(\beta)=0$.

Equation (3.8) follows from (3.5) and (2.3) via the chain rule,

$$a(1)=\frac{\partial u}{\partial\alpha}(0,1)=\left(\frac{\partial u}{\partial\xi}\right)_l\frac{\partial\xi}{\partial\alpha}(0,1)+\left(\frac{\partial u}{\partial t}\right)_l\frac{\partial t}{\partial\alpha}(0,1)=\left(\frac{\partial u}{\partial\xi}\right)_l+A(0)\left(\frac{\partial p}{\partial\xi}\right)_l\cdot k.$$

Finally, specializing to the γ-law case, (3.11) is replaced by

$$g=g_0\left(\frac{p}{p_0}\right)^{(\gamma+1)/2\gamma} \tag{3.17}$$

and the well-known solution of the Riemann problem in this case yields

$$\begin{aligned} &p(0,\beta)=p_l\beta^{2\gamma/(\gamma+1)},\quad c(0,\beta)=c_l\beta^{(\gamma-1)/(\gamma+1)},\quad \rho(0,\beta)=\rho_l\beta^{2/(\gamma+1)},\\ &u(0,\beta)=u_l+\frac{2}{\gamma-1}c_l-\frac{2}{\gamma-1}c(0,\beta)=u_l+\frac{2c_l}{\gamma-1}(1-\beta^{(\gamma-1)/(\gamma+1)}). \end{aligned} \tag{3.18}$$

It follows from the definition of $I(\beta)$ in (3.14) that

$$(3.19)_\gamma \qquad I(\beta)=\left[\left(\frac{\partial g}{\partial \xi}\right)_l-\frac{\gamma+1}{2c_l}\left(\frac{\partial p}{\partial \xi}\right)_l\right]\cdot\beta,$$

so that (3.16) becomes,

$$(3.20)_\gamma \qquad H(\beta)=\left[\frac{\gamma+1}{2c_l}\left(\frac{\partial p}{\partial \xi}\right)_l-\left(\frac{\partial g}{\partial \xi}\right)_l\right]\cdot\frac{1}{\rho_l(\gamma+1)}\beta^{1/2-2/(\gamma+1)}.$$

Similarly, we have in this case,

$$T(\beta)\frac{\partial t}{\partial \alpha}(0,\beta)=\frac{\lambda}{2\rho_l A(0)\cdot(\gamma+1)}\left\{\left[(\gamma-1)u_l+2c_l\right]\beta^{-(\gamma+5)/2(\gamma+1)}-4c_l\beta^{(\gamma-7)/2(\gamma+1)}\right\}.$$

Equation $(3.9)_\gamma$ is now obtained by a straightforward integration. Observe that for $\gamma=\frac{5}{3}$, 3, we get

$$(3.21)_\gamma \qquad \begin{aligned}\alpha(\beta)&=a(1)+(4g_l)^{-1}\left[c_l\left(\frac{\partial g}{\partial \xi}\right)_l-2\left(\frac{\partial p}{\partial \xi}\right)_l\right]\cdot(\beta-1)\\&\quad-\frac{\lambda}{4\rho_l A(0)}[(u_l+c_l)\log\beta-4c_l(\beta^{1/2}-1)], \qquad (\gamma=3);\end{aligned}$$

$$(3.22)_\gamma \qquad \begin{aligned}a(\beta)&=a(1)+(2g_l)^{-1}\left[c_l\left(\frac{\partial g}{\partial \xi}\right)_l-\frac{4}{3}\left(\frac{\partial p}{\partial \xi}\right)_l\right]\cdot(\beta^{3/4}-1)\\&\quad+\frac{3\lambda}{4\rho_l A(0)}\left[\left(\frac{2}{3}u_l+2c_l\right)\cdot(\beta^{-1/4}-1)+c_l\log\beta\right], \qquad \left(\gamma=\frac{5}{3}\right).\end{aligned}$$

Q.E.D.

As was stated earlier, all other directional derivatives are now easy to compute. We collect them in the following corollary, where, analogously to (3.11), we denote by,

$$(3.23) \qquad \rho=J(\rho_0,p_0,p),$$

the isentropic curve through (ρ_0, p_0).

COROLLARY 3.3. *Given $a(\beta)=\partial u/\partial\alpha\ (0,\beta)$, the derivatives of g, ρ, p are given by*

$$(3.24) \qquad \frac{\partial p}{\partial \alpha}(0,\beta)=g_l\beta[a(\beta)+\lambda k u(0,\beta)c(0,\beta)\beta^{-1/2}], \qquad k=(g_lA(0))^{-1},$$

$$(3.25) \qquad \begin{aligned}\frac{\partial g}{\partial \alpha}(0,\beta)&=\left[G_{g_0}(g_l,p_l,p(0,\beta))\left(\frac{\partial g}{\partial \xi}\right)_l+G_{p_0}(g_l,p_l,p(0,\beta))\left(\frac{\partial p}{\partial \xi}\right)_l\right]\cdot\beta^{1/2}\\&\quad+G_p(g_l,p_l,p(0,\beta))\cdot\frac{\partial p}{\partial \alpha}(0,\beta),\end{aligned}$$

$$(3.26) \qquad \begin{aligned}\frac{\partial \rho}{\partial \alpha}(0,\beta)&=\left[J_{\rho_0}(\rho_l,p_l,p(0,\beta))\left(\frac{\partial \rho}{\partial \xi}\right)_l+J_{p_0}(\rho_l,p_l,p(0,\beta))\left(\frac{\partial p}{\partial \xi}\right)_l\right]\cdot\beta^{1/2}\\&\quad+c(0,\beta)^{-2}\frac{\partial p}{\partial \alpha}(0,\beta).\end{aligned}$$

Specializing to the γ-law case, we have instead of (3.25), (3.26),

$$(3.27)_\gamma \qquad \frac{\partial g}{\partial \alpha}(0,\beta)=\left[\left(\frac{\partial g}{\partial \xi}\right)_l-\frac{\gamma+1}{2c_l}\left(\frac{\partial p}{\partial \xi}\right)_l\right]\cdot\beta^{3/2}+\frac{\gamma+1}{2\gamma}\frac{g_l}{p_l}\frac{\partial p}{\partial \alpha}(0,\beta)\cdot\beta^{-(\gamma-1)/(\gamma+1)},$$

$$(3.28)_\gamma \qquad \frac{\partial \rho}{\partial \alpha}(0,\beta)=\left[\left(\frac{\partial \rho}{\partial \xi}\right)_l-c_l^{-2}\left(\frac{\partial p}{\partial \xi}\right)_l\right]\cdot\beta^{(\gamma+5)/2(\gamma+1)}+c(0,\beta)^{-2}\frac{\partial p}{\partial \alpha}(0,\beta).$$

Proof. Equation (3.24) follows from (3.4) and $g(0,\beta)=g_l\beta$. Equation (3.25) is just (3.14) and (3.26) is its analogue, where $J_p=c^{-2}$.

To obtain $(3.27)_\gamma$ we use in (3.14) the expression $(3.19)_\gamma$ for $I(\beta)$ and $(3.17)_\gamma$ for G. To prove $(3.28)_\gamma$ we see that in this case

$$J(\rho_0,p_0,p)=\rho_0\left(\frac{p}{p_0}\right)^{1/\gamma},$$

and $c^2=\gamma p/\rho$. Q.E.D.

Remark 3.4. Note that given $\beta_0\in[\beta^*,1]$, all the derivatives $\partial Q/\partial\alpha$ $(0,\beta_0)$ depend only on data ahead of the characteristic $\beta=\beta_0$ (i.e., initial data for $\xi<0$ and values for $\beta_0\leqq\beta\leqq 1$). This, of course, is in accordance with the fact that this is really the domain of dependence for this line, or, otherwise stated, that no sonic signals (i.e., characteristics) from earlier time impinge on the curve from the other side.

4. Time derivatives of p, u on the interface. We continue the discussion of the previous section, assuming the configuration of Fig. 1. We seek here expressions for $(\partial p/\partial t)^*$, $(\partial u/\partial t)^*$, the time derivatives along the contact discontinuity $\xi=0$. As we shall see, these derivatives satisfy a pair of linear (algebraic) equations. Basically we follow in this section the idea of van Leer [12].

In the present setup we have a shock travelling to the right. Using the conventions of § 2 we let V_r (resp. V_r^*) be the pre-shock (resp. post-shock) values of V at the singularity (obviously, the subscript is suppressed in u^*, p^*). The (Lagrangian) shock speed W_r (i.e., initial slope of shock trajectory in ξ,t coordinates) is given by the well-known jump condition,

$$W_r=A(0)\frac{p^*-p_r}{u^*-u_r}. \tag{4.1}$$

We may now state the following theorem (recall that $\lambda=A'(0)/A(0)$).

THEOREM 4.1. *The derivatives $(\partial p/\partial t)^*$, $(\partial u/\partial t)^*$ are determined by a pair of linear equations,*

$$a_l\left(\frac{\partial u}{\partial t}\right)^*+b_l\left(\frac{\partial p}{\partial t}\right)^*=d_l, \tag{4.2}$$

$$a_r\left(\frac{\partial u}{\partial t}\right)^*+b_r\left(\frac{\partial p}{\partial t}\right)^*=d_r, \tag{4.3}$$

where, with $\beta^*=g_l^*/g_l$ and $a(\beta)$ as in Theorem 3.2,

$$a_l=1,\quad b_l=(g_l^*)^{-1},\quad d_l=-A(0)(g_lg_l^*)^{1/2}a(\beta^*)-\lambda u^*c_l^*. \tag{4.4}$$

As for a_r, b_r, d_r, they can be determined explicitly from the values of V_r^*, V_r, $(\partial V/\partial\xi)_r$ and the Hugoniot (u,p) relation.

Specializing to the case of a γ-law we have, with $\mu^2=(\gamma-1)/\gamma+1)$,

$$\begin{aligned}
a_r&=-2+\frac{1}{2}\frac{p^*-p_r}{p^*+\mu^2p_r},\\
b_r&=-\frac{u^*-u_r}{2(p^*+\mu^2p_r)}+A(0)W_r^{-1}+A(0)^{-1}W_r(g_r^*)^{-2},\\
d_r&=L_u\cdot\left(\frac{\partial u}{\partial\xi}\right)_r+L_p\left(\frac{\partial p}{\partial\xi}\right)_r+L_\rho\left(\frac{\partial\rho}{\partial\xi}\right)_r+\lambda\cdot L_\lambda,
\end{aligned} \tag{4.5}_\gamma$$

where

$$L_u = \frac{A(0)}{2}(u_r - u^*)\left(\rho_r + \mu^2 \frac{g_r^2}{p^* + \mu^2 p_r}\right) - A(0)^2 W_r^{-1} g_r^2 - W_r,$$

$$L_p = 2A(0) + \frac{\mu^2}{2} W_r \frac{u^* - u_r}{p^* + \mu^2 p_r},$$

$$L_\rho = \frac{A(0)}{2} \frac{p^* - p_r}{\rho_r},$$

$$L_\lambda = -W_r A(0)^{-1} u^* (\rho_r^*)^{-1} - u_r \rho_r^{-1}(u^* - u_r)\left[g_r^2 \left(\frac{1}{p^* - p_r} + \frac{\mu^2}{2} \frac{1}{p^* + \mu^2 p_r} \right) + \frac{\rho_r}{2} \right].$$

Proof. To establish (4.2) we note that the flow in the region $\xi(\alpha, \beta^*) \leqq \xi \leqq 0$ (i.e., between the tail characteristic of the rarefaction fan and the contact discontinuity) is smooth. Using the chain rule at $\alpha = 0$, $\beta = \beta^*$ we get

$$\frac{\partial p}{\partial \alpha}(0, \beta^*) = \left(\frac{\partial p}{\partial t}\right)^* \frac{\partial t}{\partial \alpha}(0, \beta^*) + \left(\frac{\partial p}{\partial \xi}\right)_l^* \frac{\partial \xi}{\partial \alpha}(0, \beta^*). \tag{4.6}$$

From (2.3) we have $(\partial p/\partial \xi)_l^* = -A(0)^{-1}(\partial u/\partial t)^*$. Also, Corollary 3.3 with $\beta = \beta^*$ yields

$$\frac{\partial p}{\partial \alpha}(0, \beta^*) = g_l^* a(\beta^*) + \lambda u^* c_l^* A(0)^{-1} (\beta^*)^{1/2}.$$

Using this expression and (3.5) in (4.6) gives (4.4).

Turning now to (4.3), recall that the Hugoniot (u, p) relation can be written in the general form,

$$K(u, p, u_+, p_+, \rho_+) = 0, \tag{4.7}$$

where (u, p) are the post-shock values, (u_+, p_+, ρ_+) are the pre-shock values. The shock trajectory in (ξ, t) plane has the slope,

$$W_+ = A(r(\xi, t)) \frac{p - p_+}{u - u_+},$$

so that by differentiation,

$$\left(\frac{\partial}{\partial t} + W_+ \frac{\partial}{\partial \xi}\right) K(u, p, u_+, p_+, \rho_+) = 0. \tag{4.8}$$

As $t \to 0$, we have $\partial/\partial t\, Q_+ \to (\partial Q/\partial t)_r$ for $Q = u, p, \rho$ (see Table 1, § 2), while $(\partial u/\partial t, \partial p/\partial t) \to ((\partial u/\partial t)^*, (\partial p/\partial t)^*)$. Similarly, $\partial/\partial \xi\, Q_+ \to (\partial Q/\partial \xi)_r$ while $(\partial u/\partial \xi, \partial p/\partial \xi) \to ((\partial u/\partial \xi)_r^*, (\partial p/\partial \xi)_r^*)$. Now, using (2.3) we may write

$$\left(\frac{\partial u}{\partial t}\right)_r = -A(0)\left(\frac{\partial p}{\partial \xi}\right)_r, \quad \left(\frac{\partial p}{\partial t}\right)_r = -g_r^2 \left(\frac{\partial (Au)}{\partial \xi}\right)_r, \quad \left(\frac{\partial \rho}{\partial t}\right)_r = -\rho_r^2 \left(\frac{\partial (Au)}{\partial \xi}\right)_r,$$

while

$$\left(\frac{\partial p}{\partial \xi}\right)_r^* = -A(0)^{-1}\left(\frac{\partial u}{\partial t}\right)^*,$$

$$\left(\frac{\partial u}{\partial \xi}\right)_r^* = -A(0)^{-1}(g_r^*)^{-2}\left(\frac{\partial p}{\partial t}\right)^* - A(0)^{-1} u^* \left(\frac{\partial A}{\partial \xi}\right)_r^*. \tag{4.9}$$

Observe that it follows from the definition (2.2) of the coordinate ξ that,

$$\left(\frac{\partial A}{\partial \xi}\right)_r = \rho_r^{-1}\lambda, \quad \left(\frac{\partial A}{\partial \xi}\right)_r^* = (\rho_r^*)^{-1}\lambda, \quad \lambda = \frac{A'(0)}{A(0)}. \tag{4.10}$$

Incorporating these considerations in (4.8) we get a relation of the type (4.3), with the dependence of a_r, b_r, d_r as stated.

Specializing to the γ-law case we have [5, § 81],

$$K(u, p, u_+, p_+, \rho_+) = u - u_+ - (p - p_+)\left[\frac{1-\mu^2}{\rho_+(p+\mu^2 p_+)}\right]^{1/2}, \qquad \mu^2 = \frac{\gamma-1}{\gamma+1}, \tag{4.11}$$

and carrying out the computations in (4.8) we obtain $(4.5)_\gamma$.

Remark on regions of smooth flow. In such regions $|V^* - V_-|$, $|V^* - V_+|$ and $|\beta^* - 1|$ are all $O(\Delta x)$. Thus it follows easily from (4.4),(4.5) that within $O(\Delta x)$,

$$a_l = 1, \quad b_l = g_l^{-1}, \quad d_l = -A(0)g_l a(1) - \lambda u_l c_l, \tag{4.12}$$

$$a_r = -1, \quad b_r = g_r^{-1}, \quad d_r = -A(0)\left(g_r\left(\frac{\partial u}{\partial \xi}\right)_r - \left(\frac{\partial p}{\partial \xi}\right)_r\right) - \lambda u_r c_r, \tag{4.13}$$

where in (4.13) we have used the approximation $W_r = A(0)g_r$. Observe that in view of (3.8) the case of "weak rarefaction" ((4.12)) is in complete agreement with the case of a "weak shock" ((4.13)) (changes in signs are due to the reflection $\xi \to -\xi$).

Using the coefficients (4.12), (4.13) in (4.2), (4.3) the values of $(\partial u/\partial t)^*$, $(\partial p/\partial t)^*$ are determined with an error which is again $0(\Delta x)$ and in this case we deduce from (1.5) that the flux values $F(U)_{i+1/2}^{n+1/2}$, $G(U)_{i+1/2}^{n+1/2}$, are evaluated with an error which is $0(\Delta x^2)$ whereas the differences appearing in the right-hand side of (1.2) are $0(\Delta x^3)$. Hence, the scheme (1.2) retains its second-order accuracy in this case.

5. The Eulerian GRP, nonsonic case. In this section we take up our main goal in this paper, namely, the solution of the GRP for the Eulerian system (1.1), in the sense of (1.4). Our setup is again as shown in Fig. 1, with the jump located at $r = 0$, and flow variables linearly distributed on both sides. As was discussed in the Introduction, we use a "local" Lagrangian coordinate, as defined by (2.2). The passage from r-derivatives to ξ-derivatives is given by,

$$\left(\frac{\partial Q}{\partial \xi}\right)_l = A(0)^{-1}\rho_l^{-1}\left(\frac{\partial Q}{\partial r}\right)_l, \qquad \left(\frac{\partial Q}{\partial \xi}\right)_r = A(0)^{-1}\rho_r^{-1}\left(\frac{\partial Q}{\partial r}\right)_r. \tag{5.1}$$

(See the paragraph following (2.3) considering the consistency of the assumptions of simultaneous linearity in terms of ξ, r.)

The results of § 4 now serve to determine $(\partial p/\partial t)^*$, $(\partial u/\partial t)^*$, the time-derivatives in the direction of the contact discontinuity. We are looking for derivatives $(\partial Q/\partial t)_0$, $Q = u, p, \rho$, in the direction of the grid-line $r = 0$. Let $\xi = \xi(t)$ be the representation of this grid-line in (ξ, t). It is easily seen that

$$\frac{d\xi}{dt} = -A(r(\xi, t))\rho(\xi, t)u(\xi, t), \qquad \xi(0) = 0. \tag{5.2}$$

Indeed, this follows by differentiating the relation $r(\xi(t), t) = 0$ and noting that $\partial \xi/\partial r = A\rho$.

The values in the right-hand side of (5.2) at $\xi = t = 0$ are given by $U_0 = R_E(0; U_l, U_r)$, the solution of the associated RP (see (1.6)).

In this section we consider the *nonsonic case*, namely, *the line* $\xi = \xi(t)$ *is not contained in a rarefaction fan.* From (5.2) and the chain rule we get

$$\left(\frac{\partial Q}{\partial t}\right)_0 = \frac{\partial}{\partial t} Q(\xi, t)_{t=0} - A(0)\rho_0 u_0 \frac{\partial}{\partial \xi} Q(\xi, t)_{t=0}, \qquad Q = p, \rho, u. \tag{5.3}$$

In dealing with (5.3) one must distinguish among several cases, corresponding to the various possible locations of $r = 0$ relative to the wave system. We label them as $k = 1$ to 4. Let $W_e = A(0)^{-1}\rho_r^{-1} W_r + u_r$ be the "Eulerian" shock velocity. Then:

$k = 1$ if $u_l - c_l > 0$. In this case $U_0 = U_l$.

$k = 2$ if $u^* - c_l^* < 0 < u^*$. In this case $U_0 = U_l^*$.

$k = 3$ if $u^* < 0 < W_e$. In this case $U_0 = U_r^*$.

$k = 4$ if $W_e < 0$. In this case $U_0 = U_r$.

The t-derivatives in the right-hand side of (5.3) are obtained directly from the results of the previous section (note that, e.g., $(\partial\rho/\partial t)_r^* = (c_r^*)^{-2}(\partial p/\partial t)^*$ is the derivative needed for the case $k = 3$, etc.), or, in the cases $k = 1$ and $k = 4$, from (1.1).

The ξ-derivatives in the right-hand side of (5.3) are, of course, equal to the initial slopes for $k = 1, 4$. For the case $k = 2$ (resp. $k = 3$) they correspond to $(\partial Q/\partial\xi)_l^*$ (resp. $(\partial Q/\partial\xi)_r^*$). See Table 1, § 2. It follows from (2.3) that the derivatives of p, u, can be expressed in terms of $(\partial p/\partial t)^*$, $(\partial u/\partial t)^*$, (see (4.9)). The following lemma handles the density.

LEMMA 5.1. *Assume the setup of Fig.* 1. *Then*

$$\left(\frac{\partial\rho}{\partial\xi}\right)_l^* = (\beta^*)^{-1/2}\frac{\partial\rho}{\partial\alpha}(0, \beta^*) + A(0)^{-1}(g_l^*)^{-1}(c_l^*)^{-2}\left(\frac{\partial p}{\partial t}\right)^*, \tag{5.4}$$

$$\begin{aligned}\left(\frac{\partial\rho}{\partial\xi}\right)_r^* &= \left(\frac{\partial p}{\partial t}\right)^* \cdot [-A(0)^2(\rho_r^*)^2 W_r^{-3} - 3(c_r^*)^{-2} W_r^{-1}] \\ &+ \left(\frac{\partial u}{\partial t}\right)^* \cdot [3(\rho_r^*)^2 A(0) W_r^{-2}] \\ &+ \left(\frac{\partial p}{\partial\xi}\right)_r \cdot [3(\rho_r^*)^2 A(0)^2 W_r^{-2}] + \left(\frac{\partial\rho}{\partial\xi}\right)_r \cdot \left(\frac{\rho_r^*}{\rho_r}\right)^2 \\ &+ \left(\frac{\partial(Au)}{\partial\xi}\right)_r \cdot [-g_r^2(\rho_r^*)^2 A(0)^2 W_r^{-3} - W_r^{-1}(\rho_r^*)^2] \\ &+ \left(\frac{\partial u}{\partial\xi}\right)_r \cdot [-2A(0) W_r^{-1}(\rho_r^*)^2] - 2(\rho_r^*)^2 u^* \left(\frac{\partial A}{\partial\xi}\right)_r^* W_r^{-1}.\end{aligned} \tag{5.5}$$

Remark. Observe that $(\partial A/\partial\xi)_r$, $(\partial A/\partial\xi)_r^*$ are given by (4.10).

Proof. To establish (5.4) we proceed as in (4.6) to obtain,

$$\frac{\partial\rho}{\partial\alpha}(0, \beta^*) = (c_l^*)^{-2}\left(\frac{\partial p}{\partial t}\right)^* \frac{\partial t}{\partial\alpha}(0, \beta^*) + \left(\frac{\partial\rho}{\partial\xi}\right)_l^* \frac{\partial\xi}{\partial\alpha}(0, \beta^*).$$

Using (3.5) and solving for $(\partial\rho/\partial\xi)_l^*$ we get (5.4). To prove (5.5) we use a procedure similar to (4.8). Thus, using a well-known shock relation [5, (59.05)] we may write,

$$\left(\frac{\partial}{\partial t} + W_+\frac{\partial}{\partial\xi}\right)[(p - p_+)(\tau_+ - \tau) - (u - u_+)^2] = 0.$$

As in the case of (4.8), pre-shock time derivatives are translated into spatial derivatives whereas the post-shock derivatives $(\partial p/\partial \xi)_r^*$, $(\partial u/\partial \xi)_r^*$ are evaluated by means of (4.9). Hence, the last equation can be solved for $(\partial \rho/\partial \xi)_r^*$, resulting in (5.5). In the process of obtaining (5.5), use is made of the two expressions for the Lagrangian shock speed,

$$W_r = A(0)\frac{P^* - p_r}{u^* - u_r} = A(0)\frac{u^* - u_r}{\tau_r - \tau_r^*}. \qquad \text{Q.E.D.}$$

To sum up the present case, we see that the results of the previous section and Lemma 5.1 enable us to implement (5.3) for all flow variables, thus concluding the treatment of the GRP for the nonsonic case.

6. The Eulerian GRP, sonic case. In this section we assume once again the setup of Fig. 1 and the same "local" Lagrangian representation. However, we assume now that the grid line $r=0$ is contained in the Γ^--rarefaction fan, for which we use the (characteristic) notation and analysis of § 3. Of course, the representation (5.2) of $r=0$ is of no use here since (5.3) is meaningless in the situation at hand. We are forced, therefore, to replace the (ξ, t) representation by a more refined characteristic parametrization.

Let $(\alpha(t), \beta(t))$ be the trajectory $r=0$ in the (α, β) plane. We have $(\alpha(0), \beta(0)) = (0, \beta_0)$, where β_0 is determined by,

$$u(0, \beta_0) = c(0, \beta_0). \tag{6.1}$$

For a γ-law gas we obtain from $(3.18)_\gamma$,

$$\beta_0 = \left[\frac{\gamma-1}{\gamma+1}\left(\frac{u_l}{c_l} + \frac{2}{\gamma-1}\right)\right]^{(\gamma+1)/(\gamma-1)}. \tag{6.2}_\gamma$$

In the domain covered by the rarefaction fan all flow variables are represented as functions of α, β (see § 3) and (5.3) is replaced by,

$$\left(\frac{\partial Q}{\partial t}\right)_0 = \frac{\partial Q}{\partial \alpha}(0, \beta_0)\cdot \alpha'(0) + \frac{\partial Q}{\partial \beta}(0, \beta_0)\cdot \beta'(0). \tag{6.3}$$

In view of our basic hypothesis (1.6) all derivatives $\partial Q/\partial \beta\ (0, \beta_0)$ are evaluated from the solution of the associated RP. Also, the values of $\partial Q/\partial \alpha\ (0, \beta_0)$ are all given in Theorem 3.2 and Corollary 3.3. We are left with the problem of evaluating $\alpha'(0)$, $\beta'(0)$.

LEMMA 6.1. *We have*

$$\alpha'(0) = -A(0) g_l \beta_0^{1/2}, \tag{6.4}$$

$$\beta'(0) = \frac{1}{2}\beta_0^{1/2} A(0)\left[\frac{\partial(\rho c)}{\partial \alpha}(0, \beta_0) - \frac{\partial(\rho u)}{\partial \alpha}(0, \beta_0)\right]. \tag{6.5}$$

Proof. Equation (6.4) is easy. Simply regard t as a function along $(\alpha(t), \beta(t))$ and use (6.3) to get,

$$1 = \frac{\partial t}{\partial \alpha}(0, \beta_0)\cdot \alpha'(0) + \frac{\partial t}{\partial \beta}(0, \beta_0)\cdot \beta'(0) = -g_l^{-1} A(0)^{-1} \beta_0^{-1/2} \alpha'(0).$$

On the other hand, (6.5) is much harder to establish and requires further details about $\varepsilon(\alpha, \beta)$, $\eta(\alpha, \beta)$ in the expansion (3.5). We prove it in the Appendix.

The description of the Eulerian scheme is now complete.

7. Numerical experiments. As was mentioned in the Introduction, our scheme is a straightforward implementation of the formulas (1.2), (1.5). Thus, we need to solve one Riemann problem per point per time-step. In addition to that, the main bulk of computations is involved in the evaluation of $[(\partial/\partial t)U]^n_{i+1/2}$ (see (1.5)). For simplicity let us assume a γ-law equation-of-state and that all points are nonsonic. Then the time derivatives are determined by equations (4.2), (4.3) for the Lagrangian case and the additional modification (5.3) for the Eulerian case. The evaluation of the coefficients in (4.2)-(4.3) is given in (4.4)-(4.5)$_\gamma$. It is seen that the evaluation of $a(\beta^*)$ forces one exponentiation in the planar case, three exponentiations in the nonplanar case. Thus, roughly, the computation of the time-derivatives requires one to three exponentiations and the order of magnitude of 20 multiplications per point per time-step. Recall that the solution of a Riemann problem involves iterations. Hence, the additional amount of work needed for the computation of time derivatives does not usually exceed 50% of that required by the Riemann solver. To summarize, without any attempt at efficiency, the total amount of work involved is much less than two Riemann solvers per point per time-step. As mentioned earlier, the monotonicity algorithm is of the simplest type and requires virtually no additional computational effort.

Another contributing factor to the overall efficiency of the present scheme lies in the fact that it turns out to be very stable numerically. We can use a CFL number which is as high as 0.9 without affecting the results or, alternatively, use a constant fixed Δt which is not required to be very small even at times of interactions (such as reflection of a shock wave from a center of symmetry).

We present here the results of two test problems.

(a) *A converging cylindrical shock.* Initially a cylindrical diaphragm of radius $r = 100$ separates two uniform regions of gas ($\gamma = 1.4$) at rest. We take $p = \rho = 4$ in the outer region, $p = \rho = 1$ in the inner region. Removing the diaphragm at $t = 0$ leads to a shock wave followed by a contact discontinuity, both moving inward, and a diverging rarefaction wave. As is well known, the shock accelerates as it approaches the axis of symmetry, is reflected from the axis, interacts with the (still converging) contact discontinuity, which results in a transmitted shock, a contact discontinuity (converging), and a weak reflected (converging) shock. This problem was treated by Lapidus [9], Abarbanel-Goldberg [1] and Sod [11]. Before presenting our results we must refer to the boundary conditions at the singularity $r = 0$.

The analytic expressions derived above become singular as $A(0) = 0$. Thus, naturally, one obtains a solution U_ε, using an approximating cross-section $A_\varepsilon(r)$, and then lets $\varepsilon \to 0$ in order to get the limit solution U. One possibility that comes to mind is,

$$A_\varepsilon(r) = \begin{cases} A(r), & r \geqq \varepsilon, \\ \varepsilon, & r \leqq \varepsilon. \end{cases}$$

In this case, however, $A'_\varepsilon(r)$ is discontinuous, or, if "smoothed" at $r = \varepsilon$, requires $A''_\varepsilon(r)$ to grow indefinitely, contrary to our basic hypothesis that all functions are "approximately linear". Hence, the following seems to be a better choice,

$$A_\varepsilon(r) = A(r) + \varepsilon. \tag{7.1}$$

At $r = 0$ we impose the condition $u = 0$. This amounts to a "reflection procedure" at $r = 0$, or, equivalently, to the fact that equations (4.2)-(4.3) reduce to the single equation,

$$b_r \left(\frac{\partial p}{\partial t}\right)^* = d_r. \tag{7.2}$$

Note that the coefficients in (7.2) depend on ε. After determining $(\partial p/\partial t)^*$ we may

proceed as before to obtain the solution $U_\varepsilon(r)$. It is remarkable that the passage to the limit $\varepsilon \to 0$ can be effected by the application of the following simple rule.

CLAIM 7.1. *Let* $p(0, t) = \lim_{\varepsilon \to 0} p_\varepsilon(0, t)$ *be the pressure at the singularity* $r = 0$, *where* $A(0) = 0$. *Then* $(\partial p/\partial t)^* = p'(0, t)|_{t=0}$ *can be computed from* (7.2) *by setting,*

$$b_r = g_r^{-1}, \qquad d_r = -\left(c_r \left(\frac{\partial u}{\partial r}\right)_r - \rho_r^{-1} \left(\frac{\partial p}{\partial r}\right)_r \right). \tag{7.3}$$

Then, $(\partial \rho/\partial t)^* = c_r^{-2}(\partial p/\partial t)^*$.

Observe that $(\partial u/\partial r)_r$, $(\partial p/\partial r)_r$ are the slopes (with respect to r) of the initial data at $r = 0$ (and $t = 0$).

Proof. Consider (7.2) with $A_\varepsilon = A_\varepsilon(r)$. At $r = 0$ we apply the condition $u_r = u^* = 0$ so that the solution of the RP there gives just an "acoustic wave", i.e., the characteristic line with slope $c(0)$. Thus, we are in the situation discussed in the remark at the end of § 4, and the coefficients in (7.2) are given by (4.13). Using that $d\xi = A\rho\, dr$ (see (2.2)) we replace $A(0)(\partial Q/\partial \xi)_r$ by $\rho_r^{-1}(\partial Q/\partial r)_r$, $Q = u, p$. This leads to (7.3).

Turning back to the cylindrical shock problem described above, we used our scheme with 200 points (i.e., $\Delta r = 1$), applying (7.3) at $r = 0$. There was no additional dissipative mechanism, except for a simple monotonicity algorithm suggested by van Leer [12, Fig. 3] and which says, roughly, that slopes are set equal to zero at extremal points (of average values) and are so adjusted that cell-boundary values never go beyond the average values of neighboring cells.

The results are shown in Figs. 2–7. In each figure profiles of the velocity, pressure, density and entropy ($= p/\rho^\gamma$) are given at the specified time-level. Let us discuss briefly

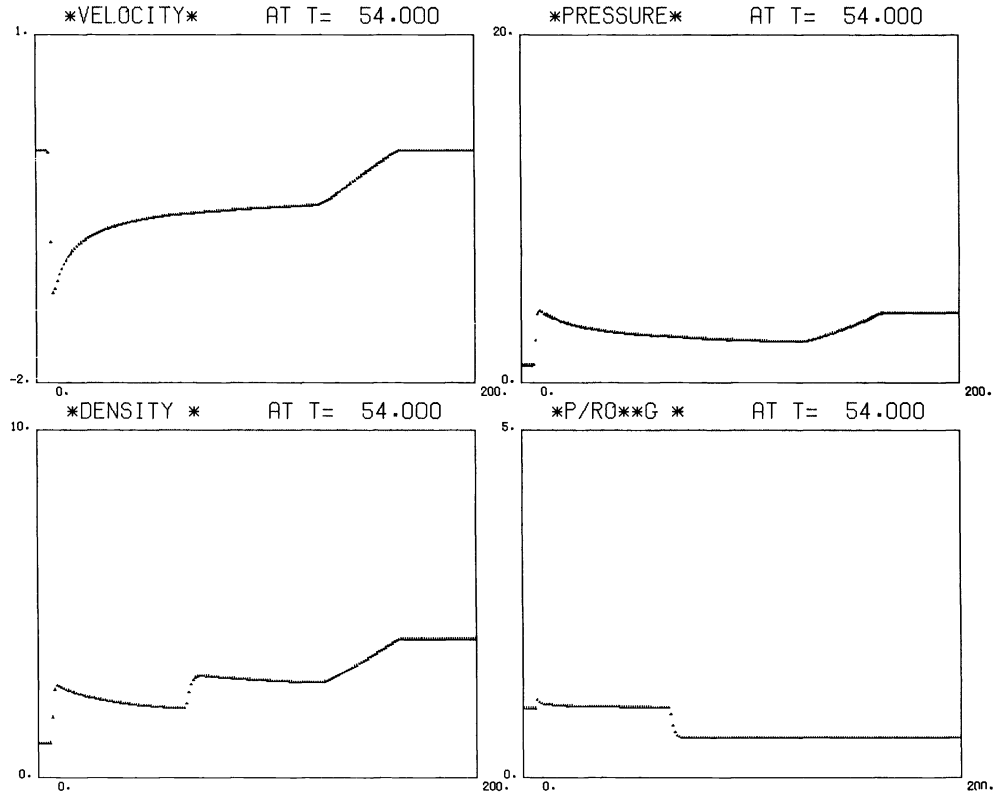


FIG. 2. *A cylindrical converging shock at* $t = 54$.

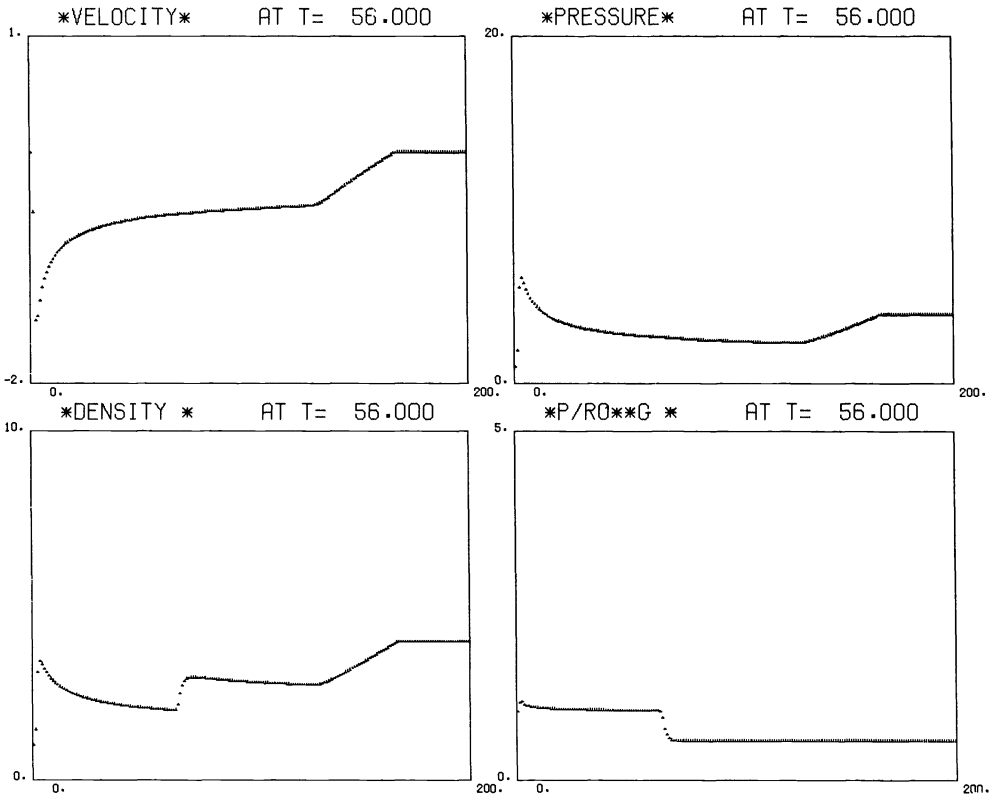


FIG. 3. *A cylindrical converging shock at $t = 56$ is about to hit the axis of symmetry.*

each figure:

Fig. 2. At $t = 54$ the converging shock approaches $r = 0$.

Fig. 3. At $t = 56$ the shock is about to hit the axis.

Fig. 4. At $t = 58$ the shock has been reflected from the axis.

Fig. 5. At $t = 70$ the diverging shock (followed by a rarefaction wave) is headed towards the converging contact discontinuity.

Fig. 6. At $t = 90$, interaction of shock and contact discontinuity.

Fig. 7. At $t = 110$, as a result of the interaction we see, in addition to the original shock and contact discontinuity, a weak converging shock. In previous calculations this weak shock was detected only by Sod [11], using the sharp resolution of the random choice method.

Overall our results here are in very good agreement with those of Abarbanel-Goldberg [1] and Sod [11]. This includes the arrival time of the (main) shock at $r = 0$, as well as the shape and extremal values of the flow profiles.

(b) *Quasi* 1-D *nozzle problem.* We have calculated the steady-state solution to the duct flow problem in Shubin, Stephens and Glaz [10]. The cross-section is given by

$$A(r) = 1.398 + 0.347 \tanh(0.8\, r - 4), \qquad 0 \leqq r \leqq 10,$$

and the boundary conditions are:

$$p(0, t) = 0.3809, \qquad p(0, t) = 0.502,$$

$$u(0, t) = 1.299, \qquad \rho(10, t) = 0.776.$$

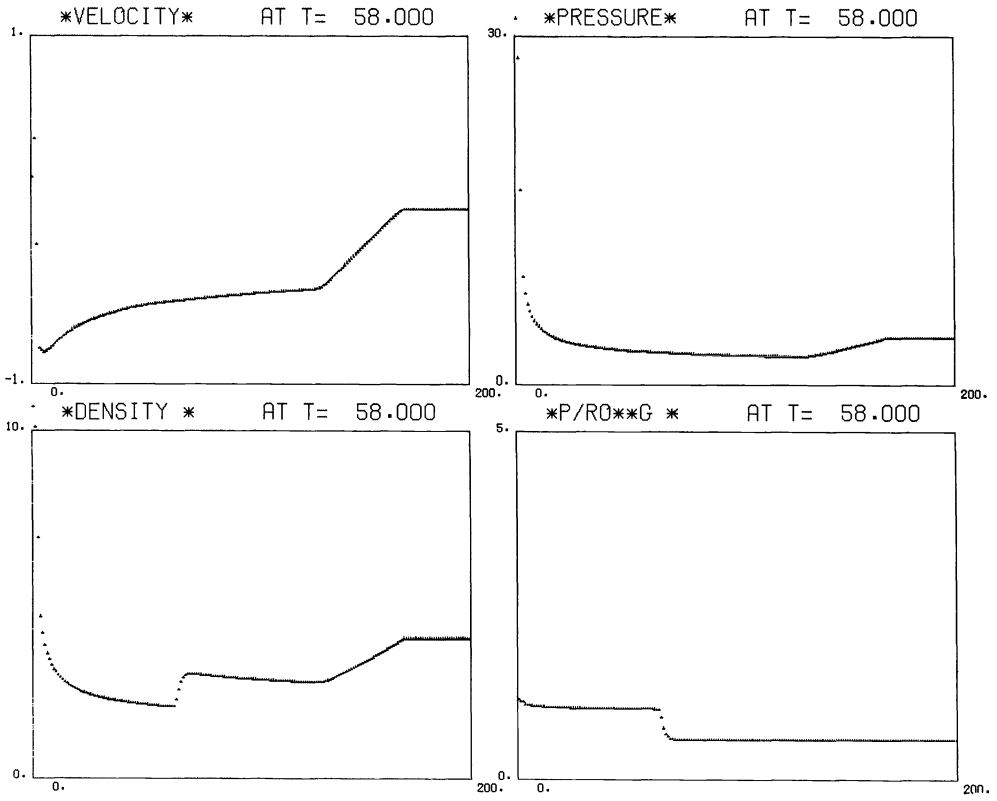


FIG. 4. *At $t = 58$ the shock has been reflected from the axis of symmetry.*

The results for the steady-state solution are given in Figs. 8 and 9, where 16 and 32 equally spaced mesh-points are used, respectively. The exact solution is plotted as a solid line. This problem has been worked out by Colella [3] and Harten [8] using their high-resolution methods. Observe that the finer mesh produces an improved numerical result. In particular, there is no significant deviation at the shock (compare with [3], where an undershoot is present in the finer mesh).

We emphasize once again that the scheme used here is precisely that used in the previous test problem and we did not have to make any special adjustments.

Appendix. The sonic case, proof of (6.5). The expansion of ξ, t in terms of the characteristic coordinates α, β inside a rarefaction fan (as in Fig. 1) is given by (see (3.5))

$$\begin{aligned} &\xi(\alpha,\beta) = \alpha\beta^{1/2} + \varepsilon(\alpha,\beta)\alpha^2, \\ &t(\alpha,\beta) = -k\alpha\beta^{-1/2} + \eta(\alpha,\beta)\alpha^2, \qquad k = (g_l A(0))^{-1}, \end{aligned} \tag{A.1}$$

where $(\alpha,\beta) \in [-\alpha_0, 0] \times [\beta^*, 1]$. Our first objective here is to study $\varepsilon(\alpha,\beta)$, $\eta(\alpha,\beta)$ in some more detail. Recall that the functions $\xi(\alpha,\beta)$, $t(\alpha,\beta)$ satisfy the characteristic equations,

$$\frac{\partial \xi}{\partial \alpha} = -gA\frac{\partial t}{\partial \alpha}, \tag{A.2}$$

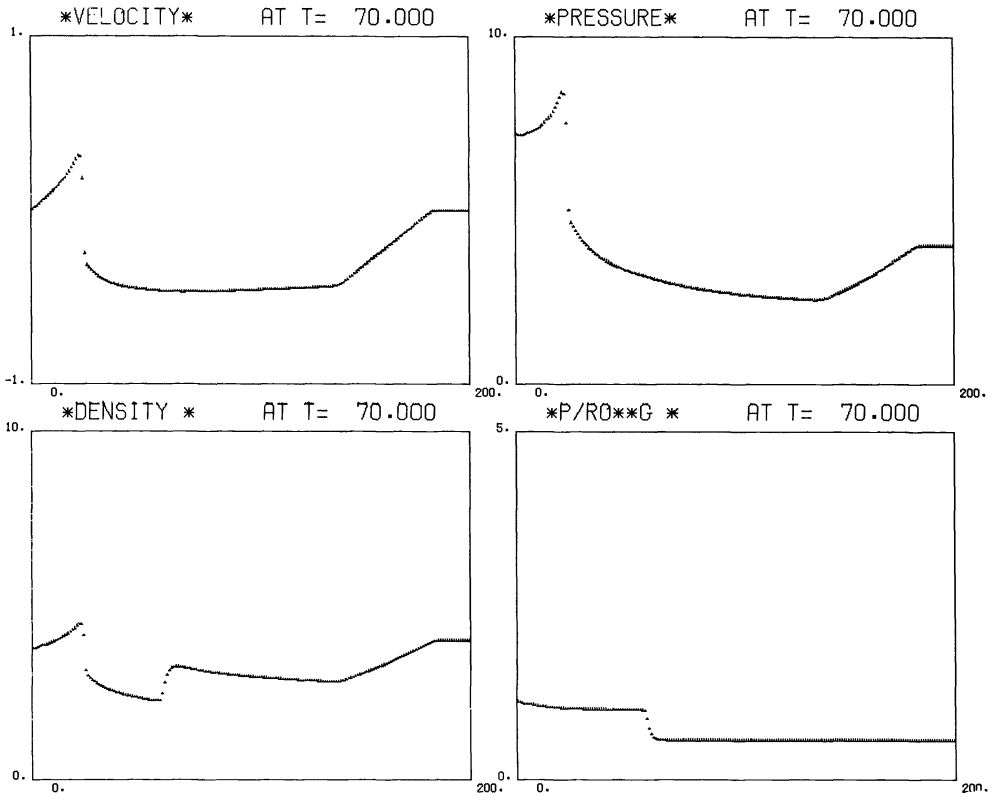


FIG. 5. *At $t=70$ the shock is diverging while contact discontinuity is converging.*

$$\frac{\partial \xi}{\partial \beta} = gA\frac{\partial t}{\partial \beta}. \tag{A.3}$$

Inserting (A.1) in (A.2) and dropping all terms which are $O(\alpha^2)$ we have,

$$\begin{aligned}\beta^{1/2}+2\alpha\varepsilon(0,\beta) &= -g(0,\beta)A(0)[-k\beta^{-1/2}+2\alpha\eta(0,\beta)] \\ &\quad +\frac{\partial(gA)}{\partial\alpha}(0,\beta)\cdot k\beta^{-1/2}\alpha.\end{aligned} \tag{A.4}$$

Using $g(0,\beta)=g_l\beta$ we get,

$$\varepsilon(0,\beta)=\frac{k}{2}\frac{\partial(gA)}{\partial\alpha}(0,\beta)\cdot\beta^{-1/2}-k^{-1}\beta\eta(0,\beta). \tag{A.5}$$

Similarly, inserting (A.1) in (A.3) and dropping all terms which are $O(\alpha^3)$ we get

$$\begin{aligned}\tfrac{1}{2}\alpha\beta^{-1/2}+\varepsilon_\beta(0,\beta)\alpha^2 &= g(0,\beta)A(0)[\tfrac{1}{2}k\alpha\beta^{-3/2}+\eta_\beta(0,\beta)\alpha^2] \\ &\quad +\frac{k}{2}\cdot\frac{\partial(gA)}{\partial\alpha}(0,\beta)\alpha^2\beta^{-3/2},\end{aligned}$$

so that

$$\varepsilon_\beta(0,\beta)=k^{-1}\beta\eta_\beta(0,\beta)+\frac{k}{2}\frac{\partial(gA)}{\partial\alpha}(0,\beta)\cdot\beta^{-3/2}. \tag{A.6}$$

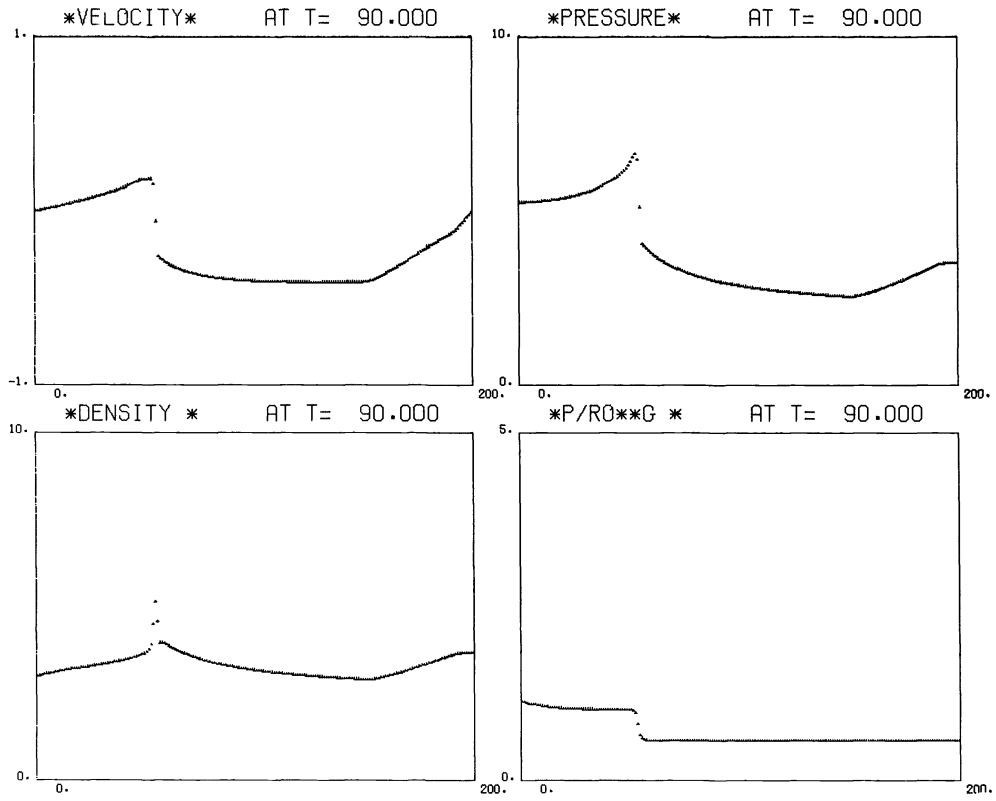


FIG. 6. *Interaction of shock and contact discontinuity at* $t = 90$.

Combining (A.5) and (A.6) we obtain

$$\frac{d}{d\beta}(\beta^{-1/2}\varepsilon(0,\beta)) = \frac{k}{4}\beta^{-1/2}\frac{d}{d\beta}\left[\beta^{-1/2}\frac{\partial(gA)}{\partial\alpha}(0,\beta)\right], \tag{A.7}$$

supplemented with the initial condition (see (3.2)),

$$\varepsilon(0,1) = 0. \tag{A.8}$$

Next, we examine the trajectory $(\alpha(t), \beta(t))$ which is the image of the grid-line $r = 0$.

Using the Lagrangian representation $\xi = \xi(t)$ for this curve, we get,

$$\xi'(t) = \frac{\partial\xi}{\partial\alpha}\alpha'(t) + \frac{\partial\xi}{\partial\beta}\beta'(t) = (\beta^{1/2} + 2\alpha\varepsilon(\alpha,\beta))\alpha'(t) + \tfrac{1}{2}\alpha\beta^{-1/2}\beta'(t) + O(\alpha^2).$$

On the other hand, from (5.2),

$$\xi'(t) = -A\rho u = -A(0)(\rho u)(0,\beta_0) - \frac{\partial(A\rho u)}{\partial\alpha}(0,\beta_0)\cdot\alpha(t)$$

$$-\frac{\partial(A\rho u)}{\partial\beta}(0,\beta_0)\cdot(\beta(t)-\beta_0) + O(t^2),$$

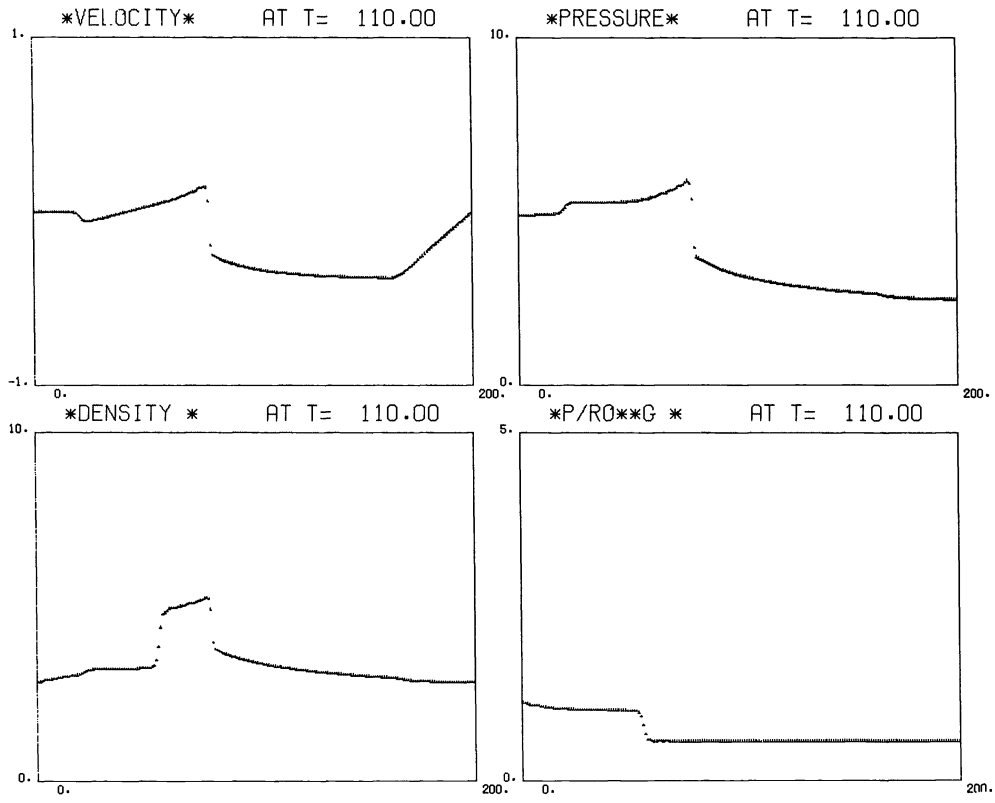


FIG. 7. *At $t=110$ the wave pattern after the interaction includes a weak converging shock, the original (diverging) shock, and a contact discontinuity.*

where β_0 is determined by (6.1). But (6.1) in conjunction with the characteristic relation $dp+gdu=0$, which is valid for the solution of the associated RP along the Γ^+-characteristic $\alpha=0$, implies

$$\frac{\partial(A\rho u)}{\partial\beta}(0,\beta_0)=A(0)\frac{\partial(\rho u)}{\partial\beta}(0,\beta_0)=0.$$

Hence, equating zero order terms in the two expressions for $\xi'(t)$ we have

$$\alpha'(0)=-k^{-1}\beta_0^{1/2}, \tag{A.9}$$

which is exactly (6.4), while for first order terms (in t) we have

$$\alpha'(0)\beta'(0)\beta_0^{-1/2}+\beta_0^{1/2}\alpha''(0)+2\varepsilon(0,\beta_0)\alpha'(0)^2=-\frac{\partial(A\rho u)}{\partial\alpha}(0,\beta_0)\cdot\alpha'(0).$$

Similarly,

$$\frac{dt}{dt}=\frac{\partial t}{\partial\alpha}\alpha'(t)+\frac{\partial t}{\partial\beta}\beta'(t)\equiv 1$$

implies

$$k\alpha'(0)\beta'(0)\beta_0^{-3/2}+2\alpha'(0)^2\eta(0,\beta_0)-k\alpha''(0)\beta_0^{-1/2}=0.$$

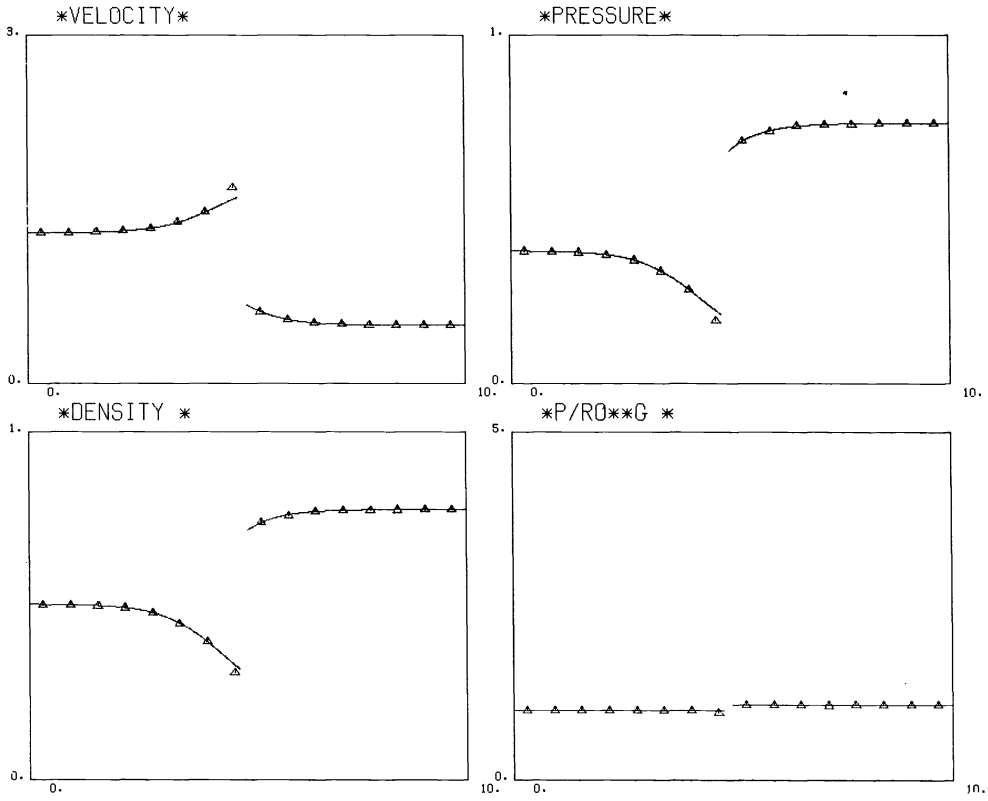


FIG. 8. *A steady-state flow profile in an expanding nozzle, using* 16 *mesh-points.*

Let us write the last two equations in the form

$$\alpha'(0)\beta_0^{-1/2}\beta'(0)+\beta_0^{1/2}\alpha''(0)=-2\varepsilon(0,\beta_0)\alpha'(0)^2-\frac{\partial(A\rho u)}{\partial\alpha}(0,\beta_0)\cdot\alpha'(0), \tag{A.10}$$

$$\beta_0^{-3/2}\alpha'(0)\beta'(0)-\alpha''(0)\beta_0^{-1/2}=-2k^{-1}\alpha'(0)^2\eta(0,\beta_0). \tag{A.11}$$

Multiplying (A.11) by β_0 and adding to (A.10) we get

$$2\beta_0^{-1/2}\beta'(0)=-2\alpha'(0)[\varepsilon(0,\beta_0)+k^{-1}\beta_0\eta(0,\beta_0)]-\frac{\partial(A\rho u)}{\partial\alpha}(0,\beta_0).$$

The expression in square brackets can be evaluated by (A.5), so that, in view of (A.9),

$$2\beta_0^{-1/2}\beta'(0)=-k\beta_0^{-1/2}\alpha'(0)\frac{\partial(gA)}{\partial\alpha}(0,\beta_0)-\frac{\partial(A\rho u)}{\partial\alpha}(0,\beta_0)$$

$$=\frac{\partial(gA)}{\partial\alpha}(0,\beta_0)-\frac{\partial(A\rho u)}{\partial\alpha}(0,\beta_0).$$

Finally, note that along $\xi(t)$, $A(\alpha,\beta)\equiv A(0)$, hence, $\partial A/\partial\alpha\,\alpha'(0)+\partial A/\partial\beta\cdot\beta'(0)=\partial A/\partial\alpha\cdot\alpha'(0)=0$, so that (6.5) follows from the last expression.

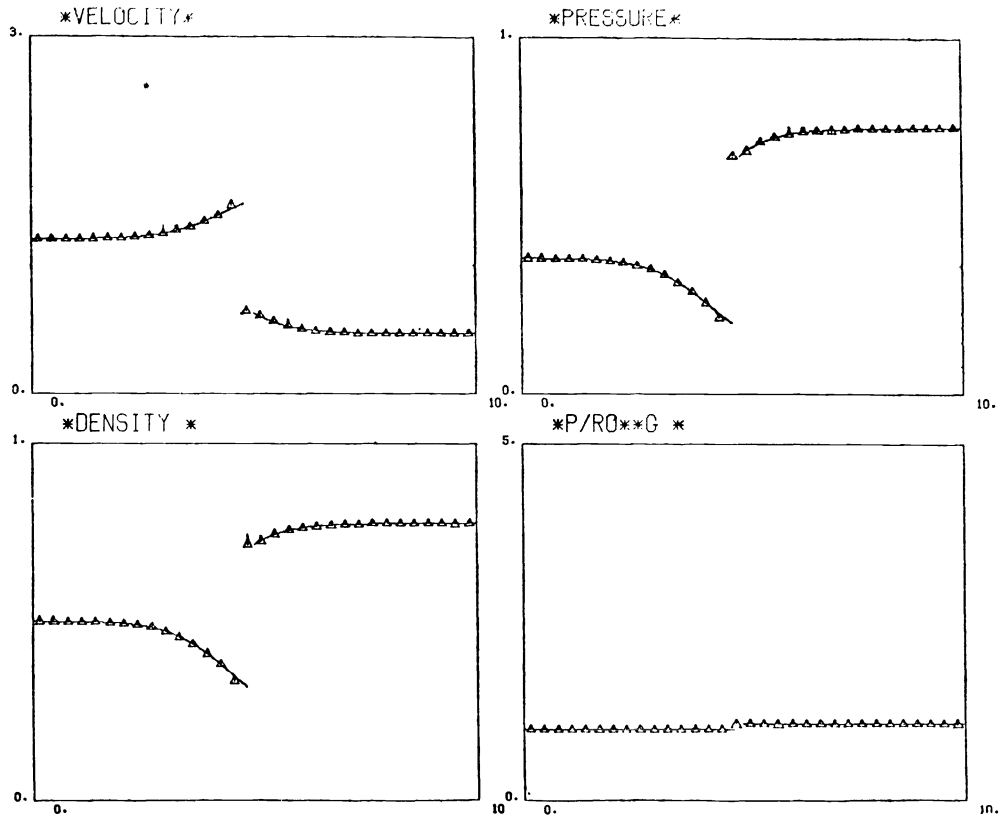


FIG. 9. *A steady-state flow profile as in Fig.* 8, *but using* 32 *mesh-points.*

REFERENCES

[1] S. ABARBANEL AND M. GOLDBERG, *Numerical solution of quasi-conservative hyperbolic systems—the cylindrical shock problem*, J. Comp. Phys., 10 (1972), pp. 1-21.

[2] M. BEN-ARTZI AND J. FALCOVITZ, *A second-order Godunov-type scheme for compressible fluid dynamics*, J. Comp. Phys., 55 (1984), pp. 1-32.

[3] P. COLELLA, *A direct Eulerian* MUSCL *scheme for gas dynamics*, Lawrence Berkeley Laboratory Report LBL-14104, Berkeley, CA, 1982.

[4] P. COLELLA AND P. R. WOODWARD, *The piecewise-parabolic method* (PPM) *for gas dynamical simulations*, J. Comp. Phys., 54 (1984), pp. 174-201.

[5] R. COURANT AND K. O. FRIEDRICHS, *Supersonic Flow and Shock Waves*, Interscience, New York, 1948.

[6] R. COURANT AND D. HILBERT, *Methods of Mathematical Physics* II, Interscience, New York, 1962.

[7] S. K. GODUNOV, *A finite difference method for the numerical computation and discontinuous solutions of the equations of fluid dynamics*, Mat. Sbornik, 47 (1959), pp. 271-295.

[8] A. HARTEN, *High resolution schemes for hyperbolic conservation laws*, J. Comp. Phys., 49 (1983), pp. 357-393.

[9] A. LAPIDUS, *Computation of radially symmetric shocked flows*, J. Comp. Phys., 8 (1971), pp. 106-118.

[10] G. R. SHUBIN, A. B. STEPHENS AND H. M. GLAZ, *Steady shock tracking and Newton's method applied to one-dimensional duct flow*, J. Comp. Phys., 39 (1981), pp. 364-374.

[11] G. A. SOD, *A numerical study of a converging cylindrical shock*, J. Fluid Mech., 83 (1977), pp. 785-794.

[12] B. VAN LEER, *Towards the ultimate conservative difference scheme* V, J. Comp. Phys., 32 (1979), pp. 101-136.

SIAM J. SCI. STAT. COMPUT.
Vol. 7, No. 3, July 1986

005

ON GENERATING TEST PROBLEMS FOR NONLINEAR PROGRAMMING ALGORITHMS*

RICHARD H. BARTELS† AND NEZAM MAHDAVI-AMIRI‡

Abstract. We present techniques for the generation of test problems which provide a way of constructing nonlinear programming problems from a wide variety of given functions. These techniques permit one to specify an arbitrary number of points, at each one of which the problem should satisfy some "interesting" conditions (i.e. be optimal or stationary or degenerate), and to determine the characteristics of functions and derivatives at these points (i.e. choose predetermined values for functions, gradients, Hessians and Lagrange multipliers). Moreover, there is a limited capacity to create test problems with special structure.

We give a sample of results obtained by using these techniques to generate test problems for two algorithms to solve problems having nonlinear-least-squares structure with nonlinear constraints.

Key words. optimization, testing, nonlinear constraints, numerical problem generation

1. Introduction. During the last decade a number of methods have been developed for minimizing general objective functions subject to general nonlinear constraints. These methods are characterized by their properties of global convergence, asymptotic superlinear rates of convergence, and formulation in terms of processes that can be efficiently and accurately implemented as numerical algorithms. Examples would include the following references: [5], [6], [15], [16], [25], [26], [29].

No case has to be made for the fact that, in translating these or other methods into mathematical software, a ready supply of test examples can be extremely useful. It is less often recognized that test examples can also be used to probe for oversights in the theory, and can therefore be useful in earlier and more fundamental stages of algorithm development. It was with the intention of exploring ways of making the generation of test examples more flexible that the present work was done. The goal in mind was that it should be easy to craft examples to probe an algorithm's behaviour, producing local "scenario" conditions that test the theoretical assumptions upon which the algorithm is based or that exercise critical sections of the algorithm. We believe that this work represents a step toward that goal.

In § 2 of this paper we review the commonly given necessary and sufficient conditions for a point to be an optimizer of a nonlinear programming problem. In § 3 we propose techniques for perturbing any given function so as to impose upon it a preselected value, gradient, and/or Hessian. These techniques permit the construction of the Hessian in a partitioned form so that its "projections" onto given subspaces can be specified. In § 4 we apply the content of §§ 2 and 3 for the construction of nonlinear programming problems. The material of this section is suggestive rather than comprehensive, and it is oriented toward the crafting of individual problems with any number of "interesting" points: maxima, minima, saddles, degeneracies, etc. In the last two sections we explore whether these techniques are at all amenable to the automatic generation of problems as well as the individualized design of test examples.

* Received by the editors June 16, 1983, and in final revised form April 8, 1985. This research was supported by the Natural Sciences and Engineering Research Council under grant A4076 and A2472 and by a Laidlaw Fellowship administered by the University of Waterloo Computer Science Department. Part of this work was submitted to the Department of Mathematical Sciences, Johns Hopkins University, Baltimore, Maryland, by Nezam Mahdavi-Amiri as a Ph.D. thesis.

† Department of Computer Science, University of Waterloo, Waterloo, Ontario, Canada N2L 3G1.

‡ Department of Computer Science, York University, Downsview, Ontario, Canada M3J 1P3.

In § 5 we give a brief survey of two constrained nonlinear least-squares programs that were written to explore the usefulness of these techniques. In § 6 we give some samples of our experience in automatically constructing problems for these codes.

The idea of devising methods to generate test problems is not new. References [2], [18], [20], [21], [23], [27], [31], [30], [35] provide a sampling of the literature. We believe that this work represents the first one in which multiple points can be specified for the test examples. An example of the specification process, as it might be done by hand, is included for illustration. An automatic generator, using the material to be presented here, has been prepared for the simple case of specifying single points as optimizers, and the techniques had enough flexibility that we were able to impose a special structure on the problems generated.

Test generation methods provide a supplement to collections of optimization problems; e.g. [4], [9], [17], [32]. The following references provide some discussions on software testing, and the reporting of test results, from a broader perspective: [7], [8], [22].

It should be brought out that this paper proposes some tools for generating test examples rather than a single, concrete method. The approach derives from some observations about the chain rule coupled with some conditions of interpolation that can be associated with objective and constraint functions in order to force one or more given points to satisfy predetermined conditions of activity, stationarity, optimality, degeneracy, or conditioning.

2. Optimality conditions. We are interested in solving problems of the form

$$\begin{aligned} &\underset{x}{\text{minimize}}\ \phi_0(x) \\ &\text{such that} \\ &\phi_j(x) = 0, \qquad j \in \mathbf{J}_E, \\ &\phi_j(x) \geqq 0, \qquad j \in \mathbf{J}_I, \end{aligned} \tag{2.1}$$

where $\mathbf{J}_E = \{1, \cdots, l\}$ is the index set of equality constraints and $\mathbf{J}_I = \{l+1, \cdots, l+m\}$ is the index set of inequality constraints. We will denote the combined index set by

$$\mathbf{J} \equiv \mathbf{J}_E \cup \mathbf{J}_I.$$

We denote the index set of the active inequality constraints at any point x by

$$\mathbf{J}_A(x) \equiv \{j \in \mathbf{J}_I \colon \phi_j(x) = 0\}.$$

Throughout this paper we will assume that all functions are twice continuously differentiable. We first state necessary conditions for a given point to be an optimizer for problem (2.1). In order that these conditions be applicable, constraint qualifications must hold at the point. Appropriate qualifications, and their underlying theory, can be found in [12]. In practice, the constraint qualifications given in this reference (or any other of suitable generality) are difficult to verify computationally. In order to implement algorithms, it is often assumed that the following more strict constraint qualifications hold:

Nondegeneracy assumption. Letting $\bar{x}$ stand for the optimizer, as well as for any point encountered by the algorithm, the vectors

$$\nabla \phi_j(\bar{x}), \qquad j \in \mathbf{J}_E \cup \mathbf{J}_A(\bar{x}) \tag{2.2}$$

are linearly independent.

This is more than enough to imply satisfaction of all of the constraint qualifications given in [12].

First-order necessary conditions. Let x^* be an optimizer for (2.1), and let an appropriate constraint qualification be satisfied at x^*. Then there exist λ_j^*, $j \in \mathbf{J}$ such that

$$\nabla\phi_0(x^*) - \sum_{j\in\mathbf{J}_E} \lambda_j^*\nabla\phi_j(x^*) - \sum_{j\in\mathbf{J}_A(x^*)} \lambda_j^*\nabla\phi_j(x^*) = 0, \tag{2.3}$$

$$\lambda_j^* \geqq 0, \qquad j \in \mathbf{J}_A(x^*),$$

$$\phi_j(x^*) = 0, \qquad j \in \mathbf{J}_E,$$

$$\phi_j(x^*) \geqq 0, \qquad j \in \mathbf{J}_I,$$

$$\lambda_j^*\phi_j(x^*) = 0, \qquad j \in \mathbf{J}_I.$$

Second-order necessary conditions. Let x^* be an optimizer for (2.1), and let an appropriate constraint qualification be satisfied at x^*. Suppose that there exist λ_j^* satisfying the first-order necessary conditions. Then for all

$$z \in \mathbf{R}^n \quad \text{s.t.} \quad z^T\nabla\phi_j(x^*) = 0, \qquad j \in \mathbf{J}_E,$$

$$z^T\nabla\phi_j(x^*) = 0, \qquad j \in \mathbf{J}_A(x^*)$$

we have

$$z^T\left[\nabla^2\phi_0(x^*) - \sum_{j\in\mathbf{J}_E} \lambda_j^*\nabla^2\phi_j(x^*) - \sum_{j\in\mathbf{J}_A(x^*)} \lambda_j^*\nabla^2\phi_j(x^*)\right] z \geqq 0. \tag{2.4}$$

For x^* to be a strict local minimum of problem (2.1) we need the conditions stated in the following result.

Second-order sufficient conditions. Let (x^*, λ^*) satisfy the first-order necessary conditions and an appropriate constraint qualification. If for every nonzero $z \in \mathbf{R}^n$ such that

$$z^T\nabla\phi_j(x^*) = 0, \qquad j \in J_E,$$

$$z^T\nabla\phi_j(x^*) = 0, \qquad j \in \mathbf{J}_A(x^*) \quad \text{and } \lambda_j^* > 0,$$

$$z^T\nabla\phi_j(x^*) \geqq 0, \qquad j \in \mathbf{J}_A(x^*) \quad \text{and } \lambda_j^* = 0,$$

it follows that

$$z^T\left[\nabla^2\phi_0(x^*) - \sum_{j\in\mathbf{J}_E} \lambda_j^*\nabla^2\phi_j(x^*) - \sum_{j\in\mathbf{J}_A(x^*)} \lambda_j^*\nabla^2\phi_j(x^*)\right] z > 0, \tag{2.5}$$

then x^* is a strict local minimum of problem (2.1).

DEFINITION. The function

$$L(x, \lambda) = \phi_0(x) - \sum_{j\in\mathbf{J}} \lambda_j\phi_j(x)$$

is the Lagrangian of the problem.

If we regard ∇ and ∇^2 to be differentiation operators with respect to x alone, and if the columns of the matrix Z form a basis for the orthogonal complement of the space spanned by the vectors $\nabla\phi_j(x^*)$ for $j \in \mathbf{J}_E \cup \mathbf{J}_A(x^*)$, then

(2.3) can be written as $\nabla L(x^*, \lambda^*) = 0$;

(2.4) says that $Z^T\nabla^2 L(x^*, \lambda^*)Z$ will be positive semidefinite at an optimizer;

(2.5) gives us conditions under which $Z^T\nabla^2 L(x^*, \lambda^*)Z$ will be positive definite.

The first-order conditions clearly imply that $\lambda_j^* = 0$ for all $j \in \mathbf{J}_I \backslash \mathbf{J}_A(x^*)$; hence at (x^*, λ^*) the function L can be written in the more explicit form indicated by the first- and second-order conditions.

The idea to be presented for generating test problems is straightforward: the functions $\phi_j(x)$, $j \in \mathbf{J} \cup \{0\}$, will be constructed from given functions $q_j(x)$ to which perturbing functions are added. The perturbing functions are chosen so as to force selected values, gradients, and/or Hessians to be taken on at selected points. It will be seen that this choice is achieved by the introduction of simple conditions of interpolation on the perturbing functions. The values, gradients, and/or Hessians of all of the ϕ_j can, in their turn, all be selected in a coordinated way so as to force the first- and/or second-order conditions to hold at a number of selected points. The choice of the $q_j(x)$ will be left open throughout the paper. Within limits, the choice may serve to provide a desired characteristic or structure (e.g. exponential growth, oscillation, boundedness below, sum of squares format, some degree of discontinuity, etc.).

3. Specifying values, gradients, and Hessians. Take a finite number, $s \geqq 1$, of selected points $x^{(1)}, \cdots, x^{(s)}$. Let

$$q_j: \mathbf{R}^n \to \mathbf{R}^1$$

and

$$\delta_{ji}, \gamma_{ji}, \Omega_{ji}: \mathbf{R}^1 \to \mathbf{R}^1$$

be functions,

$$\alpha_{ji} \in \mathbf{R}^1$$

be scalars,

$$d_{ji}, h_{ji} \in \mathbf{R}^n$$

be vectors, and

$$D_{ji} \in \mathbf{R}^{n \times n}$$

be symmetric matrices.

Consider letting

$$\phi_j(x) = q_j(x) + \sum_{i=1}^{s} \{\alpha_{ji}\delta_{ji}[d_{ji}^T(x - x^{(i)})] + \gamma_{ji}[h_{ji}^T(x - x^{(i)})] + \Omega_{ji}[\tfrac{1}{2}(x - x^{(i)})^T D_{ji}(x - x^{(i)})]\}, \tag{3.1}$$

for $j \in \mathbf{J} \cup \{0\}$ and $i = 1, \cdots, s$.

The functions ϕ_j will provide us, later on, with the objective and constraint functions for a nonlinear programming problem of the form (2.1). The functions

$$q_j(x), \qquad j \in \mathbf{J} \cup \{0\}$$

may be chosen at will to provide some behavioural properties for the ϕ_j.

We will see that the values of the ϕ_j can be determined through the selection of the scalars and vectors

$$\alpha_{ji} \in \mathbf{R}^1 \quad \text{and} \quad d_{ji} \in \mathbf{R}^n,$$

that the gradients can be determined through the selection of the matrices

$$h_{ji} \in \mathbf{R}^n,$$

and that the Hessians can be determined through the selection of the matrices

$$D_{ji} \in \mathbf{R}^{n \times n}.$$

The process of imposing the desired values, gradients, and Hessians on the ϕ_j will require that the functions

$$\Omega_{ji},\ \gamma_{ji},\ \delta_{ji}$$

satisfy conditions of interpolation to be discussed below.

We begin by establishing that the gradient of ϕ_j is:

$$\nabla\phi_j(x) = \nabla q_j(x) + \sum_{k=1}^{s} \{\alpha_{ji}\delta'_{ji}[d_{ji}^T(x-x^{(i)})]d_{ji} + \gamma'_{ji}[h_{ji}^T(x-x^{(i)})]h_{ji} + \Omega'_{ji}[\tfrac{1}{2}(x-x^{(i)})^T D_{ji}(x-x^{(i)})]\} \tag{3.2}$$

and the Hessian of ϕ_j is:

$$\nabla^2\phi_j(x) = \nabla^2 q_j(x) + \sum_{i=1}^{s} \{\alpha_{ji}\delta''_{ji}[d_{ji}^T(x-x^{(i)})]d_{ji}d_{ji}^T + \gamma''_{ji}[h_{ji}^T(x-x^{(i)})]h_{ji}h_{ji}^T + \Omega''_{ji}[\tfrac{1}{2}(x-x^{(i)})^T D_{ji}(x-x^{(i)})]D_{ji}(x-x^{(i)})(x-x^{(i)})^T D_{ji} + \Omega'_{ji}[\tfrac{1}{2}(x-x^{(i)})^T D_{ji}(x-x^{(i)})]D_{ji}\}. \tag{3.3}$$

Let us introduce the following quantities:

$$\theta_{ji}(x) = d_{ji}^T(x-x^{(i)}), \qquad \theta_{ji}^{(t)} \equiv \theta_{ji}(x^{(t)}) = d_{ji}^T(x^{(t)}-x^{(i)}),$$

$$\eta_{ji}(x) = h_{ji}^T(x-x^{(i)}), \qquad \eta_{ji}^{(t)} \equiv \eta_{ji}(x^{(t)}) = h_{ji}^T(x^{(t)}-x^{(i)}),$$

$$\tau_{ji}(x) = \tfrac{1}{2}(x-x^{(i)})^T D_{ji}(x-x^{(i)}), \qquad \tau_{ji}^{(t)} \equiv \tau_{ji}(x^{(t)}) = \tfrac{1}{2}(x^{(t)}-x^{(i)})^T D_{ji}(x^{(t)}-x^{(i)}).$$

Note that $\theta_{ji}^{(i)} = \eta_{ji}^{(i)} = \tau_{ji}^{(i)} = 0$.

In terms of these quantities (3.2) and (3.3) become, respectively,

$$\nabla\phi_j(x) = \nabla q_j(x) + \sum_{i=1}^{s} \{\alpha_{ji}\delta'_{ji}[\theta_{ji}(x)]d_{ji} + \gamma'_{ji}[\eta_{ji}(x)]h_{ji} + \Omega'_{ji}[\tau_{ji}(x)]D_{ji}(x-x^{(i)})\}$$

and

$$\nabla^2\phi_j(x) = \nabla^2 q_j(x) + \sum_{i=1}^{s} \{\alpha_{ji}\delta''_{ji}[\theta_{ji}(x)]d_{ji}d_{ji}^T + \gamma''_{ji}[\eta_{ji}(x)]h_{ji}h_{ji}^T + \Omega''_{ji}[\tau_{ji}(x)]D_{ji}(x-x^{(i)})(x-x^{(i)})^T D_{ji} + \Omega'_{ji}[\tau_{ji}(x)]D_{ji}\}.$$

The vectors d_{ji} and the functions δ_{ji} may be exploited to define $\phi_j(x^{(t)})$, for all j, t. The vectors h_{ji} and the functions γ_{ji} may be used to specify the gradient of $\phi_j(x^{(t)})$ for all j, t. The matrices D_{ji} and the functions Ω_{ji} may be defined so that the Hessian of $\phi_j(x^{(t)})$ is specified for all j and t. To make these observations more concrete, we begin by pointing out how the quantities

$$\phi_j(x^{(t)}), \quad \nabla\phi_j(x^{(t)}), \quad \nabla^2\phi_j(x^{(t)})$$

depend upon the values of the functions

$$\delta_{ji}, \quad \gamma_{ji}, \quad \Omega_{ji}$$

and their first and second derivatives at the argument values

$$\theta_{ji}^{(t)}, \quad \eta_{ji}^{(t)}, \quad \tau_{ji}^{(t)}.$$

Some critical assumptions must be made about the points $x^{(i)}$, $i = 1, \cdots, s$ in order to proceed. For a fixed j the points must interact with the vectors d_{ji}, $i = 1, \cdots, s$ so that

$$\theta_{ji}^{(t)} = d_{ji}^T(x^{(t)} - x^{(i)}) \neq 0$$

for all i, $t = 1, \cdots, s$ but $i \neq t$. Similarly, for h_{ji} and D_{ji}, $i = 1, \cdots, s$, we must have

$$\eta_{ji}^{(t)} = h_{ji}^T(x^{(t)} - x^{(i)}) \neq 0$$

and

$$\tau_{ji}^{(t)} = \tfrac{1}{2}(x^{(t)} - x^{(i)})D_{ji}(x^{(t)} - x^{(i)}) \neq 0$$

for i, $t = 1, \cdots, s$ but $i \neq t$. These assumptions frequently hold for small s, even for randomly chosen x's, d's, h's and random nonsingular D's. Should the assumption fail to hold, it will be necessary to re-select an offending x, or alternatively revise a d, h, or D.

Now, if a fixed index t is selected, and if the functions δ_{ji}, γ_{ji}, and Ω_{ji} have been selected so that the following conditions hold

$$\delta_{ji}[\theta_{ji}^{(t)}] = 0 \quad \text{for all } i \neq t,\ i = 1, \cdots, s,$$
$$\phi_{jt}[\theta_{jt}^{(t)}] = \delta_{jt}[0] = 1,$$
$$\gamma_{ji}[\eta_{ji}^{(t)}] = 0 \quad \text{for all } i = 1, \cdots, s \text{ (including } t),$$

and

$$\Omega_{ji}[\tau_{ji}^{(t)}] = 0 \quad \text{for all } i = 1, \cdots, s \text{ (including } t),$$

then, from (3.1), $\phi_j(x^{(t)})$ reduces to

$$\phi_j(x^{(t)}) = q_j(x^{(t)}) + \alpha_{jt}. \tag{3.4}$$

The above conditions on the functions δ, γ, and Ω are the conditions of interpolation mentioned previously. It should be clear from (3.1) why these conditions have their particular form and why they lead to (3.4). Since the choice of these functions is up to us, we may construct them so that the given values of 1 and 0 are taken on at the given points θ, η, and τ. Using these observations, we see that the value of $\phi_j(x^{(t)})$ can be specified through the selection of α_{jt}.

A similar discussion involving interpolated values of 1 and 0 derived from (3.2) for δ'_{ji}, γ'_{ji}, Ω'_{ji} will lead to the equation

$$\nabla\phi_j(x^{(t)}) = \nabla q_j(x^{(t)}) + h_{jt} \tag{3.5}$$

from which the choice of h_{jt} will serve to specify $\nabla\phi_j(x^{(t)})$. Finally, a discussion involving interpolated values of 1 and 0 derived from (3.3) for δ''_{ji}, γ''_{ji}, and Ω''_{ji}, will serve to specify $\nabla^2\phi_j(x^{(t)})$ as

$$\nabla^2\phi_j(x^{(t)}) = \nabla^2 q_j(x^{(t)}) + D_{jt}. \tag{3.6}$$

It should be remarked that, if any functional value $\phi_j(x^{(t)})$ is not to be set specifically, then the corresponding function δ_{jt} may be set identically to zero. This choice of δ_{jt} would not violate any conditions of interpolation that may arise from other specifications, since those conditions only insist that the values, first, or second derivatives of δ_{jt} are to be zero at other points. Likewise, γ_{jt} or Ω_{jt} may be set identically to zero when the corresponding gradient or Hessian is not demanded.

If all of the conditions of interpolation stated or alluded to above are collected together, we find that the following must be imposed upon δ_{ji}:

$$
\begin{aligned}
&\delta_{ji}[0] = 1, \\
&\delta_{ji}[\theta_{ji}^{(t)}] = 0 \quad \text{for all } i \neq t, \text{ and for all } j \text{ such that } \phi_j \text{ is specified at } x^{(t)}, \\
&\delta'_{ji}[\theta_{ji}^{(t)}] = 0 \quad \text{for all } i, \text{ and for all } j \text{ such that } \nabla\phi_j \text{ is specified at } x^{(t)}, \\
&\delta''_{ji}[\theta_{ji}^{(t)}] = 0 \quad \text{for all } i, \text{ and for all } j \text{ such that } \nabla^2\phi_j \text{ is specified at } x^{(t)}.
\end{aligned} \tag{3.7}
$$

All other values and derivatives of δ_{ji} are immaterial.

Likewise, for γ_{ji}:

$$
\begin{aligned}
&\gamma'_{ji}[0] = 1, \\
&\gamma_{ji}[\eta_{ji}^{(t)}] = 0 \quad \text{for all } i, \text{ and for all } j \text{ such that } \phi_j \text{ is specified at } x^{(t)}, \\
&\gamma'_{ji}[\eta_{ji}^{(t)}] = 0 \quad \text{for all } i \neq t, \text{ and for all } j \text{ such that } \nabla\phi_j \text{ is specified at } x^{(t)}, \\
&\gamma''_{ji}[\eta_{ji}^{(t)}] = 0 \quad \text{for all } i, \text{ and for all } j \text{ such that } \nabla^2\phi_j \text{ is specified at } x^{(t)}.
\end{aligned} \tag{3.8}
$$

And finally for Ω_{ji}:

$$
\begin{aligned}
&\Omega'_{ji}[0] = 1, \\
&\Omega_{ji}[\tau_{ji}^{(t)}] = 0 \quad \text{for all } i, \text{ and for all } j \text{ such that } \phi_j \text{ is specified at } x^{(t)}, \\
&\Omega'_{ji}[\tau_{ji}^{(t)}] = 0 \quad \text{for all } i \neq t, \text{ and for all } j \text{ such that } \nabla\phi_j \text{ or } \nabla^2\phi_j \\
&\qquad\qquad \text{is specified at } x^{(t)}, \\
&\Omega''_{ji}[\tau_{ji}^{(t)}] = 0 \quad \text{for all } i \neq t, \text{ and for all } j \text{ such that } \nabla^2\phi_j \text{ is specified at } x^{(t)}.
\end{aligned} \tag{3.9}
$$

Programmatically:

(1) Choose $x^{(1)}, \cdots, x^{(s)}$.

(2) For each $t = 1, \cdots, s$

(2.a) To specify ϕ_j at $x^{(t)}$, select a value for $\phi_j(x^{(t)})$, choose d_{jt} at will, and set

$$\alpha_{jt} = \phi_j(x^{(t)}) - q_j(x^{(t)}).$$

(2.b) To specify $\nabla\phi_j$ at $x^{(t)}$, select a vector for $\nabla\phi_j(x^{(t)})$ and set

$$h_{jt} = \nabla\phi_j(x^{(t)}) - \nabla q_j(x^{(t)}).$$

(2.c) To specify $\nabla^2\phi_j$ at $x^{(t)}$, select a symmetric matrix for $\nabla^2\phi_j(x^{(t)})$ and set

$$D_{jt} = \nabla^2\phi_j(x^{(t)}) - \nabla^2 q_j(x^{(t)}).$$

(3) For all i, j, and $t \neq i$

(3.a) Determine the quantities

$$
\begin{aligned}
\theta_{ji}^{(t)} &= d_{ji}^T(x^{(t)} - x^{(i)}), \\
\eta_{ji}^{(t)} &= h_{ji}^T(x^{(t)} - x^{(i)}), \\
\tau_{ji}^{(t)} &= \tfrac{1}{2}(x^{(t)} - x^{(i)})^T D_{ji}(x^{(t)} - x^{(i)}).
\end{aligned}
$$

(3.b) Check that each of these quantities is nonzero. In the (infrequent) event that a zero occurs for any $i, j, t \neq i$, either $x^{(i)}$ or $x^{(t)}$ must be discarded, or d_{ji}, h_{ji}, or D_{ji} must be revised. Return to step (1).

(4) Collect together from (3.7), (3.8), and (3.9) all of the conditions of interpolation that are required by the specifications made in step (2) above, and construct any functions δ_{ji}, γ_{ji}, Ω_{ji} satisfying those conditions.

This outline provides only the background for constructing test examples. The arbitrary determination of values, gradients, and Hessians is not enough. We proceed in the next section to suggest ways of determining $\phi_j(x^{(t)})$, $\nabla\phi_j(x^{(t)})$, and $\nabla^2\phi_j(x^{(t)})$ to take account of the material in § 2.

As a closing remark, it should be stressed that the functions δ, γ, and Ω can be drawn from any class of functions for which it makes sense to set up and solve the Hermite interpolation problems listed above. Reference [1] gives algorithms for doing this in case we choose to use polynomials. Other function classes are possible, however, such as splines, trigonometric functions, etc. If the functions q are also chosen from the same class, the structure of that class, if any, may colour the problem as a whole. We illustrate this in a trivial way at the end of the paper by arranging to generate some least-squares problems.

4. Constructing test examples. The optimization problem (2.1) takes on the following form when ϕ_j is structured as in (3.1):

$$\underset{x}{\text{minimize}}\ \phi_0(x) \equiv q_0(x) + \sum_{i=1}^{s} \{\alpha_{0i}\delta_{0i}[d_{0i}^T(x-x^{(i)})] + \gamma_{0i}[h_{0i}^T(x-x^{(i)})] + \Omega_{0i}[\tfrac{1}{2}(x-x^{(i)})^T D_{0i}(x-x^{(i)})]\},$$

such that

$$\phi_j(x) \equiv q_j(x) + \sum_{i=1}^{s} \{\alpha_{ji}\delta_{ji}[d_{ji}^T(x-x^{(i)})] + \gamma_{ji}[h_{ji}^T(x-x^{(i)})] + \Omega_{ji}[\tfrac{1}{2}(x-x^{(i)})^T D_{ji}(x-x^{(i)})]\} = 0, \qquad j \in \mathbf{J}_E,$$

$$\phi_j(x) \equiv q_j(x) + \sum_{i=1}^{s} \{\alpha_{ji}\delta_{ji}[d_{ji}^T(x-x^{(i)})] + \gamma_{ji}[h_{ji}^T(x-x^{(i)})] + \Omega_{ji}[\tfrac{1}{2}(x-x^{(i)})^T D_{ji}(x-x^{(i)})]\} \geqq 0, \qquad j \in \mathbf{J}.$$

Let $(x^{(t)}, \lambda^{(t)})$ be a chosen Lagrange pair, $t \in \{1, \cdots, s\}$. The point $x^{(t)}$ will be "interesting" if this pair interacts in some substantial way with the first- and second-order conditions given by (2.3), (2.4), and (2.5). Specifying that $x^{(t)}$, $\lambda^{(t)}$ will satisfy all of the equalities and inequalities of these conditions is, of course, the way of specifying that $x^{(t)}$ shall be one of the optimizers of (2.1), but there are other possibilities. Lagrange pairs that violate some subset of the equalities or inequalities (or that satisfy more than their fair share of these equalities and inequalities or that do not satisfy enough of the inequalities in a strict sense) are also of interest in providing stationary points, saddle points, problems with degeneracies, problems without feasibilities, etc.

In this presentation it is neither possible nor desirable to consider all of the conditions that one might wish to impose upon an (x, λ) pair. We will restrict ourselves to a general discussion about how to cause $x^{(t)}$ to be an optimizer. Then we will close with a concrete example in which we specify two points for a problem in two variables, choosing one of the points to be optimal and the other to be stationary. Hopefully, this will be sufficient to suggest other possibilities.

4.1. Specifying optimizers. For $x^{(t)}$ to be one of the optimizers of problem (2.1), we are interested, primarily, in making sure that $x^{(t)}$ is feasible, that $\lambda^{(t)}$ is "in kilter",

that $\nabla L(x^{(t)}, \lambda^{(t)}) = 0$, and that $Z^T \nabla^2 L(x^{(t)}, \lambda^{(t)}) Z$ is positive definite for an appropriate matrix Z. This will make it our primary concern to specify nonnegative values for $\phi_j, j \in \mathbf{J}_I$, zero values for $\phi_j, j \in \mathbf{J}_E \cup \mathbf{J}_{A(x^{(t)})}$, the zero vector for ∇L, and a specified matrix for $\nabla^2 L$. These last two specifications will be made on L indirectly by first setting all of the $\phi_j, j \in \mathbf{J}$, and then setting $\nabla \phi_0$ and $\nabla^2 \phi_0$ appropriately.

(1) Choose functions q_0 and q_j for $j \in \mathbf{J} \cup \{0\}$. Choose $\mathbf{J}_E$ and $\mathbf{J}_I$.

(2) For each $t = 1, \cdots, s$

(2.a) Choose $\mathbf{J}_{A^{(t)}} \subseteq \mathbf{J}_I$ to be the index set denoting the active inequality constraints at $x^{(t)}$.

(2.b) Select $x^{(t)}$ and $\lambda^{(t)}$. The values $\lambda_j^{(t)}$, $j \in \mathbf{J}$ are to be consistent with $\mathbf{J}_E$, $\mathbf{J}_I$, and $\mathbf{J}_{A^{(t)}}$ in the sense that $\lambda_j^{(t)} = 0$, $j \in \mathbf{J}_I \backslash \mathbf{J}_{A^{(t)}}$ and $\lambda_j^{(t)} \geqq 0$, $j \in \mathbf{J}_{A^{(t)}}$.

(2.c) Set

$$\alpha_{jt} = -q_j(x^{(t)}), \qquad j \in \mathbf{J}_E \cup \mathbf{J}_A(x^{(t)})$$

and

$$\alpha_{jt} > -q_j(x^{(t)}), \qquad j \in \mathbf{J}_I \backslash \mathbf{J}_A(x^{(t)}).$$

(2.d) Choose d_{ji} so that $\theta_{ji}^{(t)} = d_{ji}^T (x^{(t)} - x^{(i)}) \neq 0$ for all j and all $i \neq t$.

(3) Functions δ_{ji} must be constructed for $j \in \mathbf{J}$ and $i = 1, \cdots, s$ so that they satisfy a simplified form of (3.7), namely:

$$\delta_{ji}[0] = 1 \quad \text{for all } i,$$
$$\delta_{ji}[\theta_{ji}^{(t)}] = 0 \quad \text{for all } i \neq t.$$

(4) Define

$$\phi_j(x) \equiv q_j(x) + \sum_{i=1}^{s} \alpha_{ji} \delta_{ji}[d_{ji}^T (x - x^{(i)})].$$

(5) A gradient must be imposed upon ϕ_0 so that $\nabla L(x^{(t)}, \lambda^{(t)}) = 0$ for each $t = 1, \cdots, s$. Set

$$h_{0t} = -\nabla q_0(x^{(t)}) + \sum_{j \in \mathbf{J}} \lambda_j^{(t)} \nabla \phi_j(x^{(t)}).$$

(6) A Hessian must be imposed upon ϕ_0 so that $\nabla^2 L(x^{(t)}, \lambda^{(t)})$ is predetermined for each $t = 1, \cdots, s$.

(6.a) Choose $\nabla^2 L(x^{(t)}, \lambda^{(t)})$ as the desired Hessian of L.

(6.b) Set

$$D_{0t} = \nabla^2 L(x^{(t)}, \lambda^{(t)}) - \nabla^2 q_0(x^{(t)}) + \sum_{j \in \mathbf{J}} \lambda_j^{(t)} \nabla^2 \phi_j(x^{(t)}).$$

(7) Determine whether

$$\eta_{0i}^{(t)} = h_{0i}^T (x^{(t)} - x^{(i)}) \neq 0,$$
$$\tau_{0i}^{(t)} = \tfrac{1}{2} (x^{(t)} - x^{(i)})^T D_{0t} (x^{(t)} - x^{(i)}) \neq 0$$

for all $i \neq t$. If any of these quantities are zero, return to (2) to select new x's.

(8) Functions γ_{0i}, Ω_{0i} must be constructed for $i = 1, \cdots, s$ so that

$$\begin{aligned}
&\gamma'_{0i}[0] = 1, \\
&\gamma'_{0i}[\eta_{0i}^{(t)}] = 0 \quad \text{for all } i \neq t, \\
&\gamma''_{0i}[\eta_{0i}^{(t)}] = 0 \quad \text{for all } i \text{ (including } t), \\
&\Omega'_{0i}[\tau_{0i}^{(t)}] = 0 \quad \text{for all } i \neq t, \\
&\Omega'_{0i}[0] = 1, \\
&\Omega''_{0i}[\tau_{0i}^{(t)}] = 0 \quad \text{for all } i \neq t.
\end{aligned}$$

(9) Define

$$\phi_0(x) = q_0(x) + \sum_{i=1}^{s} \{\gamma_{0i}[h_{0i}^T(x - x^{(i)})] + \Omega_{0i}[\tfrac{1}{2}(x - x^{(i)})^T D_{0i}(x - x^{(i)})]\}.$$

Remarks. (a) Vectors can be selected arbitrarily to define $\nabla\phi_j(x^{(t)})$ for $j \in \mathbf{J}$. By setting

$$h_{jt} = \nabla\phi_j(x^{(t)}) - \nabla q_j(x^{(t)})$$

and by checking that $h_{ji}^T(x^{(t)} - x^{(i)}) \neq 0$ for all j and $i \neq t$, functions γ_{ji} can be constructed according to the conditions of interpolation given in (3.8) in order to impose the desired gradients on the equality and active-inequality constraint functions. Degenerate problems can be attempted by choosing gradient vectors which are linearly dependent.

The definition of each ϕ_j in step (4) above for which $\nabla\phi_j$ is to be specified will have to be expanded to include appropriate terms of the form

$$\gamma_{ji}[h_{ji}^T(x - x^{(i)})].$$

Similarly, symmetric matrices can be selected arbitrarily to define $\nabla^2\phi_j(x^{(t)})$ for $j \in \mathbf{J}$. The definition of each ϕ_j in step (4) above for which $\nabla^2\phi_j$ is to be specified will have to be expanded to include appropriate terms of the form

$$\Omega_{ji}[\tfrac{1}{2}(x - x^{(i)})^T D_{ji}(x - x^{(i)})].$$

(b) The $\lambda_j^{(t)}$ may be chosen arbitrarily, consistent only with the inequalities of (2.3). The selection of $\lambda_j^{(t)} = 0$ for one or more $j \in \mathbf{J}_A(x^{(t)})$ often provides difficulty for algorithms.

(c) If the matrix $\nabla^2 L$ is chosen to be positive definite, then the inequalities of (2.4) and (2.5) will be satisfied automatically.

(d) Indefinite or ill-conditioned test problems can be obtained by letting

$$h_{jt} = -\nabla q_j(x^{(t)}) + w_j^{(t)}, \qquad j \in \mathbf{J}_E \cup \mathbf{J}_{A^{(t)}},$$

where the vectors $w_j^{(t)}$ form an orthonormal set. This will result in:

$$\nabla\phi_j(x^{(t)}) = w_j^{(t)}, \qquad j \in \mathbf{J}_E \cup \mathbf{J}_{A^{(t)}}.$$

Let $W^{(t)}$ stand for the matrix whose columns are given by the $w_j^{(t)}$; let the vectors $z_l^{(t)}$ constitute a basis for the orthogonal complement of the space spanned by the $w_j^{(t)}$, and let $Z^{(t)}$ represent the matrix whose columns are given by the $z_l^{(t)}$. Consequently

$$\begin{bmatrix} W^{(t)T} \\ Z^{(t)T} \end{bmatrix} [W^{(t)} \quad Z^{(t)}] = \begin{bmatrix} I & 0 \\ 0 & I \end{bmatrix}.$$

Then let

$$\nabla^2 L(x^{(t)}, \lambda^{(t)}) = [\, W^{(t)} \quad Z^{(t)}] \begin{bmatrix} \sigma V_1^{(t)} & 0 \\ 0 & \tau V_2^{(t)} \end{bmatrix} \begin{bmatrix} W^{(t)T} \\ Z^{(t)T} \end{bmatrix} \begin{matrix} \}l+|\mathbf{J}_{A^{(t)}}| \\ \}n-l-|\mathbf{J}_{A^{(t)}}| \end{matrix}$$

with positive-definite matrices $V_i^{(t)}$, $i=1,2$. The matrix $\nabla^2 L$ will be indefinite if and only if one of σ or τ is nonpositive. For any nonzero vector $y \in \mathbf{R}^n$ we may write

$$y = y_1 + y_2,$$

where y_1 is in the space spanned by $w_j^{(t)}$ and y_2 is orthogonal to this space. The equality

$$y^T \nabla \phi_j(x^{(t)}) = 0, \qquad j \in \mathbf{J}_E \cup \mathbf{J}_{A^{(t)}},$$

is equivalent to $y_1 = 0$ since $\nabla \phi_j(x^{(t)}) = w_j^{(t)}$. This leads to the equation

$$y^T \nabla^2 L(x^{(t)}, \lambda^{(t)}) y = \tau y_2^T V_2^{(t)} y_2,$$

which guarantees the satisfaction of the second-order conditions for all values of σ, provided that $\tau > 0$. For ill-conditioned examples, $V_1^{(t)}$ and $V_2^{(t)}$ can be chosen accordingly. This scheme of generating $\nabla^2 L$ in a partitioned format in order to obtain ill-conditioned projected Hessians is necessary in the light of the results contained in [34].

4.2. Something specific. We close this section with a concrete example which uses the material under discussion in a more ambitious way.

Let

$$q_0(x) = q_0(x_1, x_2) = -e^{-(x_1^2 - x_2^2)},$$
$$q_1(x) = q_1(x_1, x_2) = 5 - x_1^3 x_2^3,$$
$$q_2(x) = q_2(x_1, x_2) = 2 - x_1^4 - x_2^4.$$

We will use these functions to construct ϕ_0, ϕ_1, and ϕ_2 for the problem

$$\begin{aligned} &\underset{x}{\text{minimize}}\ \phi_0(x) \\ &\text{subject to} \\ &\phi_1(x) \geqq 0, \\ &\phi_2(x) \geqq 0 \end{aligned}$$

so that at the point

$$x^{(1)} = \begin{bmatrix} x_1^{(1)} \\ x_2^{(1)} \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

we will have

$$\begin{aligned} &\phi_1(x^{(1)}) = 0, \\ &\phi_2(x^{(1)}) > 0, \\ &\nabla \phi_1(x^{(1)}) = \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \\ &\lambda_1^{(1)} = 0, \\ &\lambda_2^{(1)} = 1, \\ &\nabla L(x^{(1)}, \lambda^{(1)}) = 0 \end{aligned}$$

and

$$\nabla^2 L(x^{(1)}, \lambda^{(1)}) = \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix}.$$

The ϕ's are further to be constructed so that at the point

$$x^{(2)} = \begin{bmatrix} x_1^{(2)} \\ x_2^{(2)} \end{bmatrix} = \begin{bmatrix} -1 \\ -1 \end{bmatrix}$$

we will have

$$\phi_1(x^{(2)}) > 0,$$
$$\phi_2(x^{(2)}) = 0,$$
$$\lambda_1^{(2)} = 1,$$
$$\lambda_2^{(2)} = -1$$

and

$$\nabla L(x^{(2)}, \lambda^{(2)}) = 0.$$

The above (subject to *post facto* verification) should cause $x^{(1)}$ to be a degenerate optimizer and $x^{(2)}$ to be a stationary point.

Since we wish to set the value of ϕ_1 at both $x^{(1)}$ and $x^{(2)}$, and its gradient at $x^{(1)}$, we need to set

$$\phi_1(x) \equiv q_1(x) + \alpha_{11}\delta_{11}[d_{11}^T(x - x^{(1)})] + \alpha_{12}\delta_{12}[d_{12}^T(x - x^{(2)})] + \gamma_{11}[h_{11}^T(x - x^{(1)})].$$

Since only the value of ϕ_2 is to be set at both points, we need

$$\phi_2(x) \equiv q_2(x) + \alpha_{21}\delta_{21}[d_{21}^T(x - x^{(1)})] + \alpha_{22}\delta_{22}[d_{22}^T(x - x^{(2)})].$$

Finally, the gradient of the Lagrangian is to be set to zero at both points, by setting $\nabla\phi_0$ appropriately, and the Hessian of the Lagrangian is to be set at $x^{(1)}$ by setting $\nabla^2\phi_0$. This means that

$$\phi_0(x) \equiv q_0(x) + \gamma_{01}[h_{01}^T(x - x^{(1)})] + \gamma_{02}[h_{02}^T(x - x^{(2)})] + \Omega_{01}[\tfrac{1}{2}(x - x^{(1)})^T D_{01}(x - x^{(1)})].$$

It is convenient to begin by recording the values, gradients, and Hessians which we might need from q_0, q_1, and q_2:

For $x^{(1)}$:

Function	Value	Gradient	Hessian
q_0	-1	$\begin{bmatrix} -2 \\ 2 \end{bmatrix}$	$\begin{bmatrix} -2 & 4 \\ 4 & -6 \end{bmatrix}$
q_1	4	$\begin{bmatrix} -3 \\ -3 \end{bmatrix}$	$\begin{bmatrix} -6 & -9 \\ -9 & -6 \end{bmatrix}$
q_2	0	$\begin{bmatrix} -4 \\ -4 \end{bmatrix}$	$\begin{bmatrix} -12 & 0 \\ 0 & -12 \end{bmatrix}$

For $x^{(2)}$:

Function	Value	Gradient
q_0	-1	$\begin{bmatrix}-2\\2\end{bmatrix}$
q_1	4	$\begin{bmatrix}3\\3\end{bmatrix}$
q_2	0	$\begin{bmatrix}4\\4\end{bmatrix}$

We begin by setting the values of ϕ_1 and ϕ_2. The value $\phi_1(x^{(1)})=0$ may be set by letting

$$\alpha_{11}=-q_1(x^{(1)})=-4.$$

The value $\phi_2(x^{(2)})=0$ may be set by letting

$$\alpha_{22}=-q_2(x^{(2)})=0.$$

To set $\phi_2(x^{(1)})>0$, we let

$$\alpha_{21}>-q_2(x^{(1)})=0,$$

specifically:

$$\alpha_{21}=1.$$

To set $\phi_1(x^{(2)})>0$, we let

$$\alpha_{12}>-q_1(x^{(2)})=-4,$$

specifically:

$$\alpha_{12}=0.$$

Choose d_{11}, d_{12}, d_{21}, and d_{22} "arbitrarily":

$$d_{11}=d_{21}=\begin{bmatrix}1\\1\end{bmatrix}$$

and

$$d_{12}=d_{22}=\begin{bmatrix}-1\\-1\end{bmatrix}.$$

This will mean that

$$d_{ji}^T(x^{(t)}-x^{(i)})=-4$$

for $j=1,2$, $i=1,2$, $t=1,2$, and $i\neq t$. This (particularly economical) choice of d's will mean that the functions δ must satisfy only the following conditions of interpolation:

for δ_{11}:

$$\delta_{11}(0)=1,$$

$$\delta_{11}(-4)=0,$$

$$\delta'_{11}(0)=0;$$

for δ_{12}:

$$\delta_{12}(0) = 1,$$
$$\delta_{12}(-4) = 0,$$
$$\delta'_{12}(-4) = 0;$$

for δ_{21}:

$$\delta_{21}(0) = 1,$$
$$\delta_{21}(-4) = 0;$$

and δ_{22}:

$$\delta_{22}(0) = 1,$$
$$\delta_{22}(-4) = 0.$$

These conditions will be satisfied if

$$\delta_{11}(t) = 1 - \frac{t^2}{16},$$
$$\delta_{12}(t) = 1 + \frac{t}{2} + \frac{t^2}{16}$$

and

$$\delta_{21}(t) = \delta_{22}(t) = 1 + \frac{t}{4}.$$

The final stage in the construction of ϕ_1 comes from the requirement that

$$\nabla \phi_1(x^{(1)}) = \begin{bmatrix} 0 \\ 0 \end{bmatrix}.$$

This will mean that

$$h_{11} = \nabla \phi_1(x^{(1)}) - \nabla q_1(x^{(1)}) = \begin{bmatrix} 0 \\ 0 \end{bmatrix} - \begin{bmatrix} -3 \\ -3 \end{bmatrix} = \begin{bmatrix} 3 \\ 3 \end{bmatrix}.$$

In order to proceed, we must verify that $h_{11}^T(x^{(2)} - x^{(1)})$ is nonzero:

$$h_{11}^T(x^{(2)} - x^{(1)}) = [3 \quad 3] \begin{bmatrix} -2 \\ -2 \end{bmatrix} = -12.$$

It follows that the conditions of interpolation on γ_{11} will be

$$\gamma_{11}(0) = 0,$$
$$\gamma_{11}(-12) = 0,$$
$$\gamma'_{11}(0) = 1$$

which will be satisfied for

$$\gamma_{11}(t) = t + \frac{t^2}{12}.$$

Combining all of the above results yields:

$$\phi_1(x) = \phi_1(x_1, x_2) = -(x_1^3 x_2^3 - x_2^2 - 2x_1 x_2 + x_2 - x_1^2 + x_1 + 1)$$

and

$$\phi_2(x) = \phi_2(x_1, x_2) = -\frac{4x_2^4 - x_2 + 4x_1^4 - x_1 - 10}{4}.$$

We now tabulate the gradients and Hessians for ϕ_1 and ϕ_2 that we might need.

For $x^{(1)}$:

Function	Gradient	Hessian
ϕ_1	$\begin{bmatrix} 0 \\ 0 \end{bmatrix}$	$\begin{bmatrix} -4 & -7 \\ -7 & -4 \end{bmatrix}$
ϕ_2	$\begin{bmatrix} -\frac{15}{4} \\ -\frac{15}{4} \end{bmatrix}$	$\begin{bmatrix} -12 & 0 \\ 0 & -12 \end{bmatrix}$

For $x^{(2)}$:

Function	Gradient
ϕ_1	$\begin{bmatrix} -2 \\ -2 \end{bmatrix}$
ϕ_2	$\begin{bmatrix} \frac{17}{4} \\ \frac{17}{4} \end{bmatrix}$

To impose the conditions

$$\nabla L(x^{(1)}, \lambda^{(1)}) = 0,$$
$$\nabla L(x^{(2)}, \lambda^{(2)}) = 0,$$

and

$$\nabla^2 L(x^{(1)}, \lambda^{(1)}) = \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix},$$

we must set

$$h_{01} = \begin{bmatrix} -\frac{23}{4} \\ -\frac{7}{4} \end{bmatrix}, \qquad h_{02} = \begin{bmatrix} -\frac{17}{4} \\ -\frac{33}{4} \end{bmatrix}$$

and

$$D_{01} = \begin{bmatrix} -8 & -3 \\ -3 & -4 \end{bmatrix}.$$

Observe that

$$h_{01}^T(x^{(2)} - x^{(1)}) = \begin{bmatrix} -\frac{23}{4} & -\frac{7}{4} \end{bmatrix}\begin{bmatrix} -2 \\ -2 \end{bmatrix} = 15 \neq 0,$$

$$h_{02}^T(x^{(1)} - x^{(2)}) = \begin{bmatrix} -\frac{17}{4} & -\frac{33}{4} \end{bmatrix}\begin{bmatrix} 2 \\ 2 \end{bmatrix} = -25 \neq 0,$$

and

$$\frac{1}{2}(x^{(2)}-x^{(1)})D_{01}(x^{(2)}-x^{(1)})=\frac{1}{2}\begin{bmatrix}-2 & -2\end{bmatrix}\begin{bmatrix}-8 & -3\\ -3 & -4\end{bmatrix}\begin{bmatrix}-2\\ -2\end{bmatrix}=-36\neq 0.$$

Therefore, to specify ϕ_0 we must construct γ_{01}, γ_{02}, and Ω_{01} to satisfy:

$$\begin{aligned}
&\gamma'_{01}(0)=1, && \gamma'_{02}(0)=1,\\
&\gamma'_{01}(15)=0, && \gamma'_{02}(-25)=0,\\
&\gamma''_{01}(0)=0, && \gamma''_{02}(-25)=0,\\
&\Omega'_{01}(0)=1,\\
&\Omega'_{01}(-36)=0.
\end{aligned}$$

These conditions will be achieved with

$$\Omega_{01}(t)=t+\frac{t^2}{72},$$

$$\gamma_{02}(t)=t+\frac{t^2}{25}+\frac{t^3}{1{,}875},$$

$$\gamma_{01}(t)=t+\frac{t^3}{675}.$$

Consequently,

$$\begin{aligned}
\phi_0(x)=\phi(x_1,x_2)&\\
&=-e^{x_2^2-x_1^2}+\frac{x_2^4}{18}+\frac{x_1x_2^3}{6}-\frac{367{,}429x_2^3}{540{,}000}\\
&\quad+\frac{25x_1^2x_2^2}{72}-\frac{284{,}221x_1x_2^2}{180{,}000}+\frac{1{,}583x_2^2}{3{,}600}+\frac{x_1^3x_2}{3}\\
&\quad-\frac{100{,}543x_1^2x_2}{60{,}000}+\frac{1117x_1x_2}{1{,}800}+\frac{15x_2}{8}+\frac{2x_1^4}{9}\\
&\quad-\frac{530{,}021x_1^3}{540{,}000}-\frac{7{,}417x_1^2}{3{,}600}+\frac{23x_1}{8}-\frac{199}{24}.
\end{aligned}$$

The exercise above has a value if the behaviour of an algorithm is to be explored under specific conditions, particularly in the early stages of development: How might it perform in the neighbourhood of a degenerate optimum? What difficulties arise with zero λ's? We feel that the techniques outlined above have their main value in the flexibility with which they provide such "scenarios" for use as "scalpels" in dissecting the local behaviour of algorithms on a microscopic level. However, less elaborate test examples which are produced in a more automatic fashion are of interest, too, to test the implementation of an algorithm at a later stage of development or to use for comparing optimization systems. In view of this, we felt called upon to take a preliminary look at the feasibility of using these techniques in some automatic manner.

Since the techniques present a vast number of possibilities, any automatic generation scheme will have to be made with some restrictions. We began with the simplest subset of possibilities, that of generating problems with a single prescribed point chosen

to be an optimizer. Within this context we have experimented with problems having a special structure (nonlinear least squares) and "difficult" optima (indefinite Lagrangian Hessians with prescribed positive-definite projections).

The authors' work in progress is aimed at extending our capabilities to the automatic generation of problems with several specified points.

5. Algorithms. In order to have algorithms to motivate the automatic generation of some problems, we have prepared two experimental codes to solve the constrained nonlinear least-squares problem; i.e. problem (2.1) for the special objective function

$$\underset{x}{\text{minimize}}\ \phi_0(x) = \tfrac{1}{2}F(x)^T F(x),$$

where

$$F(x) = [f_1(x), \cdots, f_p(x)]^T.$$

It is worth noting that this structure in the objective function provides the following structure to the gradient and the Hessian:

$$\nabla\phi_0(x) = J(x)^T F(x),$$

where $J(x)$ is the Jacobian of $F(x)$ (the matrix whose ith row is $\nabla f_i(x)^T$), and

$$\nabla^2\phi_0(x) = J(x)^T J(x) + \sum_{i=1}^{p} f_i(x)\nabla^2 f_i(x).$$

We took two of the currently popular methods for nonlinear programming, the successive quadratic programming technique with the watchdog modification due to Han, Powell et al. [15], [16], [26], and the exact penalty technique due to Coleman and Conn [5], [6], and we made some modifications to these methods to take account of the special structure of the objective function. Our modifications were along the lines of those suggested by Dennis et al. in [11], extended to account for constraints. Our reasons for not experimenting with (2.1) directly were twofold: (1) by treating a problem for which there is not widely available software, we were forced to write programs from scratch, which meant that we could code both methods in a reasonably uniform manner—which we hoped would suppress artifacts arising from different coding styles and put both methods on a nearly equal footing, giving us a model situation for algorithm comparison; (2) by making changes to both methods to account for the special nature of the problem, the blame for any poor showing of one method over another could be laid to us for botching the job—we were interested only in the mechanisms of test generation and of its utility to researchers and programmers, not in making enemies by passing judgement on published methods; (3) we wished to explore the capacity of these techniques to impose structure on the problems generated—the least-squares structure was a simple enough one that it was evident to us how it could be achieved.

We will give a brief summary on both optimization methods and indicate the modifications that we made. The references above are to be consulted for further details about unmodified aspects of these methods, references [5], [6], [26] being the most pertinent.

For both methods a point x^0 and an $n \times n$ positive-definite matrix B^0 are needed to begin. The starting point is user-provided and the initial choice of B^0, dependent upon the technique being used, is some approximation to second derivative information. From then on, a sequence of points $\{x^k\}$ and a sequence of positive-definite matrices $\{B^k\}(k = 1, 2, 3, \cdots)$ is generated iteratively.

5.1. The successive quadratic programming method with watchdog. At the beginning of the kth iteration, both x^k and B^k are known. A search direction d^k is obtained by solving the quadratic program:

$$\underset{d}{\text{minimize}}\ d^T\nabla\phi_0(x^k)+\tfrac{1}{2}d^TB^kd$$

subject to

$$\phi_j(x^k)+d^T\nabla\phi_j(x^k)=0, \qquad j=1,\cdots,l,$$
$$\phi_j(x^k)+d^T\nabla\phi_j(x^k)\geqq 0, \qquad j=l+1,\cdots,l+m.$$

The solution of this quadratic program also yields Lagrange multipliers λ^{k+1}. The search direction d^k is taken with a step size α^k to yield a new point $x^{k+1}=x^k+\alpha^kd^k$, and B^{k+1} is defined by

$$B^{k+1}=J(x^{k+1})^TJ(x^{k+1})+S^{k+1}.$$

We have chosen S^{k+1} as a quasi-Newton approximation to:

$$\sum_{i=1}^{p} f_i(x^{k+1})\nabla^2 f_i(x^{k+1})-\sum_{j=1}^{l+m}\lambda_j^{k+1}\nabla^2\phi_j(x^{k+1}).$$

The choice was made as follows:

In the first iteration set $S^0\leftarrow 0_{n\times n}$.

For all other iterations find $\nabla\phi_0(x^{k+1})=J(x^{k+1})^TF(x^{k+1})$ and set

$$y^{k+1}\leftarrow[J(x^{k+1})-J(x^k)]^TF(x^{k+1})-\sum_{j=1}^{l+m}\lambda_j^{k+1}[\nabla\phi_j(x^{k+1})-\nabla\phi_j(x^k)];$$
$$s^{k+1}\leftarrow x^{k+1}-x^k.$$

Use y^{k+1} and s^{k+1} in the BFGS formula (switching to DFP if using BFGS would result in a division by zero – see, e.g. [10], for a reference to these) to update S^k to S^{k+1}.

In solving the above quadratic programming problem, the following techniques presented in [13] were used:

At each step in finding the optimal value of d a working set of constraints is determined, the gradients of which, $\phi_j(x^k)$, are taken to form the columns of a matrix A.

The columns of the matrix Z are determined to form an orthonormal basis for the null space of A^T.

A modification to the current d is found by solving a system of equations based upon the matrix Z^TB^kZ.

The equation-solving process involves using the Cholesky decomposition of

$$Z^TB^kZ=LDL^T+E, \tag{5.1.1}$$

where E is a diagonal matrix of small norm chosen to make Z^TB^kZ positive definite. (E is a zero matrix when Z^TB^kZ is already positive definite.)

After d^k is determined from the quadratic model, the step size α^k is chosen with the aid of two merit functions. The primary of these is

$$\psi_k(\alpha) = \Psi(x^k + \alpha d^k, \mu^k) = \phi_0(x) + \sum_{j \in \mathbf{J}_E} \mu_j^k |\phi_j(x^k + \alpha d^k)| + \sum_{j \in \mathbf{J}_I} \mu_j^k |\min(0, \phi_j(x^k + \alpha d^k))|,$$

where the parameters μ_j^k for each $j \in \mathbf{J}$ were updated according to:

On the first iteration:
$\mu_j^k = 2|\lambda_{\lambda_j}^{k+1}|$.
On other iterations:
If $\mu_j^{k-1} < 1.5|\lambda_j^{k+1}|$,
then $\mu_j^k = 2|\lambda_j^{k+1}|$,
else $\mu_j^k = \mu_j^{k-1}$.

The secondary merit function used is merely an approximation to the Lagrangian function:

$$L_k(x^k + \alpha d^k) = \phi_0(x^k + \alpha d^k) - \sum_{j \in \mathbf{J}} \lambda_j^{k+1} \phi_j(x^k + \alpha d^k).$$

Line searches, as needed, were performed on $\psi_k(\alpha)$ using the line search of [24] with termination set by $\eta = 0.9$. This is a very tight termination requirement that should ensure satisfaction of the line search condition given in [26], namely

$$\Psi(x^{k+1}, \mu^k) \leqq \Psi(x^k, \mu^k) - \theta[\Psi(x^k, \mu^k) - \Psi_k(x^{k+1}, \mu^k)], \tag{5.1.2}$$

where θ is a constant chosen from the interval $(0, \frac{1}{2})$. To ensure that the choice $\alpha = 1$ is ultimately always made, we test whether taking the point $x^k + d^k$ as x^{k+1} will ensure that either one of the inequalities (5.1.2) or

$$L_k(x^k + d^k) < L_k(x^k)$$

is satisfied. If so, we accept $x^k + d^k$ as x^{k+1} without carrying out a line search on $\psi_k(\alpha)$.

All other details are identical to those laid out in [26].

The area in which it might be worth probing for a weakness of our implementation of this method is in its use of estimates of Lagrange multipliers at all stages of descent. At points far from any optimizer or stationary point these multipliers may not be valid for use in setting the penalty parameters μ. We expected to use the flexibility of our techniques to try setting local conditions in which the quadratic model would yield large multipliers. This proved unnecessary, as will be seen in the final section. Problems which we generated to explore weaknesses in the second optimization method, to be covered next, also proved to cause a difficulty for the successive quadratic programming method as well.

5.2. The exact penalty method. A single merit function is used by this method:

$$\Psi_\varepsilon(x) = \mu\phi_0(x) - \sum_{j \in \mathbf{J}_{V_1}^\varepsilon(x)} \phi_j(x) + \sum_{j \in \mathbf{J}_{V_2}^\varepsilon(x)} \operatorname{sgn}(\phi_j(x))\phi_j(x),$$

where for any x and $\varepsilon > 0$ we define

$\mathbf{J}_{V_1}^\varepsilon(x) = \{j: \phi_j(x) < -\varepsilon$ and $j = l+1, \cdots, l+m\}$ the set of ε-violated inequality constraints,

$\mathbf{J}_{V_2}^\varepsilon(x) = \{j: |\phi_j(x)| > \varepsilon$ and $j = 1, \cdots, l\}$ the set of ε-violated equality constraints.

Implicitly associated with these index sets there is an approximate Lagrangian, which we will denote by

$$L_\varepsilon(x, \lambda) = \Psi_\varepsilon(x) - \sum_{j \in \mathbf{J}_A^\varepsilon(x)} \lambda_j \phi_j(x),$$

where $\mathbf{J}_A^\varepsilon(x)$ complements the index sets $\mathbf{J}_{V_1}^\varepsilon(x)$ and $\mathbf{J}_{V_2}^\varepsilon(x)$ as follows:

$\mathbf{J}_{A_1}^\varepsilon(x) = \{j: |\phi_j(x)| \leqq \varepsilon \text{ and } j = l+1, \cdots, l+m\}$ the set of ε-active inequality constraints,

$\mathbf{J}_{A_2}^\varepsilon(x) = \{j: |\phi_j(x)| \leqq \varepsilon \text{ and } j = 1, \cdots, l\}$ the set of ε-active equality constraints,

$\mathbf{J}_A^\varepsilon(x) = \mathbf{J}_{A_1}^\varepsilon(x) \cup \mathbf{J}_{A_2}^\varepsilon(x)$.

The quantity ε is a parameter of the method that is set positive initially and is (possibly) revised downward at certain times during the course of the algorithm. Similarly μ is set initially to 1.0, and it may be revised downward.

We started with x^0 and

$$B^0 = J(x^0)^T J(x^0).$$

We divided the technique for determining B^k and x^k into a global, a dropping, an intermediate, and a local mode, each characterized by the use of different quadratic subproblems and/or a different selection of step directions.

The form of the quadratic subproblem is always given by

$$\begin{aligned} &\underset{h}{\text{minimize}}\ \nabla\Psi_\varepsilon(x^k)^T h + \tfrac{1}{2} h^T B^k h, \\ &\text{subject to } \nabla\phi_j^T(x^k) h = 0, \qquad j \in \mathbf{J}_A^\varepsilon(x^k), \end{aligned} \tag{5.2.1}$$

but B^k approximates either $\nabla^2\Psi_\varepsilon(x^k)$ or else $\nabla^2 L_\varepsilon(x^k, \lambda^k)$, depending upon which mode is currently in force. The modes are determined by letting

$$A^k \equiv [\cdots, \nabla\phi_j(x^k), \cdots] \quad \text{for } j \in \mathbf{J}_A^\varepsilon(x^k),$$

by selecting vectors z_l to form an orthonormal basis for the null space of A^{k^T}, by constructing the matrix Z^k with columns z_l, and then by inspecting the norm

$$\|Z^{k^T} \nabla\Psi_\varepsilon(x^k)\|_2. \tag{5.2.2}$$

Our code was based upon the version of the exact penalty algorithm given in [3], and our interpretation of that version is distinguished from the publication version in one respect: by the presence of the intermediate mode, which serves to provide a more cautious transition from the global algorithm of [5] and its asymptotic variant, discussed in [6].

Global mode. B^k is taken to approximate $\nabla^2\Psi_\varepsilon(x^k)$ for this mode—its updating will be discussed later. Whenever (5.2.2) is greater than or equal to a tolerance τ_1, then the global mode is used, and a direction h^k is found by solving (5.2.1). A new point

$$x^{k+1} = x^k + \alpha^k h^k$$

is found by using the line search of [24] with $\eta = 0.9$ on

$$\psi_\varepsilon(\alpha) = \Psi_\varepsilon(x^k + \alpha^k h^k).$$

Dropping mode. If (5.2.2) is less than τ_1, then

Lagrange multiplier estimates λ^k are obtained by solving

$$\underset{\lambda}{\text{minimize}}\ \|A^k\lambda - \nabla\Psi_\varepsilon(x^k)\|_2, \tag{5.2.3}$$

and a test is made to see whether any $\lambda_{j_k}^k$ exists satisfying

$$\begin{aligned} &|\lambda_{j_k}^k| \notin [0, +1], \qquad j_k \in \mathbf{J}_{A_1}^{\varepsilon}(x^k), \quad \text{or} \\ &|\lambda_{j_k}^k| \notin [-1, +1], \qquad j_k \in \mathbf{J}_{A_2}^{\varepsilon}(x^k). \end{aligned} \tag{5.2.4}$$

In this event, h^k is found by solving the underdetermined linear equations

$$\begin{aligned} &\nabla \phi_{j_k}^T h = \pm 1, \\ &\nabla \phi_j^T h = 0, \qquad j \in \mathbf{J}_A^k(x^k) \backslash \{j_k\}, \end{aligned} \tag{5.2.5}$$

the sign being chosen to provide descent for Ψ_ε. A tolerance δ is used to accept or reject h^k as providing sufficient local descent:

$$[\nabla \Psi_\varepsilon(x^k) + \theta_k \nabla \phi_{j_k}(x^k)]^T h^k < \delta,$$

where $\theta_k \in \{-1, +1, 0\}$ depending upon the membership of j_k in $\mathbf{J}_E$ or $\mathbf{J}_I$ and upon the sign chosen in (5.2.5).

If no λ satisfies (5.2.4), then the intermediate mode of the method is initiated. Otherwise a new point

$$x^{k+1} = x^k + \alpha^k h^k$$

is found as using the line search. If h^k does not pass the test of sufficient decrease, however, then ε, τ_1, and another tolerance τ_2 (used below) are reduced, the index sets $\mathbf{J}$ are reassessed, and the global mode is put in force.

Intermediate mode. In the intermediate mode B^k is still taken to approximate

$$\nabla^2 \Psi_\varepsilon(x^k),$$

and h^k is determined by (5.2.1). (5.2.3) is solved at the beginning of each iteration of the mode to provide values of λ_j^k, and the conditions (5.2.4) are checked. To remain in the intermediate mode, these conditions must fail to hold. (If any $\lambda_{j_k}^k$ can be found to satisfy (5.2.4), then the dropping mode is put into force.) Assuming that the intermediate mode is permitted to continue, h^k is determined from (5.2.1), and a line search is made on $\psi_\varepsilon(\alpha) = \Psi_\varepsilon(x^k + \alpha h^k)$. This yields the point $x^k + \alpha^k h^k$. The vector

$$\Phi(x^k + \alpha^k h^k) = [\cdots, \phi_j(x^k + \alpha^k h^k), \cdots]^T, \qquad j \in \mathbf{J}_A^{\varepsilon}(x^k)$$

is formed; a direction v^k is found by solving

$$A^{k^T} v = -\Phi(x^k + \alpha^k h^k), \tag{5.2.6}$$

and the point

$$x^k + \alpha^k h^k + v^k$$

is considered. If a tolerance test (see [5], [6]) indicates that

$$\Psi_\varepsilon(x^k + \alpha^k h^k + v^k) \text{ is sufficiently smaller than } \Psi_\varepsilon(x^k),$$

then

$$x^{k+1} = x^k + \alpha^k h^k + v^k;$$

otherwise

$$x^{k+1} = x^k + \alpha^k h^k;$$

$$\varepsilon, \tau_1, \text{ and } \tau_2$$

are reduced, and the global mode is put in force.

Local mode. If the norm (5.2.2) is less than a tolerance $\tau_2 (\tau_2 < \tau_1)$, and if $x^k + \alpha^k h^k + v^k$ has been accepted as x^{k+1} on the intermediate-mode iteration step last taken, then the local mode will be in force. In this mode B^k is maintained as a quasi-Newton approximation to

$$\nabla^2 L_\varepsilon (x^k, \lambda^k),$$

and a direction h^k is determined from (5.2.1). The quantity $\alpha = \alpha^k$ is set to 1 without a line search, and the vector v^k is determined from (5.2.6) with this version of B^k. The point

$$x^{k+1} = x^k + h^k + v^k$$

is accepted, provided a tolerance test indicates that $\Psi_\varepsilon (x^k + h^k + v^k)$ shows sufficient decrease over $\Psi_\varepsilon (x^k)$. If this point is not accepted, then the intermediate mode is put into force after

$$\varepsilon, \tau_1, \text{and } \tau_2$$

have been reduced.

Updating. The quasi-Newton updating for

$$B^k = \mu J(x^k)^T J(x^k) + S^k$$

must take account of the fact that

$$S^k \approx \nabla^2 \Psi_\varepsilon (x^k) - \mu J(x^k)^T J(x^k)$$

in the global and intermediate mode, and that

$$S^k \approx \nabla^2 L_\varepsilon (x^k, \lambda^k) - \mu J(x^k)^T (x^k)$$

in the local mode. This is attempted by setting

$$\begin{aligned} y^{k+1} \leftarrow \mu [J(x^{k+1}) - J(x^k)]^T F(x^{k+1}) \\ - \sum_{j \in \mathbf{J}^\varepsilon_{V_1}(x^{k+1})} [\nabla \phi_j (x^{k+1}) - \nabla \phi_j (x^k)] \\ + \sum_{j \in \mathbf{J}^\varepsilon_{V_2}(x^{k+1})} \operatorname{sgn} (\phi_j (x^{k+1})) [\nabla \phi_j (x^{k+1})) - \nabla \phi_j (x^k)] \\ + \textbf{if } \{\text{local mode}\} \textbf{ then } \sum_{j \in \mathbf{J}^\varepsilon_{A_1}(x^{k+1})} \lambda_j^{k+1} [\nabla \phi_j (x^{k+1}) - \nabla \phi_j (x^k)] \\ \textbf{else } 0, \end{aligned} \tag{5.2.7}$$

and

$$s^{k+1} \leftarrow x^{k+1} - x^k.$$

The vectors y^{k+1} and s^{k+1} are used in the BFGS formula to update S^k to S^{k+1} (switching to DFP if division by zero is to be avoided).

In any event, the solution of (5.2.1), for B^k obtained in this fashion, is accomplished in all modes (global, intermediate, and local) via the LDL^T decomposition described in (5.1.1). It is important to note in this context that we were maintaining and updating the full B matrix, and then projecting it. This is *not* what is done in [5], [6]. There the authors advocate the use of an approximation *only* to the matrix $Z^T BZ$. The structure of our partial Hessian updates above, we felt at the time, mitigated against dealing easily with an approximation to the projected Hessian alone.

The decision was made to investigate the sensitivity of our interpretation of this method to indefinite Hessians. The next section describes a number of problems which were generated for least-squares problems with such Hessians.

6. Test results. The nonlinear programs to be considered here have the form of nonlinear least-squares problems:

$$\underset{x}{\text{minimize}}\ \tfrac{1}{2}F(x)^TF(x)+\Omega_0(\tfrac{1}{2}(x-x^*)^TD(x-x^*))+\gamma_0((x-x^*)^Th_0)$$

subject to

$$\phi_j(x)=q_j(x)+h_j^T(x-x^*)+\alpha_j=0, \qquad j=1,\cdots,l,$$

$$\phi_j(x)=q_j(x)+h_j^T(x-x^*)+\alpha_j\geqq 0, \qquad j=l+1,\cdots,l+m$$

where

$$F(x)=[f_1(x),\cdots,f_p(x)]^T$$

with

$$f_i(x)=\sum_{k=1}^{n}(x_k^2x_i)-i, \qquad i=1,\cdots,p,$$

where

$$q_j(x)=\sum_{k=1}^{n}(x_k^2x_j)-j, \qquad j=1,\cdots,l+m,$$

and where

$$\Omega_0(t)=\gamma_0(t)-\tfrac{1}{2}(t+1)^2 \quad \text{for } t\in\mathbf{R}.$$

Note that this is consistent with the choice

$$\alpha_0\equiv 0,$$

$$\gamma_j\equiv \text{identity} \quad \text{for } j>0,$$

$$\delta_j\equiv 1,$$

$$\Omega_j\equiv 0 \quad \text{for } j>0.$$

We have arranged the form of Ω_0 and γ_0 so that the objective function will be a sum of squares.

Test generator. A simple test-generating system (available from the authors) was written for the above. In all cases it assigned values to the optimizer and its associated Lagrange multipliers by uniformly distributed pseudo-random numbers. The components of x^* and the Lagrange multipliers λ^* of the active equality constraints were chosen between -1.0 and 1.0, while the Lagrange multipliers of the active inequality constraints were chosen between 0.0 and 1.0.

The Lagrangian Hessian was set to be

$$\nabla^2L=\begin{bmatrix}\sigma U_1^TU_1 & 0\\ 0 & U_2^TU_2\end{bmatrix},$$

where U_1 was a $(\nu+l)\times(\nu+l)$ upper triangular matrix.

Except for $U_1[1,1]$, which was set to 2, all elements of U_1 were generated by using a random number generator and were in the range -1.0 to 1.0. U_2 was also an upper triangular matrix with elements all generated between -1.0 and 1.0.

The scalar σ was under user control, and it dictated whether the generated problem would be definite ($\sigma > 0$) or indefinite ($\sigma \leqq 0$). A sequence of problems could be generated in which all quantities (x^* values, Lagrange multipliers, U_1, etc.) were exactly the same, and only $\nabla^2 L$ differed by having a different value of σ.

A feature for attempting degenerate optimizers was built in. Normally the gradient $\phi_j(x^*)$ was set to the jth column of the identity. A user-supplied option permitted the last several $\nabla\phi$'s in sequence to be constructed as random linear combinations of the initial $\nabla\phi$'s.

The constants α_j were set so that $\phi_j(x^*) = 0$ for the active constraints, and $\phi_j(x^*) = 1$ for the inactive constraints.

Problems. Five problems will be reported upon, the first three with $\sigma = 1, -1$, and -10, respectively, and the last two with $\sigma = 1$ and -1. For the first three values of σ, each problem was constructed to be nondegenerate at the optimizer, and for the remaining two values of σ each problem was made to be degenerate. To obtain degenerate problems, the gradient of the fourth active constraint was chosen to be linearly dependent on the gradients of the first three active constraints. The point x^* was verified computationally to be a local optimizer for all problems.

For the problems reported upon below, we had

$n = 5$—number of variables
$p = 5$—number of components in $F(x)$
$l = 2$—number of equality constraints
$m = 3$—number of inequality constraints
$\nu = 2$—number of active inequality constraints

$$\text{Optimizer } x^* = \begin{pmatrix} 0.30467 \\ 0.62076 \\ -0.80999 \\ 0.41441 \\ 0.95425 \end{pmatrix}.$$

Active constraints set $= \{1, 2, 3, 4\}$.

$$\text{Lagrange multipliers } \lambda^* = \begin{pmatrix} 0.12629 \\ 0.59378 \\ 0.31597 \\ 0.58508 \end{pmatrix}.$$

Implementations. All codes, those for the test generation as well as those for the optimization methods, were written in ANSII FORTRAN, as verified by the PFORT verifier [28], and were run on the Honeywell 66/60 of the Mathematics Faculty Computing Facility at the University of Waterloo. The basic linear algebra computations (dot products, etc.) were carried out using the Basic Linear Algebra Subroutines of [19], and the more advanced linear algebra (e.g. forming QR factorizations and finding the LDL^T factorization of a—possibly nonnegative-definite—symmetric matrix) were carried out by using or modifying programs from the NPL optimization library [14]. The quadratic programming code needed for the method of [26] was also obtained from this library, which was made available through the courtesy of P. E. Gill and W. Murray. The line search routine that was used in all codes comes from reference [24] and was kindly provided by Michael Overton. All random numbers were produced by [33] as modified for double precision.

Employment. We illustrate the use of these test problems in two typical ways, firstly in the "shotgun" mode of software testing, in which the problems are given to the codes and the results are summarized in tabular or statistical format, secondly in a "scalpel" mode, in which selected problems are followed, step-by-step, through an algorithm to see what insights can be gained. For the former mode, the computational results are given in tables that follow. Unless otherwise indicated, all tests were terminated if more than 50 iterations were taken, in the interests of saving our computing budget. For the latter mode, we include summary remarks after each table indicating a few of the interesting features of the execution. Again, we remind the reader that our remarks reflect studies of the methods of [5], [6], [26] as subjected to our interpretation, for a problem of special structure, applying quasi-Newton updates of our choice, and suffering under our blind spots in rendering these methods into computer code. It is the purpose of the test examples to indicate where we might have fallen down on our task of implementing the methods.

Table headings:
EP—exact penalty method
SQP—successive quadratic programming method with watchdog
NI—number of iterations
SP—starting point:
  starting point 1—x with all component equal to 1.0
  starting point 2—x with all component equal to 10.0
  starting point 3—x with all component equal to 100.0.
FE—number of objective function evaluations
  (each evaluation of the objective function was counted as $p+2$)
CE—number of constraint evaluations
GFE—number of gradient evaluations in the objective function
  (each Jacobian evaluation was counted as $p+2$)
GCE—number of gradient evaluations in the constraints
R—result:
  C—Constructed optimizer found
  A—Another point of termination
  F—Failure, code aborted
  M—Maximum iteration count reached

Tables:

TABLE 1
Nondegeneracy with positive-definite H ($\sigma = 1.0$).

	EP						SQP					
SP	NI	FE	CE	GFE	GCE	R	NI	FE	CE	GFE	GCE	R
1	22	245	175	210	169	C	19	147	105	105	103	C
2	35	490	350	435	375	C	10	98	70	70	65	F
3	38	497	355	420	448	C	47	434	310	310	298	C

Remarks:
(1) In SQP with starting point 1 the first 4 penalty parameters were determined by the Lagrange multipliers of the first quadratic subproblem, and the last penalty

parameter was determined by a multiplier of the second quadratic subproblem. The penalty parameters ranged from 51 to 350 and remained unchanged throughout iterations 3-19.

(2) In SQP with starting point 2, the penalty parameters were set between 16,750 and 89,016 by the first quadratic subproblem. At iteration 6, a quadratic-programming subproblem was encountered with Lagrange multipliers of the order of 10^9. This forced the penalty parameters to become large as well. On the next two iterations, multipliers became 10^{23} and 10^{36} respectively. The line search became unable to find lower merit function values with the search directions produced, and the code was aborted.

(3) In SQP, with starting point 3, the penalty parameters had been set to numbers of magnitude 10^6 by iteration 3. These values held throughout the subsequent iterations without change.

(4) It is a general observation that the first few quadratic programs encountered by SQP for all of our starting points and for all of our generated problems were associated with large Lagrange multipliers. These, in turn, set the values of the penalty parameters quite large, and subsequent quadratic programs, with their much more reasonably sized multipliers, had no influence on the magnitude of the parameters.

(5) Each of the three executions of the EP method was characterized by an initial number of steps in which all constraints were deemed active, causing (very large) Lagrange multipliers to be computed. These generated "dropping directions", (5.2.5). There followed an intermediate number of global-mode iterations in which steps were obtained from (5.2.1), during which $\|Z^T\nabla\Psi_\varepsilon\|$ decreased almost monotonically. These steps gathered the activities that represented the optimal constraint manifold, and ended when the intermediate mode was entered. After several intermediate mode iterations, the local mode was begun. Although the theory of [6] only guarantees 2-step superlinear convergence, convergence to the constructed optimizer appeared to be 1-step superlinear. For starting point 2, for example, the breakdown was: 7 iterations of dropping steps, 16 of global steps, 3 intermediate steps, and 9 local steps. The components of x^k agreed with those at termination in the final 5 steps respectively to 3, 4, 7, 12, and >15 digits.

TABLE 2
Nondegeneracy with mildly indefinite $H(\sigma = -1.0)$.

	EP						SQP					
SP	NI	FE	CE	GFE	GCE	R	NI	FE	CE	GFE	GCE	R
1	23	252	180	205	159	C	22	182	130	130	126	C
2	32	455	325	385	334	C	10	98	70	70	65	F
3	44	518	370	395	347	A	50	672	480	480	464	M

Remarks:

(1) The SQP method failed in the same manner as for starting point 2 in the previous problem—quadratic programs were encountered with large Lagrange multipliers.

(2) For starting point 3, the EP method terminated on a point (not the constructed optimizer) that satisfied the first-order necessary conditions. The code does not check second-order conditions.

TABLE 3
Nondegeneracy with strongly indefinite $H(\sigma = -10.0)$.

	EP						SQP					
SP	NI	FE	CE	GFE	GCE	R	NI	FE	CE	GFE	GCE	R
1	50	1,015	830	675	634	A	19	140	100	100	99	C
2	50	868	625	715	646	M	10	98	70	70	65	F
3	50	798	590	645	577	M	33	238	170	170	169	C

Remarks:

(1) The EP code arrived at a stationary point (a first-order point that was infeasible for the given constraints) on iteration 30. Progress to this stage involved an initial mixture of global and dropping steps, an attempted intermediate step followed by several successful intermediate steps, and a final number of local steps. The stationary point had constraints 2 and 4 active (1, 2, 3, and 4 are active at the constructed optimizer), and was infeasible in constraint 1. The penalty parameter μ was reduced to an eighth of its value, and the remaining 20 iterations brought convergence to a point that satisfied the first-order conditions. This point was [0.299, 0.610, −0.740, 0.407, 0.87], which is close to the constructed optimizer, but it has only constraints 1, 2, and 4 active (values $\approx 10^{-20}$). Constraint 3, which is zero at the constructed optimizer, has the value 0.0401 at this point. There is a good chance that the projected Hessian of the Lagrangian—by the nature of the Hessian's construction, by the proximity of this point to the constructed optimizer, and by the fact that the gradient of constraint 3 is not included in the projection—is indefinite. Our code, which maintains a full approximation to $\nabla^2 L$ rather than an approximation to $Z^T \nabla^2 L Z$ appeared sensitive to indefiniteness in the Lagrangian Hessian. This was further emphasized by poor performance on the problem of Table 5.

(2) The EP method behaved much the same way on starting points 2 and 3, but lagged enough behind its progress on starting point 1 that it was cut off at iteration 50 just on or after a reduction in μ.

(3) SQP failure was as before.

TABLE 4
Degeneracy with positive-definite $H(\sigma = 1.0)$.

	EP						SQP					
SP	NI	FE	CE	GFE	GCE	R	NI	FE	CE	GFE	GCE	R
1	33	392	280	285	237	C	50	658	470	470	445	M
2	50	791	656	615	555	M	50	728	520	520	516	M
3	49	588	420	425	455	C	22	1,519	1,085	1,085	1,080	F

Remarks:

(1) With starting points 1 and 3, EP converged to the optimizer. The Lagrange multipliers at the termination point were in agreement with the first-order conditions,

but they were different in value from those assigned due to the linear dependence of the constraint gradients.

(2) For starting point 2, EP attained a stationary point, reduced μ, and was making normal progress when iteration 50 was reached.

(3) The failures of SQP were as above.

TABLE 5
Degeneracy with mildly indefinite $H(\sigma = -1.0)$.

	EP						SQP					
SP	NI	FE	CE	GFE	GCE	R	NI	FE	CE	GFE	GCE	R
1	50	623	445	485	391	M	50	1,379	985	985	926	M
2	50	1,645	1,195	1,245	1,137	M	49	1,050	750	750	746	F
3	50	623	445	565	456	M	29	315	225	225	225	C

Remarks:

(1) For starting point 1, the behaviour of the EP, which was noted above, was to be seen. A stationary point was reached, μ was reduced, and normal progress appeared to continue as iteration 50 was reached. To verify the impression that progress was normal, a second run was made with 100 iterations allowed. The method terminated at the constructed optimum after 95 iterations.

(2) For starting point 2, the EP code reached a state at which only a minute amount of change in x was made at every iteration. The code had, to all intents, stalled, and it was aborted.

(3) For starting point 3, the EP code attained a stationary point around iteration 50. It was cut off before a reduction in μ could take place.

(4) The SQP method again developed large penalty parameters and ceased making progress.

Summary. We make note of the fact that the problems were easy to generate, that the generation process was flexible enough to handle a special request on the form of the objective function (i.e. that it be a sum of squares), that problems could be generated automatically, and that useful information pointing to the areas that need further study was obtained.

Acknowledgments. The authors wish to thank A. R. Conn, who has given freely of his suggestions and encouragement. Thanks are also due to R. H. Byrd and A. L. Goldman, who undertook careful readings of the thesis from which much of this material is derived, and to W. Murray, P. E. Gill, and M. L. Overton for providing the use of certain of their software routines. Finally, the authors are grateful to J. T. Hon for generating the problems reported here, running the algorithms, and creating the tables of this section.

REFERENCES

[1] Å. BJÖRCK AND T. ELFVING (1973), *Algorithms for confluent Vandermonde systems*, Numer. Math., 21, pp. 130–137.

[2] A. CHARNES, W. M. RAIKE, J. D. STUTZ AND A. S. WALTERS (1974), *On generation of test problems for linear programming codes*, Comm. ACM, 17(10), pp. 583–586.

[3] T. F. COLEMAN (1979), *A superlinear penalty function method to solve the nonlinear programming problem*, Ph.D. dissertation, Dept. Combinatorics and Optimization, Univ. Waterloo, Waterloo, Ontario, Canada.

[4] A. R. COLEVILLE (1970), *A comparative study on nonlinear programming codes*, in Proc. the Princeton Symposium on Mathematical Programming, H. W. Kuhn, ed., Princeton Univ. Press, Princeton, NJ.

[5] A. R. CONN AND T. F. COLEMAN (1982), *Nonlinear programming via an exact penalty function: global analysis*, Mathematical Programming, 24, North-Holland, Amsterdam, pp. 137–161.

[6] ———, (1982), *Nonlinear programming via an exact penalty function: asymptotic analysis*, Mathematical Programming, 24, North-Holland, Amsterdam, pp. 123–136.

[7] H. CROWDER, R. S. DEMBO AND J. M. MULVEY (1978), *Reporting computational experiments in mathematical programming*, Mathematical Programming, 15, North-Holland, Amsterdam, pp. 316–329.

[8] ———, (1979), *On reporting computational experiments with mathematical software*, ACM Trans. Math. Software, 5(2), pp. 193–203.

[9] R. S. DEMBO (1976), *A set of geometric programming test problems and their solutions*, Mathematical Programming, 10, North-Holland, Amsterdam, pp. 192–213.

[10] J. E. DENNIS, Jr. AND J. J. MORÉ (1977), *Quasi-Newton methods: motivation and theory*, SIAM Review, 19, pp. 46–89.

[11] J. E. DENNIS, Jr., D. M. GAY AND R. E. WELSCH (1981), *An adaptive nonlinear least-squares algorithm*, ACM Trans. Math. Software, 7, pp. 248–383.

[12] A. V. FIACCO AND G. P. MCCORMICK (1968), *Nonlinear Programming: Sequential Unconstrained Minimization Techniques*, John Wiley, New York.

[13] P. E. GILL AND W. MURRAY (1978), *Numerically stable methods for quadratic programming*, Mathematical Programming, 14, North-Holland, Amsterdam, pp. 349–372.

[14] P. E. GILL, W. MURRAY AND S. PICKEN (1978), *The* NPL *Numerical Optimization Software Library*, Division of Numerical Analysis and Computer Science, National Physical Laboratory, Teddington, Middlesex, TW11 0LW, England.

[15] S-P. HAN (1976), *Superlinearly convergent variable metric algorithms for general nonlinear programming problems*, Mathematical Programming, 11, North-Holland, Amsterdam, pp. 263–282.

[16] ———, (1977), *A globally convergent method for nonlinear programming*, J. Optim. Theory Appl., 22(3), pp. 297–310.

[17] D. M. HIMMELBLAU (1972), *Applied Nonlinear Programming*, McGraw-Hill, New York.

[18] K. L. HOFFMAN AND D. R. SHIER (1980), *A test problem generator for discrete linear L1 approximation problems*, ACM Trans. Math. Software, 6, pp. 587–593.

[19] C. L. LAWSON, R. J. HANSON, D. R. KINCAID AND F. T. KROGH (1979), *Basic linear algebra subprograms for FORTRAN usage*, ACM Trans. Math. Software, 5, pp. 308–325.

[20] J. H. MAY AND R. L. SMITH (1980), *Random polytopes: their definition, generation and aggregate properties*, 80–6, Dept. Industrial and Operations Engineering, Univ. Michigan, Ann Arbor.

[21] W. M. MICHAELS AND R. O'NEILL (1980), *A mathematical program generator* MPGENR, ACM Trans. Mathematical Software, 6, pp. 31–44.

[22] J. J. MORÉ (1978), *Implementation and testing of optimization software*, IFIP WG 2.5, Working Conference on Performance Evaluation of Numerical Software, December, Baden, Austria.

[23] J. J. MORÉ, B. S. GARBOW AND K. E. HILLSTROM (1981), *Algorithm* 566*:* FORTRAN *subroutines for testing unconstrained optimization software*, ACM Trans. Mathematical Software, 7, pp. 136–140.

[24] W. MURRAY AND M. L. OVERTON (1978), *Steplength algorithms for minimizing a class of nondifferentiable functions*, STAN-CS-78-679, Computer Science Dept, Stanford Univ., Stanford, CA, November.

[25] W. MURRAY AND M. H. WRIGHT (1978), *Projected Lagrangian methods based on the trajectories of penalty and barrier functions*, SOL 78-23, Operations Research Dept., Systems Optimization Laboratory, Stanford Univ., Stanford, CA.

[26] M. J. D. POWELL, R. M. CHAMBERLAIN, C. LEMARECHAL AND H. C. PEDERSEN (1979), *The watchdog technique for forcing convergence in algorithms for constrained optimization*, Tenth International Symposium on Mathematical Programming, Montreal, Canada, August.

[27] J. B. ROSEN AND S. SUZUKI (1965), *Construction of nonlinear programming test problems*, Comm. ACM, 8(2), p. 113.

[28] B. G. RYDER (1974), *The* PFORT *verifier*, Software Practice and Experience, 4, pp. 359–377.

[29] R. W. H. SARGENT (1982), *Recursive quadratic programming algorithms and their convergence properties*, Numerical Analysis, Proceedings, Cocoyoc, Mexico 1981, J. P. Hennert, ed., Lecture Notes Mathematics, 909, Springer-Verlag, Berlin.

[30] K. SCHITTKOWSKI (1979), *Randomly generated nonlinear programming test problems*, 9th IFIP Conference on Optimization Techniques, Warszawa.

[31] ———, (1980), *A numerical comparison of* 13 *nonlinear programming codes with randomly generated test problems*, in Numerical Optimization of Dynamic Systems, L. C. W. Dixon and G. P. Szego, eds., North-Holland, Amsterdam.

[32] K. SCHITTKOWSKI AND W. HOCK (1981), *Test examples for nonlinear programming codes*, Lecture Notes in Economic and Mathematical Systems 187, Springer-Verlag, Berlin.

[33] L. SCHRAGE (1979), *A more portable* FORTRAN *random number generator*, ACM Trans. Math. Software, 5, pp. 132–138.

[34] G. W. STEWART, (1981), *Constrained definite Hessians tend to be well conditioned*, Mathematical Programming, 21, North-Holland, Amsterdam, pp. 235–238.

[35] J. TELGEN AND W. B. VAN DAM (1979), *Randomly generated polytopes for testing mathematical programming algorithms*, 7929/0, Economics Institut, Erasmus University, Holland.

SIAM J. SCI. STAT. COMPUT.
Vol. 7, No. 3, July 1986

006

A COMPARISON BETWEEN SOME DIRECT AND ITERATIVE METHODS FOR CERTAIN LARGE SCALE GEODETIC LEAST SQUARES PROBLEMS*

G. H. GOLUB†, P. MANNEBACK‡ AND PH. L. TOINT‡

Abstract. The purpose of this paper is to describe and compare some numerical methods for solving large dimensional linear least squares problems that arise in geodesy and, more specially, from Doppler positioning. The methods that are considered are the direct orthogonal decomposition, and the combination of conjugate gradient type algorithms with projections as well as the exploitation of "Property A". Numerical results are given and the respective advantage of the methods are discussed with respect to such parameters as *CPU* time, input/output and storage requirements. Extensions of the results to more general problems are also discussed.

Key words. Doppler positioning, geodesy, orthogonal transformations, preconditioned conjugate gradient, sparse linear least squares

1. Introduction. In recent years, the numerical solution of high-dimensional overdetermined systems of linear equation has received much attention and development. There are multiple reasons for this continuing interest but one can certainly single out the strong influence of very large least squares problems arising from geodesy. Since its origin, this branch of applied mathematics has promoted research in the area of numerical analysis and many of the recent papers on the subject have been motivated by it [7], [11]. This paper will pursue the same line and will discuss several numerical methods that are relevant to the solution of a typical geodetic problem, Doppler positioning. The authors nevertheless feel that many of the conclusions that will be drawn in this context could also be of interest in other fields where large block-structured, least squares problems do arise, such as in statistical computations and earthquake predictions.

Doppler positioning is a geodetic technique that allows the determination of the relative positions of several ground stations from the observations of Doppler shifts on radio frequencies that are broadcast by artificial satellites. Each station observes several satellite passes over the horizon, and then transmit the data to a central computing center, where they are processed in order to compute corrections to the relative position of the stations, but also to some orbital parameters of the broadcasting satellites themselves. A more complete description of this problem may be found in [14], where a method based on block orthogonalization decomposition was proposed for reducing both the data transmission and the overall computing time.

This paper will focus on the solution of a linear least squares system that can be interpreted as a first reduction of the general Doppler positioning problem. More precisely, one wishes to compare in this context the respective merits of a direct *QR* type and the iterative conjugate gradient type methods.

In § 2, we will present the problem more formally, and in § 3 we describe the algorithms that are considered. Numerical results are displayed and discussed in § 4.

2. The least squares problem. As mentioned in the introduction, we are concerned with the solution of a linear least squares problem, viz. we would like to find the vector x that achieves

$$\min_x \|A\mathbf{x}-\mathbf{b}\|_2, \tag{1}$$

* Received by the editors November 1, 1984.

† Department of Computer Science, Stanford University, Stanford, California 94305.The work of this author was sponsored by U.S. Army contract DAAG 29-83-K-0124.

‡ Department of Applied Mathematics, Facultés Universitaires de Namur, Namur, Belgium.

where A is a large sparse matrix and b is a given vector. We assume the rank of A is equal to the number of columns of A.

We are especially interested in the case where the matrix A has a given block structure defined by

$$A = \begin{pmatrix} B_1 & & & C_1 \\ & B_2 & & C_2 \\ & & \ddots & \vdots \\ & & B_m & C_m \end{pmatrix} \equiv (B, C) \tag{2}$$

where

$$B_i = \begin{pmatrix} B_{i1} \\ B_{i2} \\ \vdots \\ B_{in} \end{pmatrix} \tag{3}$$

and

$$C_i = \begin{pmatrix} C_{i1} & & & \\ & C_{i2} & & \\ & & \ddots & \\ & & & C_{in} \end{pmatrix}. \tag{4}$$

In this description, the blocks B_{ij} and C_{ij} $(i = 1, \cdots, m;\ j = 1, \cdots, n)$ are dense and of dimension $(k+p) \times k$ and $(k+p) \times p$, respectively, with $k \geqq p$. The overall system (1) has therefore $mk + np$ unknowns and $mn(k+p)$ equations. Moreover, we will assume that each matrix (B_{ij}, C_{ij}) $(i = 1, \cdots, m;\ j = 1, \cdots, n)$ is $(k+p) \times (k+p)$ upper triangular, dense and nonsingular. This last assumption appears naturally in the Doppler positioning problem described in [14]. We will nevertheless consider more general structures in the final section. We exemplify the structure of A for $m = 3$ and $n = 2$ in Fig. 1.

(5) $A =$

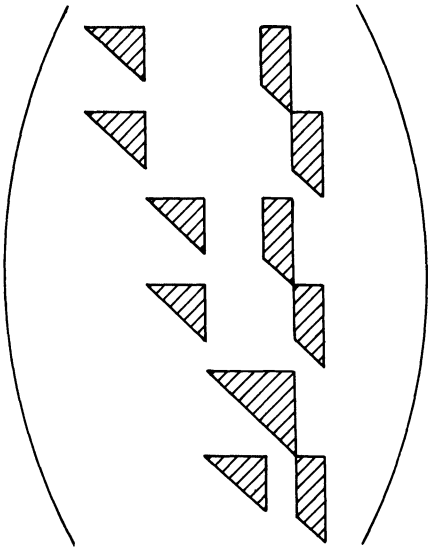


FIG. 1. *Example of structure matrix A.*

In the context of our geodetic application, m is the number of satellite passes that have been observed, n is the number of ground stations, k is the number of orbital parameters to be estimated for one satellite and p is the number of unknown coordinates of a ground station. In the European network that will be discussed below, these dimensions are

$$m = 152, \quad n = 34, \quad k = 6, \quad p = 3. \tag{6}$$

In practical cases, some block lines of (5) may be missing, indicating the fact that all stations do not necessarily observe all satellites. This creates some irregularity in the pattern, that can be exploited as we will see below. The system resulting from the European network we have treated contains 2,317 blocks, with a total number of 20,853 equations and 1,014 unknowns. It is based on the second European Doppler Observation Campaign (EDOC-2) [3].

Each of these equations has less than 9 nonzero entries. As suggested by these numbers, we consider rather large and very sparse systems.

Assume now, for the sake of exposition, that the normal equations

$$A^T A\mathbf{x} = A^T\mathbf{b} \tag{7}$$

and formed, and the Cholesky factor R of A^TA is computed. It can be readily seen that significant fill-in will occur in the matrix A^TA and in the resulting upper triangular factor R. In fact, one can check that R has the form

$$R = \begin{pmatrix} R(B) & R_1 \\ 0 & R_2 \end{pmatrix} \tag{8}$$

with

$$R(B) = \begin{pmatrix} R(B_1) & & 0 \\ & \ddots & \\ 0 & & R(B_m) \end{pmatrix} \tag{9}$$

where we have denoted by $R(X)$ the upper triangular Cholesky factor of the matrix X^TX.

In addition, R_1 is $mk \times np$ dense, while R_2 is $np \times np$ upper triangular.

Since most of the practical problems we have in mind are moderately ill conditioned, and maximal accuracy is often required, we will not use the normal equations approach (7). Instead we will work directly on the matrix A itself. The three methods we have considered are described in the next section.

3. Description of the algorithms.

3.1. A direct orthogonalization method. Following a well established theory ([12] for example), we may consider the orthogonal decomposition

$$A = Q\begin{pmatrix} R \\ 0 \end{pmatrix} \equiv Q_A R \tag{10}$$

where Q is orthogonal and R upper triangular, Q_A is the first n columns of Q. This factorization can be computed by using elementary orthogonal transformations (Householder transformations or Givens rotations) and problem (1) is then reduced to

$$\min_x \|R\mathbf{x} - Q_A^T\mathbf{b}\|_2 \tag{11}$$

whose solution can be obtained by a simple backward substitution. Furthermore, the structure of A and R can be exploited to reduce the computational effort in forming

R, as discussed in [7], [11], and [14]. In particular, the last paper advocates the use of a specific elimination procedure that will be called Algorithm D (for direct). Before describing it, let us define

$$C_{ij} = \begin{pmatrix} C_{ij}^{(a)} \\ C_{ij}^{(b)} \end{pmatrix} \qquad (i = 1, \cdots, m; j = 1, \cdots, n) \tag{12}$$

where $C_{ij}^{(b)}$ is $p \times p$ upper triangular and $C_{ij}^{(a)}$ is $k \times p$, and consider the part of the right-hand side b corresponding to the block row of C_{ij} and partition it according to (12) in

$$\mathbf{b}_{ij} = \begin{pmatrix} \mathbf{b}_{ij}^{(a)} \\ \mathbf{b}_{ij}^{(b)} \end{pmatrix} \qquad (i = 1, \cdots, m; j = 1, \cdots, n). \tag{13}$$

Now partition the vector of variables as

$$\mathbf{x}^T = (\mathbf{x}_1^{(1)}, \cdots, \mathbf{x}_n^{(1)}, \mathbf{x}^{(2)})^T \tag{14}$$

where $\mathbf{x}_i^{(1)}$ has k components ($i = 1, \cdots, m$) and $\mathbf{x}^{(2)}$ np components. Finally observe that each B_{ij} is of the form

$$B_{ij} = \begin{pmatrix} B_{ij}^{(a)} \\ 0 \end{pmatrix} \tag{15}$$

where $B_{ij}^{(a)}$ is $k \times k$ upper triangular, and define

$$B_j^{(a)} = \begin{pmatrix} B_{i1}^{(a)} \\ \vdots \\ B_{in}^{(a)} \end{pmatrix} \tag{16}$$

where $B_i^{(a)}$ now has nk rows.

ALGORITHM D.

Step 1: For $i = 1, \cdots, m$, compute the matrix $R(B_i)$ by applying Householder transformations to triangularize the matrix $B_i^{(a)}$, i.e.,

$$Q_i^T B_i^{(a)} = \begin{pmatrix} R(B_i) \\ 0 \end{pmatrix} \tag{17}$$

where Q_i^T is the product of the transformations. Apply these transformations also to the corresponding rows of C and of $\mathbf{b}$ to obtain

$$Q_i^T \begin{pmatrix} C_{i1}^{(a)} & & 0 \\ & \ddots & \\ 0 & & C_{in}^{(a)} \end{pmatrix} \equiv \begin{pmatrix} C_i^{(c)} \\ C_i^{(d)} \end{pmatrix} \tag{18}$$

and

$$Q_i^T \begin{pmatrix} \mathbf{b}_{i1}^{(a)} \\ \vdots \\ \mathbf{b}_{in}^{(a)} \end{pmatrix} \equiv \begin{pmatrix} \mathbf{b}_i^{(c)} \\ \mathbf{b}_i^{(d)} \end{pmatrix}. \tag{19}$$

Step 2: Use Givens rotations, sequentially, row by row, to obtain the $np \times np$ upper triangular matrix R_2 in the system

$$Q^T \begin{pmatrix} C_{11}^{(b)} & & & & \\ & \ddots & & & \\ & & & C_{1n}^{(b)} & \\ C_{m1}^{(b)} & & & & \\ & \ddots & & & \\ & & & C_{mn}^{(b)} & \\ & & C_1^{(d)} & & \\ & & \vdots & & \\ & & C_m^{(d)} & & \end{pmatrix} = \begin{pmatrix} R_2 \\ 0 \end{pmatrix} \tag{20}$$

and compute also

$$Q^T(\mathbf{b}_{11}^{(b)} \cdots \mathbf{b}_{1n}^{(b)} \cdots \mathbf{b}_{m1}^{(b)} \cdots \mathbf{b}_{mn}^{(b)} \mathbf{b}_1^{(d)} \cdots \mathbf{b}_m^{(d)})^T \equiv \begin{pmatrix} \mathbf{b}^{(e)} \\ \mathbf{b}^{(f)} \end{pmatrix}. \tag{21}$$

Step 3: Solve successively the triangular systems

$$R_2 \mathbf{x}^{(2)} = \mathbf{b}^{(e)} \tag{22}$$

and

$$R(B_i)\mathbf{x}_i^{(1)} = \mathbf{b}_i^{(c)} - C_i^{(c)}\mathbf{x}^{(2)} \qquad (i = 1, \cdots, m). \tag{23}$$

Observe that, in Step 1, orthogonal transformations do create fill-in in (18). Givens rotations are used in Step 2 to avoid storing the matrix to be triangularized in memory space which would require $O(mn^2pk)$ words. Instead, only R_2 has to be stored.

Because of the fill-in in the matrices $C_i^{(d)}$ the operation count is dominated by the elimination in Step 2, which requires

$$O(mn^3p^2k) \tag{24}$$

floating point multiplications. The number of additions is comparable and therefore will not be discussed further. In core storage, the complete algorithm requires

$$O(n^2kp) \tag{25}$$

words where this cost is dominated by the fill-in occurring in (18).

3.2. A conjugate gradient type method. As we have just observed, the main costs of Algorithm D are caused by the fill-in that occurs in the computation of the Cholesky factor of A. This fill-in is clearly caused by the coupling of the two blocks B and C in (2). One therefore feels motivated for considering an iterative technique that would involve no fill-in, and, hopefully, would result in less computational effort. Amongst the well-known methods of this type, conjugate gradient methods are natural candidates [12], [15]. These procedures are implicitly or explicitly based on the bidiagonalization procedure of Golub and Kahan [9], and we will see that this procedure can be used to advantage for our particular structure.

Note however that other iterative schemes have been introduced for solving large sparse least squares systems without forming normal equations [2], [5]. One problem with iterative methods is the large number of iterations often needed. Björck has proposed in [1] to accelerate convergence of the block SSOR preconditioning by conjugate gradient methods. The iterative methods presented in this paper belong to this class of algorithms.

The outline of the algorithm that we will consider is of the LSQR type, as proposed by Paige and Saunders in [15]. It is analytically equivalent to the standard method of

conjugate gradients but is recommended for its numerical stability. It is based on the transformation of problem (1) into a sequence of problems

$$\min_{\mathbf{y}_i} \|F_i \mathbf{y}_i - \beta_1 \mathbf{e}_1\|_2 \tag{26}$$

where each F_i is a lower bidiagonal matrix of the form

$$F = \begin{pmatrix} \alpha_1 & & & \\ \beta_2 & \alpha_2 & & 0 \\ & \beta_3 & \alpha_3 & \\ & & \ddots & \alpha_i \\ & & & \beta_{i+1} \end{pmatrix}. \tag{27}$$

This matrix is obtained from the original matrix A by the following procedure ($\mathbf{x}_0$ is given.)

ALGORITHM B.

Step 1:

$$\beta_1 \mathbf{u}_1 = \mathbf{b} - A\mathbf{x}_0 \tag{28}$$

$$\alpha_1 \mathbf{v}_1 = A^T \mathbf{u}_1. \tag{29}$$

Step 2: For $i = 1, 2, \cdots$

$$\beta_{i+1} \mathbf{u}_{i+1} = A\mathbf{v}_i - \alpha_i \mathbf{u}_i \tag{30}$$

$$\alpha_{i+1} \mathbf{v}_{i+1} = A^T \mathbf{u}_{i+1} - \beta_{i+1} \mathbf{v}_i. \tag{31}$$

The nonnegative scalars α_i and β_i $(i = 1, 2, \cdots)$ are defined to enforce $\|\mathbf{u}_i\|_2 = \|\mathbf{v}_i\|_2 = 1$.

As discussed in [15], the solution $\mathbf{y}_i$ to these problems can be computed by expressions of the form

$$\left.\begin{aligned} \mathbf{d}_i &= \frac{1}{\rho_i}(\mathbf{v}_i - \theta_i \mathbf{d}_{i-1}) \\ \mathbf{y}_i &= \mathbf{y}_{i-1} + \phi_i \mathbf{d}_i \end{aligned}\right\} \quad (i = 1, 2, \cdots) \tag{32}$$

where $\mathbf{d}_0$ and $\mathbf{y}_0$ are defined to be zero, and where ρ_i, θ_i, and ϕ_i are scalars computed in the course of the recursion. The cost of this algorithm is then dominated by the bidiagonalization procedure (which involves matrix-vector products), i.e., by Algorithm B. It is therefore interesting to reduce this cost. One possible way of doing this is to assume that the columns in B (and also those in C) are mutually orthogonal. It is easy to check that this is equivalent to A^TA possessing "Property A" in the sense of Young [17]. Since this property is known to reduce the cost of computing the solution to (7) when using a conjugate gradient approach [4], [16], one may expect a similar reduction of the work for Algorithm B. This is indeed the case, and one can state the following.

PROPOSITION 1. *Consider a matrix A of the form $A = (A_1, A_2)$ and assume that the columns of A_1 (and also those of A_2) are mutually orthogonal. Assume furthermore that we use Algorithm* B *with* (28) *replaced by the choice of $\mathbf{u}_1 (\neq 0)$ satisfying the condition*

$$A_2^T \mathbf{u}_1 = 0. \tag{33}$$

Then, if the vectors $\mathbf{x}_i$, $\mathbf{v}_i$ *are partitioned according to the partitioning of* A *into* $\mathbf{v}^T = (\mathbf{v}^{(1)}, \mathbf{v}^{(2)})^T$, *we have that*

$$\mathbf{v}_{2i}^{(1)} = 0, \qquad \mathbf{v}_{2i-1}^{(2)} = 0, \tag{34}$$

and if $\alpha_i \neq 0$,

$$\alpha_i^2 + \beta_{i+1}^2 = 1 \tag{35}$$

for $i = 1, 2, \cdots$.

The proof of this statement can be found in the Appendix. It has the important consequence that the matrix-vector products appearing in (30) and (31) are essentially reduced by half because the zero components of $\mathbf{v}_i$ need not be computed explicitly. This reduction is important since these products constitute the major cost in the computation, as mentioned above.

However, the assumption of mutual orthogonality of the columns in the submatrices B and C (or A_1 and A_2) is in general not satisfied. But, when considering (2) and (4), it becomes apparent that the block-columns in B and C are mutually orthogonal, so that we only have to care about orthogonalizing columns belonging to the same block-column. This can be achieved by computing a QR factorization of each one of these block-columns.

In practice, obtaining the desired orthogonality properties therefore consists in preconditioning the matrix A suitably, i.e., in solving the equivalent problem

$$\min_{\mathbf{z}} \|E\mathbf{z} - \mathbf{b}\|_2 \tag{36}$$

where

$$E = (E_1, E_2) = AM^{-1} = (BR(B)^{-1}, CR(C)^{-1}) \equiv (Q_B, Q_C), \tag{37}$$

$$M\mathbf{x} = \mathbf{z}, \tag{38}$$

with

$$M = \begin{pmatrix} R(B) & 0 \\ 0 & R(C) \end{pmatrix}. \tag{39}$$

Observe now that because of the natural orthogonality of the block-columns, we have

$$R(B) = \bigoplus_{i=1}^{m} R(B_i) \tag{40}$$

and

$$R(C) = \bigoplus_{j=1}^{n} R(\tilde{C}_j) \tag{41}$$

where

$$\tilde{C}_j = \begin{pmatrix} C_{1j} \\ C_{2j} \\ \vdots \\ C_{mj} \end{pmatrix}. \tag{42}$$

Note also that these matrices are small ($k \times k$ for $R(B_i)$ and $p \times p$ for $R(\tilde{C}_j)$) and therefore cheaply computed. The least squares problem (36) has now the desired orthonormality properties (i.e., E^TE has "Property A") and the above proposition

applies. Moreover, it can be seen that

$$E^T E = \begin{pmatrix} I & K \\ K^T & I \end{pmatrix}, \tag{43}$$

$$K = Q_B^T Q_C, \tag{44}$$

where Q_B and Q_C are defined by (37). Note that the singular values of K defines the angles between the subspaces defined by B and C (cf. [12, pp. 428-429]). The rate of convergence of conjugate gradient method will be related to the spectrum of the singular values of K. The number of iterations will strongly depend on the geometric configuration of these subspaces. A theoretical upper bound of number of iterations is clearly given by

$$2 \operatorname{rank}(K) + 1 \leqq 2 \min(mk, np) + 1. \tag{45}$$

Hence, the conjugate gradient method applied to (43) will converge, in exact arithmetic, in at most $(2np+1)$ iterations. This upper bound carries out obviously over to the LSQR algorithm of [15] when applied to (36).

If we now wish to obtain the computational benefits promised by our above proposition, we have to compute a starting vector $\mathbf{z}_0$ such that

$$\beta_1 \mathbf{u}_1 = \mathbf{b} - E\mathbf{z}_0 \tag{46}$$

and

$$E_2^T \mathbf{u}_1 = \mathbf{0}. \tag{47}$$

This can clearly be achieved by choosing

$$\mathbf{z}_0 = \begin{pmatrix} \mathbf{0} \\ \mathbf{z}_0^{(2)} \end{pmatrix} \tag{48}$$

where $\mathbf{z}_0^{(2)}$ is computed from

$$Q_C^T \mathbf{b} = \mathbf{z}_0^{(2)} \tag{49}$$

since

$$\beta_1 E_2^T \mathbf{u}_1 = E_2^T \mathbf{b} - E_2^T E_2 \mathbf{z}_0^{(2)} \tag{50}$$

and $\mathbf{z}_0^{(2)}$ is the solution to

$$\min_{\mathbf{z}} \| E_2 \mathbf{z} - \mathbf{b} \|_2 = \min_{\mathbf{z}} \| \mathbf{z} - \mathbf{z}_0^{(2)} \|_2. \tag{51}$$

In other words, the starting vector $\mathbf{z}_0$ is calculated by applying to the right-hand side $\mathbf{b}$ the same orthogonal transformations that triangularize C.

We are now ready to describe the resulting algorithm.

ALGORITHM A.

Step 1: Compute $R(B)$, $R(C)$, and $\mathbf{z}_0^{(2)}$ from (40), (41), and (49).

Step 2: Compute

$$\beta_1 \mathbf{u}_1 = \mathbf{b} - E_2 \mathbf{z}_0^{(2)}, \tag{52}$$

$$\alpha_1 \mathbf{v}_1^{(1)} = E_1^T \mathbf{u}_1. \tag{53}$$

Step 3: For $i = 1, \cdots, n_{it}$
If i is odd then

$$\beta_{i+1}\mathbf{u}_{i+1} = E_1\mathbf{v}_i^{(1)} - \alpha_i\mathbf{u}_i, \tag{54}$$

$$\alpha_{i+1}\mathbf{v}_{i+1}^{(2)} = E_2^T\mathbf{u}_{i+1}. \tag{55}$$

If i is even then

$$\beta_{i+1}\mathbf{u}_{i+1} = E_2\mathbf{v}_i^{(2)} - \alpha_i\mathbf{u}_i, \tag{56}$$

$$\alpha_{i+1}\mathbf{v}_{i+1}^{(1)} = E_1^T\mathbf{u}_{i+1}. \tag{57}$$

Compute the scalars ϕ_i, ρ_i and θ_i according to the LSQR method [15] and update $\mathbf{z}_i$ using (32).

Step 4: Solve

$$R(B)\mathbf{x}^{(1)} = \mathbf{z}_{n_{it}}^{(1)} \quad \text{and} \quad R(C)\mathbf{x}^{(2)} = \mathbf{z}_{n_{it}}^{(2)} \tag{58}$$

where n_{it} is the number of iterations to reach sufficient accuracy.

In order to derive approximate operation counts for this algorithm, we will assume that

$$m \gg n,\ k \ll n \quad \text{and} \quad p \ll n. \tag{59}$$

Furthermore, we will assume that n_{it} is given by (45). Under these weak conditions, the cost of the method, that is concentrated in (54)–(57), is

$$O(mn^2p(k+p)^2) \text{ operations.} \tag{60}$$

We observe that the storage requirements can be large if we want to store A in memory. On the other hand, it can be quite convenient to store it on auxiliary storage, block-column by block-column, at the cost of some input/output at each iteration. The number of nonzero elements of A and b is given by

$$|A| = mn(k+p+1)(k+p)/2 \tag{61}$$

and

$$|b| = mn(k+p). \tag{62}$$

It is interesting to observe that Algorithm A is naturally suited to parallel computation. Indeed, the bulk of the computation is concentrated in the matrix-vector products in (54)–(57). Since the matrices E_1 and E_2 are structured into independent blocks, the products involving these blocks can be worked out in parallel, when several processors are available. This would substantially reduce the cost of the main computational effort. Observe also that, because of (40) and (41), the work involved in Step 1 of Algorithm A can be shared between simultaneous processes. Similarly, the structure of $R(B)$ and $R(C)$ also suggests a parallel implementation of (58). As we see, all steps of Algorithm A lend themselves to parallel computation, and a substantial reduction in overall cost would be expected using this approach.

3.3. A projection algorithm. In [10], Golub and Mayers proposed a method based on the normal equations (7), which eliminates unknowns corresponding to the block B before using preconditioned conjugate gradient. As before, in order to preserve stability and avoid fill-in, we prefer to work on A directly, instead of working with A^TA.

The procedure can be viewed as the solution of two successive problems

$$\min_{\mathbf{x}} \|P(B)C\mathbf{x}^{(2)} - \mathbf{b}\|_2, \tag{63}$$

followed by

$$\min_{\mathbf{x}} \|B\mathbf{x}^{(1)} - (\mathbf{b} - C\tilde{\mathbf{x}}^{(2)})\|_2, \tag{64}$$

where $\tilde{\mathbf{x}}^{(2)}$ is the solution to (63) and

$$P(B) = I - B(B^TB)^{-1}B^T \tag{65}$$

is the orthogonal projection onto range $(B)^{\perp}$. Problem (63) is solved by using a preconditioned LSQR method again, where the preconditioner matrix M can be chosen adequately. We will choose here

$$M = R(C), \tag{66}$$

although this is not the only possible choice.

In practice, the matrix (65) is never computed explictly, but as

$$P(B) = Q\begin{pmatrix} 0 & 0 \\ 0 & I \end{pmatrix} Q^T = I - Q_B Q_B^T \tag{67}$$

where

$$Q^TB = \begin{pmatrix} R(B) \\ 0 \end{pmatrix}. \tag{68}$$

It means that $P(B)$ is stored as a product of orthogonal transformations that triangularize B. The solution of (64) is then given by

$$R(B)\mathbf{x}^{(1)} = Q_B^T(\mathbf{b} - C\tilde{\mathbf{x}}^{(2)}). \tag{69}$$

The resulting computational procedure is then as follows.

ALGORITHM P.

Step 1: Compute $R(B)$ and store the corresponding Q^T as a product of orthogonal transformations. Compute also $R(C)$.

Step 2: Compute

$$\beta_1\mathbf{u}_1 = \mathbf{b}, \tag{70}$$

$$\alpha_1\mathbf{v}_1 = R(C)^{-T}C^TP(B)\mathbf{u}_1, \tag{71}$$

$$\mathbf{z}_0^{(2)} = 0. \tag{72}$$

Step 3: For $i = 1, 2, \cdots, n_{it}$, compute

$$\beta_{i+1}\mathbf{u}_{i+1} = P(B)CR(C)^{-1}\mathbf{v}_i - \alpha_i\mathbf{u}_i, \tag{73}$$

$$\alpha_{i+1}\mathbf{v}_{i+1} = R(C)^{-T}C^TP(B)\mathbf{u}_{i+1} - \beta_{i+1}\mathbf{v}_i. \tag{74}$$

Compute also the scalars ϕ, ρ_i and θ_i according to the LSQR method [15] and update $\mathbf{z}_i^{(2)}$ using (32).

Step 4: Solve successively

$$R(C)\mathbf{x}^{(2)} = \mathbf{z}_{n_{it}}^{(2)} \tag{75}$$

and (69).

If we form the normal equations for the problem (63), using the preconditioner (66), we obtain

$$E^TE = I - Q_C^TQ_BQ_B^TQ_C. \tag{76}$$

The expected number of iterations is then half of that for Algorithm A. Note however that we cannot exploit Property A and that the cost per iteration is higher. In fact, this algorithm can be viewed as a Gauss-Seidel type acceleration of the conjugate gradient since the global preconditioning factor $\bar{M}$ can be computed as

$$\bar{M} = \begin{pmatrix} R(B) & \omega R(B)^{-T} B^T C \\ 0 & R(C) \end{pmatrix} \tag{77}$$

with $\omega = 1$. It is therefore possible to generalize this algorithm by choosing the relaxation parameter ω between 1 and 2. This corresponds to the block-SSOR acceleration, proposed in [1]. Note that, in this case, the splitting of the problem in (63) and (64) is no more valid. The optimal value of ω can be deduced from the classical SOR theory and is given in [5]. This value is related to the maximum singular value of K.

Assuming (59) and that the number of iterations is given by np, we may analyze the operation count for Algorithm P. The dominant cost is clearly located in (73) and (74). The cost of one of the involved products is

$$O(mn(kp + p(p+1)/2) + mnk(k+1)) \text{ operations} \tag{78}$$

where the first term accounts to the product by C (or C^T), while the second covers the applications of the orthogonal transformations stored to define (67). The total operation count is then evaluated as

$$O(mn^2 p((k+p)^2 + k^2)) \text{ operations.} \tag{79}$$

However, because of the freedom to choose the preconditioner $\bar{M}$ and a relaxation parameter ω, this work can possibly be reduced further. Finally, observe that (79) is always greater than (60), so we expect Algorithm P to be less efficient than Algorithm A.

Parallelism also appears naturally in Algorithm P, in a way similar to that already described for Algorithm A.

3.4. Standard errors. For linear regression problems with a matrix A, $n_1 \times n_2$ and $n_1 > n_2$, the standard error in the ith component of the solution $\mathbf{x}$ is taken to be s_i, where

$$s_i^2 \equiv \sigma^2 c_i, \tag{80}$$

$$\sigma^2 \equiv \|A\mathbf{x} - \mathbf{b}\|_2^2 / (n_1 - n_2), \tag{81}$$

and c_i is the ith diagonal element of $(A^T A)^{-1}$. For direct methods, c_i can be computed by the formula

$$c_i = \|R^{-T} \mathbf{e}_i\|_2^2, \tag{82}$$

where R is the upper triangular factor and $\mathbf{e}_i$ is the ith coordinate vector. Unfortunately, an exact computation of c_i is not possible when iterative methods are used since R is never formed. However an estimation of c_i can be given as described by Paige and Saunders [15]. Using LSQR on (1) with a preconditioning factor M, c_i is estimated by

$$c_i = \gamma_{n_{it}}^{(i)} \tag{83}$$

where

$$\gamma_k^{(i)} = \gamma_{k-1}^{(i)} + ([M^{-1}\mathbf{d}_k]^{(i)})^2, \qquad k = 1, \cdots, n_{it}, \quad \gamma_0^{(i)} = 0, \tag{84}$$

where the superscript i denotes the ith component of vector; $\mathbf{d}_k$ is computed as in (32).

4. Numerical tests and discussion. In this section, we will present some numerical experience and discuss the relative advantages of three methods described above. This

comparison will be made by running the three algorithms on data arising from Doppler positioning. This problem, described briefly above, has motivated this whole study. Both simulated and real data will be used in the tests; the real data coming from the European Network (EDOC2) [3].

All experiments were run in double precision on the DEC 2060 of the Facultés Universitaires de Namur (Belgium) using the FORTRAN 77 compiler without optimization. This machine has a word length of 36 bits, and the mantissa of a double precision number uses 63 bits. All CPU times will be reported in seconds, and we give the time needed by an algorithm to obtain maximal accuracy, when an iterative procedure is involved. Data volumes and storage requirements will be expressed in thousands of double precision numbers.

In all the test problems, the parameters k and p were 6 and 3 respectively, corresponding to 3 (positions)+3 (speed) components of the satellite orbits, and 3 position corrections for the ground stations. The nonzero coefficients were chosen using a pseudo-random generator. The parameters m and n range for the simulated problems between 10 and 60, and 5 and 30 respectively. Problems were also generated where only 25% of the block-rows associated with the structure (5) appear. These will be referred to as the "sparse" test cases, because this can be interpreted as a certain sparsity structure in the $m \times n$ table describing the occurrence of the block-rows in any given problem. The smallest of the simulated problems we have run had 450 equations and 75 variables ($m = 10$, $n = 5$) while the largest one had 10,800 equations and 420 unknowns ($m = 60$, $n = 20$). The dimensions of the European network were described by (6).

The first comparison we consider is that between Algorithm A and Algorithm P. As observed above, the operation count is asymptotically higher for Algorithm P, so that we expect it to be slower than Algorithm A. This can be checked by considering Table 1.

TABLE 1
A comparison between Algorithms A *and* P.

m	n	CPU (Alg. A)	CPU (Alg. P)
10	5	1.6	2.3
15	10	4.0	6.9
30	15	10.4	18.5
40	25	23.7	36.3

In these tests, the problems considered were well scaled, and both algorithms produced accurate solutions.

Since Algorithm A is better than Algorithm P, both theoretically and experimentally, no further tests will be described using Algorithm P. Before dismissing it completely, it is worth mentioning that, possibly, a better choice of preconditioner could reverse the present conclusion. Note also that the use of a relaxation parameter as in [1] may improve Algorithm P. However, the reduction of the number of iterations would have to be about three in order to compete with Algorithm A.

We now wish to compare more extensively Algorithms A and D. We first examine their dependence on n, i.e., on the number of block-columns in (4). In the Doppler positioning context, this amounts to the number of ground stations.

TABLE 2
Algorithms A *and* D: *dependence of* CPU *time on n.*

$m = 50$	CPU Alg. A		CPU Alg. D	
n	sparse	dense	sparse	dense
5	8.2	17.3	2.6	21.7
10	16.7	31.3	14.2	111.7
15	29.7	79.3	45.2	338.0
20	38.1	88.5	102.6	724.9
25	43.7	180.2	204.7	1,299.0
30	52.5	183.6	326.3	2,069.4

A comparison is made in Table 2, on the basis of CPU time, excluding all the input/output operations. It is important to observe that for the sparse problems, a heuristic block-column ordering [13] was used in Algorithm D for exploiting the sparsity. Results without this reordering are consistently about 20% worse, and are therefore omitted. It is possible to adapt other column ordering strategies such as proposed by George and Heath [6] or George and Ng [8] but, because of the relative high density of the problems, one could not expect very significant savings in computations using them.

Except for small n, i.e., $n = 5$ and $n = 10$, Algorithm A significantly outperforms Algorithm D in execution speed. The observed behavior in $O(n^3)$ operations for Algorithm D is expected from (24). On the other hand, Algorithm A exhibits a behavior that is better than $O(n^2)$, as would be suggested by (60). This is due to the very slow increase in the number of iterations, which, in these experiments, are substantially slower than that predicted by (44). Although one may attribute this fact to a poor choice of test problems, it should be observed that a similar behavior was obtained on real data, which was rather poorly conditioned.

In order to complete our comparison on n, we should also examine the respective amounts of file and memory storage and the amount of input/output operations for the two methods. These amounts are summarized in Table 3 for the dense problems and in Table 4 for the sparse counterparts.

In both tables, the columns "incore" and "outcore" under the heading of Algorithm A refer to two extreme ways of manipulating the storage for this method. The "incore" version keeps the entire matrix A in main memory, while the "outcore" version stores it on auxiliary files and reads it in every time it is needed, i.e., on every iteration of the LSQR process. As one can see, the "incore" version requires more memory but much less I/O operations. The file space is the same for both variants, since the matrix is stored on file at least once (when computing the preconditioning factor).

Algorithm A "incore" seems a reasonable choice when possible. This may not be the case if the machine memory is very small. In this case, one could choose Algorithm A "outcore" if file space is also scarce but I/O speed adequate, and Algorithm D in the other case. Observe however that one has to balance the relative costs of a larger number of arithmetic operations and a large volume of I/O. Observe also that the I/O counts for Algorithm D increase faster than those of Algorithm A "outcore". It

TABLE 3
Storage and I/O comparison for Algorithms A and D on dense problems with variable n.

$m = 50$	MEMORY SPACE			FILE SPACE		INPUT/OUTPUT		
n	Algorithm A		Alg. D	Alg. A	Alg. D	Algorithm A		Alg. D
	incore	outcore				incore	outcore	
5	16.5	5.3	1.2	13.5	38.5	27.0	465.8	77.1
10	30.0	7.6	2.1	27.0	121.0	54.0	1,246.5	242.1
15	43.6	9.9	4.5	40.5	248.5	81.0	2,409.8	497.1
20	57.2	12.2	7.8	54.0	421.5	108.0	2,808.0	843.0
25	70.1	14.5	12.0	67.5	538.5	135.0	3,741.4	1,077.1
30	84.3	16.8	17.0	81.0	901.0	162.0	4,144.5	1,802.1

TABLE 4
Storage and I/O comparison for Algorithms A and D on sparse problems with variable n.

$m = 50$	MEMORY SPACE			FILE SPACE		INPUT/OUTPUT		
n	Algorithm A		Alg. D	Alg. A	Alg. D	Algorithm A		Alg. D
	incore	outcore				incore	outcore	
5	6.0	3.2	0.8	3.3	6.0	6.7	132.4	12.1
10	9.5	3.9	1.3	6.7	14.8	13.5	367.9	29.6
15	13.0	4.5	1.9	10.1	28.5	20.2	710.2	56.9
20	16.4	5.2	3.9	13.5	38.5	27.0	972.0	77.1
25	19.8	5.8	4.7	16.8	55.8	33.7	1,156.9	111.5
30	23.3	6.5	6.3	20.2	70.0	40.5	1,458.0	140.1

is also obvious that, in practice, some compromise could be worked out between the two variants of Algorithm A, in order to use all the available memory space to advantage.

It is fortunate that time and a lack of suitable hardware did not allow the authors to test the inherent parallelism of Algorithm A. The authors nevertheless feel that this experimentation is worthwhile and should be carried out. As discussed above, substantial savings can be expected when the number of available processors increases.

We now compare the same two methods, but when varying m, the number of block-columns in (3). This is the number of satellite passes in the Doppler positioning problem.

Table 5 shows the CPU times (exlcuding I/O again) for both methods and both dense and sparse cases. Algorithm A is again better on this ground. Tables 6 and 7 report the storage and I/O for the corresponding problems.

TABLE 5
Algorithms A *and* D: *dependence of* CPU *time on m.*

$n = 20$	CPU Alg. A		CPU Alg. D	
m	sparse	dense	sparse	dense
20	15.9	35.0	37.1	301.5
30	23.5	52.1	56.4	446.9
40	28.8	67.9	68.4	540.4
50	38.1	88.5	102.6	724.9
60	42.8	138.5	118.5	812.5

These three tables show again that, when it is feasible, Algorithm A "incore" is better. When the machine does not allow it, the choice between Algorithm A "outcome" and Algorithm D is again dependent on the relative cost of arithmetic and *I/O.*

TABLE 6
Storage and I/O comparison for Algorithms A *and* D *on dense problems.*

$n = 20$	MEMORY SPACE			FILE SPACE		INPUT/OUTPUT		
m	Algorithm A		Alg. D	Alg. A	Alg. D	Algorithm A		Alg. D
	incore	outcore				incore	outcore	
20	22.9	5.0	7.8	21.6	168.4	43.2	1,033.2	336.8
30	34.4	7.4	7.8	32.4	252.6	64.8	1,468.8	505.3
40	45.8	9.8	7.8	43.2	336.8	86.4	2,210.4	673.7
50	57.2	12.2	7.8	54.0	421.5	108.0	2,808.0	843.0
60	68.6	14.6	7.8	64.8	505.3	129.6	3,423.6	1,010.5

TABLE 7
Storage and I/O comparison for Algorithms A *and* D *on sparse problems.*

$n \times 20$	MEMORY SPACE			FILE SPACE		INPUT/OUTPUT		
m	Algorithm A		Alg. D	Alg. A	Alg. D	Algorithm A		Alg. D
	incore	outcore				incore	outcore	
20	6.8	2.9	2.8	5.4	15.4	10.8	370.8	30.8
30	10.0	3.2	3.1	8.1	23.1	16.2	556.2	45.3
40	13.2	4.2	3.5	10.8	30.8	21.6	759.6	61.7
50	16.4	5.2	3.9	13.5	38.5	27.0	972.0	77.1
60	19.6	6.1	3.9	16.2	46.3	32.4	1,139.4	92.5

Finally, Table 8 shows the results obtained by running both algorithms on the European network.

In this table, $d = 0.45$ stands for the proportion of block-rows present in the structure (5). The density is therefore intermediate between the dense and the sparse problems considered above. In our experience, and although the I/O was programmed very crudely in our test code, Algorithm A outperformed Algorithm D in overall cost. We expect this to be reinforced when a more careful coding is done that will compromise between the "incore" and "outcore" approaches more adequately.

TABLE 8
Results on the European network.

m	n	d	CPU Alg. A	CPU Alg. D
152	34	0.45	438.8	3,516.9

MEMORY SPACE			FILE SPACE		INPUT/OUTPUT		
Algorithm A		Alg. D	Alg. A	Alg. D	Algorithm A		Alg. D
incore	outcore				incore	outcore	
133.5	29.2	17.3	125.1	981.6	250.2	9,529.8	1,963.1

It is also worth noting that the standard errors estimates produced by Algorithm A were usually rather poor (about one significant digit) in contrast to results obtained by Algorithm D. This can provide an incentive to use Algorithm D, when accuracy on these quantities is very important.

Finally, we may now consider the case where the matrices (B_{ij}, C_{ij}) $(i = 1, \cdots, m, j = 1, \cdots, n)$ are no longer triangular but generally rectangular. Most of the conclusions still apply since it is advantageous, in most cases, to reduce these matrices to upper triangular form first before applying the algorithms developed here (see [14]). One should also observe that B and C in (2) could be exchanged if $mk < np$, yielding a system with a similar structure after a trivial column permutation.

5. Conclusion. According to our numerical tests, an iterative technique for solving the linear least squares problem (1), structures in dense blocks as described in the second section, is often useful and can produce significant savings in overall execution time, especially when the main memory of the computer is sufficiently large, or when the input/output operations are fast.

Appendix. We consider here the proof of Proposition 1. From the initialization step (29) and hypothesis (33), it is straightforward that

$$\alpha_1 \mathbf{v}_1 = \begin{pmatrix} A_1^T \mathbf{u}_1 \\ \mathbf{0} \end{pmatrix}. \tag{85}$$

Assume now that, for $\alpha_i \neq 0$,

$$\alpha_i \mathbf{v}_i = \begin{pmatrix} \mathbf{v}_i^{(1)} \\ \mathbf{v}_i^{(2)} \end{pmatrix} = \begin{pmatrix} A_1^T \mathbf{u}_i \\ \mathbf{0} \end{pmatrix}. \tag{86}$$

Then, from (30), we have that

$$\beta_{i+1}^2 = \mathbf{v}_i^{(1)^T} A_1^T A_1 \mathbf{v}_i^{(1)} + \alpha_i^2 - 2\alpha_i \mathbf{v}_i^{(1)^T} A_1^T \mathbf{u}_i. \tag{87}$$

By the hypothesis of orthogonality ($A_1^T A_1 = I$), it follows that

$$\beta_{i+1}^2 = 1 - \alpha_i^2. \tag{88}$$

Now consider equation (31). We have

$$\alpha_{i+1}\mathbf{v}_{i+1}^{(1)} = A_1^T \mathbf{u}_{i+1} - \beta_{i+1}\mathbf{v}_i^{(1)}, \tag{89}$$

$$\alpha_{i+1}\mathbf{v}_{i+1}^{(2)} = A_2^T \mathbf{u}_{i+1}. \tag{90}$$

Replacing $\mathbf{u}_{i+1}$ by its expression in (30) and using (86), we obtain when $\beta_{i+1} \neq 0$

$$\alpha_{i+1}\mathbf{v}_{i+1}^{(1)} = \frac{1}{\beta_{i+1}}(1-\alpha_i^2)\mathbf{v}_i^{(1)} - \beta_{i+1}\mathbf{v}_i^{(1)} = \mathbf{0}. \tag{91}$$

We have thus proved that using Algorithm B with $A_1^T A_1 = I$, the following:

$$\alpha_i \mathbf{v}_i = \begin{pmatrix} A_1^T \mathbf{u}_i \\ \mathbf{0} \end{pmatrix} \Rightarrow \begin{cases} \text{(a)} \ \ \alpha_i^2 + \beta_{i+1}^2 = 1 \\ \text{(b)} \ \ \alpha_{i+1}\mathbf{v}_{i+1} = \begin{pmatrix} \mathbf{0} \\ A_2^T \mathbf{u}_{i+1} \end{pmatrix} \end{cases} \tag{92}$$

since the corollary result,

$$\alpha_i \mathbf{v}_i = \begin{pmatrix} \mathbf{0} \\ A_2^T \mathbf{u}_i \end{pmatrix} \Rightarrow \begin{cases} \text{(a)} \ \ \alpha_i^2 + \beta_{i+1}^2 = 1 \\ \text{(b)} \ \ \alpha_{i+1}\mathbf{v}_{i+1} = \begin{pmatrix} A_1^T \mathbf{u}_{i+1} \\ \mathbf{0} \end{pmatrix} \end{cases} \tag{93}$$

assuming $A_2^T A_2 = I$, is clearly deduced, and hence Proposition 1 is demonstrated.

Acknowledgments. The authors wish to thank P. Pâquet who introduced them to the problem and who provided data of the European network and Ch. Murigande for helpful discussions. Thanks are also due to Åke Björck who provided very useful advice.

REFERENCES

[1] ÅKE BJÖRCK, SSOR *preconditioning methods for sparse least squares problems*, in Proc. Computer Science and Statistics 12th Annual Symposium on the Proceedings Interface, 21-25. Univ. Waterloo, Ontario, Canada, 1979.

[2] ÅKE BJÖRCK AND T. ELFVING, *Accelerated projection methods for computing pseudoinverse solutions of systems of linear equations*, BIT, 19 (1979), pp. 145-163.

[3] C. BOUCHER, P. PÂQUET AND P. WILSON, *The second European Doppler observation campaign* (EDOC-2): *results and conclusions obtained by* EDOC-2 *computing centers*, in Proc. Second International Geodetic Symposium on Satellite Doppler Positioning, Univ. Texas, Austin, 1979, pp. 819-849.

[4] P. CONCUS, G. H. GOLUB AND D. P. O'LEARY, *A generalized conjugate gradient method for the numerical solution of elliptic partial differential equations*, in Sparse Matrix Computation, J. Bunch and D. Rose, eds., Academic Press, New York, 1976, pp. 309-322.

[5] T. ELFVING, *Block-iterative methods for consistent and inconsistent linear equations*, Numer. Math., 35 (1980), pp. 1-12.

[6] J. A. GEORGE AND M. T. HEATH, *Solution of sparse linear least squares problems using Givens rotations*, Linear Algebra Appl., 34 (1980), pp. 69-73.

[7] J. A. GEORGE, M. T. HEATH AND R. J. PLEMMONS, *Solution of large scale least squares problems using auxiliary storage*, this Journal, 2 (1981), pp. 416-429.

[8] J. A. GEORGE AND E. NG, *On row and column orderings for sparse least squares problems*, SIAM J. Numer. Anal., 20 (1983), pp. 326-344.

[9] G. J. GOLUB AND W. KAHAN, *Calculating the singular values and pseudo-inverse of a matrix*, SIAM J. Numer. Anal., 2 (1965), pp. 205-224.

[10] G. H. GOLUB AND D. MAYERS, *The use of preconditioning over irregular regions*, Numerical Analysis Project, Manuscript NA-83-27, Stanford Univ., Stanford, CA, 1983.

[11] G. H. GOLUB AND R. J. PLEMMONS, *Large scale geodetic least squares adjustment by dissection and orthogonal decomposition*, Linear Alg. Appl., 34 (1980), pp. 3-27.

[12] G. H. GOLUB AND CH. F. VAN LOAN, *Matrix Computations*, North Oxford Academic, New York, 1983.

[13] P. MANNEBACK, *A direct approach or column ordering and symbolic factorization in large sparse least squares problems*, Technical Report 83/11, Dept. Mathematics, FNDP Namur, Belgium, 1983.

[14] P. MANNEBACK, CH. MURIGANDE AND PH. L. TOINT, *A modification of an algorithm by Golub and Plemmons for large linear least squares in the context of Doppler positioning*, Technical Report 84/01, Dept. of Mathematics, FNDP Namur, Belgium, 1984.

[15] C. C. PAIGE AND M. A. SAUNDERS, LSQR: *an algorithm for sparse linear equations and sparse least squares*, ACM Trans. Math. Software, 8 (1982), pp. 43-71.

[16] J. K. REID, *The use of conjugate gradients for systems of linear equations possessing 'Property* A', SIAM J. Numer. Anal., 9 (1972), pp. 325-332.

[17] D. M. YOUNG, *Iterative Solution of Large Linear Systems*, Academic Press, New York, 1971.

SIAM J. SCI. STAT. COMPUT.
Vol. 7, No. 3, July 1986

GENERALIZATIONS OF DAVIDSON'S METHOD FOR COMPUTING EIGENVALUES OF SPARSE SYMMETRIC MATRICES*

RONALD B. MORGAN† AND DAVID S. SCOTT‡

Abstract. This paper analyzes Davidson's method for computing a few eigenpairs of large sparse symmetric matrices. An explanation is given for why Davidson's method often performs well but occasionally performs very badly. Davidson's method is then generalized to a method which offers a powerful way of applying preconditioning techniques developed for solving systems of linear equations to solving eigenvalue problems.

Key words. eigenvalues, eigenvectors, sparse matrices

AMS(MOS) subject classifications. 65, 15

1. Introduction. The Lanczos algorithm is a powerful technique for computing a few eigenvalues of a symmetric matrix A. If it is practical to factor the matrix $(A-\sigma)$ (which is the notation which will be used for the matrix $(A-\sigma I)$) for one or more values of σ near the desired eigenvalues, then the Lanczos algorithm can be used with the inverted operator and convergence will be very rapid. Otherwise the Lanczos algorithm can be used with the original matrix A but convergence can be very slow.

Slow convergence can also plague the conjugate gradients method for solving systems of linear equations which is an analog of the Lanczos algorithm. Convergence of the conjugate gradients algorithm can be accelerated by computing and using an approximate inverse (preconditioner). Much work has been done developing effective preconditioning techniques ([3], [5], [8], [1]). Unfortunately preconditioning cannot be directly applied to eigenvalue problems. If the preconditioner multiplies both sides of the equation $Az=\lambda z$ then the problem becomes a generalized eigenvalue problem which is no easier to solve. If only A is multiplied by the preconditioner, then the eigenpairs are changed.

One approach for using preconditioners on eigenvalue problems is to convert them to linear equation problems by using inverse iteration or the Rayleigh quotient iteration and then use preconditioned conjugate gradients (actually SYMMLQ [6] since the matrices involved will be indefinite). This approach was investigated by Szyld [9]. A different approach is Davidson's method [2] which can be interpreted as a method for using diagonal preconditioning in solving eigenvalue problems. This paper analyzes Davidson's method from this point of view and then generalizes it to obtain a method which uses more powerful types of preconditioners.

2. Davidson's method. Davidson [2] introduced a new method for computing a few eigenvalues of sparse symmetric matrices arising in quantum chemistry calculations. The standard solution technique for such problems is the Lanczos algorithm [7, Chap. 13] which is a clever implementation of the Rayleigh–Ritz procedure applied to a Krylov subspace (that is, a space of the form span $(s, As, \cdots, A^k s)$). Davidson's method also uses the Rayleigh–Ritz procedure (see [7, p. 213]) but on a non-Krylov subspace. Formally Davidson's method is as follows.

* Received by the editors October 31, 1984, and in revised form April 26, 1985.

† Mathematics Department, University of Texas at Austin, Austin, Texas 78712.

‡ Intel Scientific Computers, Beaverton, Oregon 97006. The work of this second author was partially supported by the Applied Mathematical Sciences Research Program, Office of Energy Research, U.S. Department of Energy, under contract W-7405-eng-26.

Davidson's Method.

Initialize: Let A be the matrix of interest and let D be the diagonal of A. Choose an initial trial space $P_k = \text{span}\,(p_1, p_2, \cdots, p_k)$ and compute (y_k, θ_k) the best approximation to the eigenpair of interest using the Rayleigh-Ritz procedure and compute the residual vector

$$r_k = Ay_k - y_k\theta_k.$$

Then FOR $j = k+1, k+2, \cdots$ until convergence DO 1 to 4

1. Compute $p_j = (D - \theta_{j-1})^{-1} r_{j-1}$
2. Set $P_j = \text{span}\,(P_{j-1}, p_j)$
3. Compute (y_j, θ_j) from P_j using the Rayleigh-Ritz procedure
4. Compute the residual $r_j = (A - \theta_j)y_j$

Convergence is measured by the norm of the residual vector.

Thus the new trial vector is just $(D-\theta)^{-1}(A-\theta)y$. Except for the cost of forming the matrix vector product (which is the same for either method), this algorithm is much more expensive per step than the Lanczos algorithm since a full Gram-Schmidt process is needed to compute an orthogonal basis for the space H and a full (rather than tridiagonal) reduced matrix is generated by the Rayleigh-Ritz procedure. Despite this higher overhead, Davidson reported favorable results for his method compared to the Lanczos algorithm for problems arising in molecular energy calculations. In one example of dimension 372, Davidson's method reduced the residual norm to $1e$-6 in 10 iterations while 28 iterations of Lanczos reduced the residual norm to only $2e$-2. The improvement is due entirely to the premultiplication by the matrix $(D-\theta_j)^{-1}$ in step 1, since without this perturbation, Davidson's method would just reduce to a very expensive way of implementing the Lanczos algorithm (provided $k = 1$).

The new trial vector obtained by Davidson's method is the correction which would be obtained by one step of the Jacobi method for solving the system of equations

$$(A - \theta)x = 0$$

with y_j as the initial guess for x. This looks a little strange since in general $A-\theta$ is not singular and the only solution to the system of equations is $x = 0$. A more satisfying explanation of the algorithm is as follows. Let (y, θ) be the current approximation to the desired eigenpair. For a given coordinate i, the best improvement which can be made in y by perturbing its ith component can be determined by the Rayleigh-Ritz procedure. Let X be the $n \times 2$ matrix with y as its first column and e_i (the ith coordinate vector) as its second column. Then the best approximations to eigenpairs of A obtainable from the space spanned by the columns of X are obtained by solving the 2×2 generalized eigenvalue problem $H_i - \alpha W_i$ where

$$H_i = X^T A X = \begin{bmatrix} y^T Ay & y^T Ae_i \\ e_i^T Ay & e_i^T Ae_i \end{bmatrix} = \begin{bmatrix} \theta & (Ay)_i \\ (Ay)_i & a_{ii} \end{bmatrix},$$

$$W_i = X^T X = \begin{bmatrix} y^T y & y^T e_i \\ e_i^T y & e_i^T e_i \end{bmatrix} = \begin{bmatrix} 1 & y_i \\ y_i & 1 \end{bmatrix}$$

and y_i is the ith component of the vector y. If (s, α) is an eigenpair of this system, then α is the approximate eigenvalue (called Ritz value) and $z = Xs$ is the corresponding approximate eigenvector (Ritz vector).

If we are near convergence then the residual of y will be small and one of the α's will be near θ. The corresponding eigenvector can be approximated by examining

the matrix

$$H_i - \theta W_i = \begin{bmatrix} 0 & (Ay - y\theta)_i \\ (Ay - y\theta)_i & a_{ii} - \theta \end{bmatrix} = \begin{bmatrix} 0 & r_i \\ r_i & d_i - \theta \end{bmatrix}$$

where r_i is the ith component of the residual vector and d_i is the ith diagonal element of A. The eigenvector is (approximately)

$$s \approx (1, -(d_i - \theta)^{-1} r_i)^T.$$

The corresponding Ritz vector is

$$z = Xs \approx y - (d_i - \theta)^{-1} r_i e_i.$$

Davidson's method just lumps all of these perturbations into one vector and adds this composite vector to the trial space. (The dropping of the minus sign is of no account since only the trial space is important for the Rayleigh–Ritz procedure not the particular basis chosen.)

Example 1. To compare Davidson's method to the Lanczos algorithm, the smallest eigenpair of a symmetric matrix A of order 20 was computed using both methods. A was (unreduced) tridiagonal except that a_{1n} and a_{n1} were nonzero. $a_{ii} = i$ for all i while all other nonzero elements were 1. The starting vector was $p_1 = (1, 0.1, 0.1, \cdots, 0.1)^T$ which is moderately but not exceptionally accurate. Table 2.1 illustrates the behavior of the methods. The Ritz value for Davidson's method was accurate to 9 decimal digits at step 10 while the Lanczos value at step 10 was accurate only to 2 digits. This behavior is very similar to that reported by Davidson [2]. Note that the Lanczos algorithm has better global convergence than Davidson's method (better for the first five steps).

TABLE 2.1
A comparison of the Lanczos and Davidson methods.

Step	Davidson Ritz value	Residual norm	Lanczos Ritz value	Residual norm
1	3.23529	5.27	3.23529	5.27
2	3.17006	3.17	1.21302	1.83
3	1.65718	1.80	.784054	1.34
4	1.48600	1.78	.476551	1.07
5	.291006	.953	.320862	.664
6	.223536	.0764	.2603809	.423
7	.222866	.01177	.2352622	.264
8	.222847	.00241	.2263713	.149
9	.222846	.000229	.2237563	.0783
10	.222846	.0000249	.2230518	.0381

In the next section we examine our derivation of Davidson's method in more detail.

3. Local analysis of Davidson's method. One major assumption in the derivation of Davidson's method is the form of s, the eigenvector of interest of the 2×2 problem $H_i - \alpha W_i$. Let $\pi_i = r_i/(d_i - \theta)$ be the components of the Davidson vector p. If $|\pi_i| < 1$ then the eigenvalue near θ and corresponding eigenvector can be expanded in a power series in π_i.

$$\alpha = \theta - r_i\pi_i - 2r_i y_i \pi_i^2 + O(\pi_i^3)$$

and normalizing the first component of s to be 1,

$$s = (1, -\pi_i - y_i\pi_i^2 + O(\pi_i^3))^T.$$

(It is interesting to note that the lowest order terms do not depend on y_i.) Thus Davidson's method does implement the correct first order perturbation correction provided that $|\pi_i| < 1$.

Asymptotically r_i converges to zero while $d_i - \theta$ converges to $d_i - \lambda$, where λ is the desired eigenvalue. Thus π_i converges to zero for all i *unless* λ is equal to some diagonal element of A. If $\lambda = d_i$ for some i then the behavior of π_i depends on whether or not the corresponding eigenvector is e_i. In any case θ, the Rayleigh quotient of y, will satisfy

$$d_i - \theta = O(\|r\|^2)$$

(see [7, p. 222]). If $z = e_i$ then $y_i \approx 1$, r_i will be just $(d_i - \theta)y_i$ and π_i will approach 1. Otherwise r_i will be some constant times $\|r\|$ and π_i will diverge to infinity. In either case Davidson's method may perform badly.

Example 2. The matrix A from Example 1 was modified so that $a_{12} = a_{21} = a_{1n} = a_{n1} = 0$. This made the smallest eigenvalue of A equal to 1 with e_1 as the corresponding eigenvector. The next smallest eigenvalue is $1.2538\cdots$. The same starting vector was used. By step 8 of the algorithm the second smallest eigenvalue had been computed to 8 decimal places but no approximation to the smallest eigenvalue had appeared at all. On step 9 a poor approximation to the smallest eigenvalue appeared (1.21315) but by step 16 it had only converged to 1.0285.

Example 3. To obtain an example of the other kind of unusual behavior, the matrix A from Example 1 was modified by deleting its last row and column. The resulting matrix has 10 both as an eigenvalue and as a diagonal element but the eigenvector of 10 is not a coordinate vector. The starting vector had all equal components except the tenth which was ten times larger. By step fourteen the desired eigenvector approximation had a residual of .000587. This is almost as fast as Example 1 and much faster than Example 2. This difference in behavior will be examined in the next section.

4. Convergence of Davidson's method. Despite the dramatic results reported by Davidson for molecular energy calculations, Kalamboukis [4] reported that Davidson's method converged no faster than Lanczos on nuclear modeling problems. Since the overhead is much higher in Davidson's method, Kalamboukis recommended that Lanczos be used for this type of problem. Kalamboukis also suggested that the differing behavior of Davidson's algorithm could be explained by the degree of diagonal dominance of the matrices involved—the more diagonally dominant the matrix was, the better Davidson's method worked. This is not entirely true. If the diagonal of A is constant, then Davidson's method is equivalent to Lanczos regardless of the degree of diagonal dominance. More distressing is the fact that Davidson's method fails completely when A is a diagonal matrix since the new trial vector will just be y and the trial basis will become linearly dependent.

The best way to understand the behavior of Davidson's method is to analyze the operator $N(\theta) = (D - \theta)^{-1}(A - \theta)$. Each new trial vector is $N(\theta)$ times some vector in the space. If θ were constant, the trial space would just be the Krylov space generated by N. (Unfortunately since N is nonsymmetric in general, it would not be possible to use the symmetric Lanczos algorithm on it.) Of course θ is not constant but it does converge to the desired eigenvalue λ and so the properties of $N(\theta)$ for values of θ near λ are crucial to the behavior of the algorithm. $N(\theta)$ is just the operator obtained by applying diagonal scaling to $(A - \theta)$. It is well known, from studying diagonal scaling as a preconditioner for conjugate gradients, that this diagonal scaling will tend to compress the spectrum of $(A - \theta)$ so that it is centered on 1. For conjugate gradients,

this is the goal in itself since the compressed spectrum will have a lower condition number and conjugate gradients will converge faster. For eigenvalue problems we are interested in how rapidly the Krylov subspace generated by N will contain good approximations to the desired eigenvector.

The dominant term in the convergence rate for Krylov subspaces depends on the gap ratio of the desired eigenvalue which measures the *relative* separation of the desired eigenvalue from the rest of the spectrum (see [7, p. 244]). Convergence to interior eigenvalues is also possible but usually implies earlier convergence to *all* of the eigenvalues on one side of the desired eigenvalue. Compression of the spectrum by itself is not sufficient to insure rapid convergence. Two additional conditions must be met:

1. The desired eigenvalue (the smallest eigenvalue of $(A-\theta)$) must be moved less than the rest of the spectrum so that the gap ratio of the desired eigenvalue is increased.

2. The preconditioning must not greatly perturb the desired eigenvector so that convergence to the eigenvector of N implies convergence to the desired eigenvector of A.

If the desired eigenvector is a coordinate vector, e_i say, then e_i is also an eigenvector of $N(\theta)$ for all θ and the second condition is satisfied. Unfortunately the diagonal preconditioning makes the corresponding eigenvalue exactly 1 which lies right in the middle of the compressed spectrum and so convergence is *very* slow. In the special case of a diagonal matrix A, all the eigenvalues of N are 1 and the method breaks down. Table 4.1 shows the spectrum of $N(1.0001)$ for Example 2. Note that the desired eigenvalue lies in the middle of the spectrum.

The situation is rather different in Example 3. For θ's near the desired eigenvalue, $(D-\theta)$ is nearly singular and $N(\theta)$ will have a very large singular value. After one matrix multiply a vector very close to e_{10} will be in the trial space. Essentially one step is wasted obtaining the approximation to e_{10} and then the process proceeds as if the large singular value did not exist.

TABLE 4.1
Spectrum of $N(1.0001)$ *for Example* 2.

0.1683	0.6377	0.7689	0.8304
0.8661	0.8902	0.9128	0.9392
0.9688	0.9999	*1.0000*	1.0311
1.0608	1.0872	1.1098	1.1339
1.1696	1.2311	1.3623	1.8317

In the regular case, $(D-\theta)^{-1}$ remains bounded as θ approaches λ, the desired eigenvalue of N becomes zero, and the corresponding eigenvector of N converges to the desired eigenvector of A. Thus the method does converge to the desired eigenvector with a better (and perhaps much better) convergence rate. In Example 1 the gap ratio of the desired eigenvalue (λ_1) for the original matrix A is

$$(\lambda_2-\lambda_1)/(\lambda_{20}-\lambda_2) = (1.854160-.222851)/(20.41594-1.866517)$$
$$= .087943.$$

For different values of θ, Table 4.2 gives the smallest eigenvalue of $N(\theta)$, its gap ratio, and the sine of the angle between the corresponding eigenvector of $N(\theta)$ and the smallest eigenvector of A.

TABLE 4.2
Gap ratios and sines for different values of θ.

θ	Eigenvalue of N	Gap ratio	Sine
0.9	−2.16723	0.722	0.3776
0.5	−0.30592	0.499	0.1010
0.3	−0.06670	0.447	0.0246
0.22	0.00227	0.431	0.0009
0.2	0.01788	0.427	0.0069

It is clear from Table 4.2 that the effectiveness of N is not very sensitive to changes in θ. The gap ratio for N for θ anywhere near λ_1 is fives times the gap ratio for A. Since the convergence rate depends exponentially on the gap ratio, this change makes an enormous difference in the convergence rate of the algorithm. Furthermore the eigenvector is hardly perturbed. Thus rapid local convergence is to be expected. The first value of θ encountered in Example 1 is 3.23529. This lies between the third eigenvalue (2.95594) and the fourth eigenvalue (3.99522) of A. Some of the eigenvalues of $N(3.23529)$ are complex (which is not disastrous by itself) but the desired eigenvector is not well represented in the extreme eigenvalues of N. This explains why the first few Davidson steps in Example 1 were not as effective as the first few Lanczos steps. Davidson's method was happily trying to converge to the wrong eigenvector. It is only after θ is closer to the desired eigenvalue than to any of the others that the method starts converging rapidly.

5. Generalizing Davidson's method. Unfortunately Davidson's method does not always increase the gap ratio. If the diagonal of a matrix A is constant, then Davidson's method reduces to the Lanczos algorithm. If the diagonal is almost constant then Davidson's method may be slightly faster than Lanczos but it will probably not be worth the higher overhead. This was the conclusion of Kalamboukis [4] when he investigated using Davidson's method on nuclear modeling problems.

Interpreting the operator $(D-\theta_{j-1})^{-1}$ as a preconditioner for $(A-\theta_{j-1})$, the obvious way to improve Davidson's method is to use a better preconditioner. The generalized algorithm requires modification of only step one of the original algorithm as follows:

$$1.\ \text{Compute } p_j = (M-\theta_{j-1})^{-1} r_{j-1},$$

where $M-\theta_{j-1}$ is some easily inverted approximation to $(A-\theta_{j-1})$. One potential advantage of this use of preconditioners over conjugate gradients is that here there is no requirement that the preconditioner be positive definite. This allows the preconditioner to more closely approximate the indefinite matrix $(A-\sigma)$. In what follows the generalized algorithm will be referred to as the GD algorithm.

As before, the effectiveness of the GD algorithm depends on the nature of the operator $N=(M-\theta)^{-1}(A-\theta)$. If $M=A$ then N is the identity matrix and the method breaks down (just as ordinary Davidson's method fails when A is a diagonal matrix). Otherwise the same two conditions on N for Davidson's method must also be satisfied here. θ is chosen so that the desired eigenvalue of $A-\theta$ is near zero. This in turn means that the preconditioning is likely to compress this eigenvalue less than the others and leave it better separated than before. It is this tendency which makes the method successful.

The GD algorithm was applied to Example 1 using the preconditioner $(T-\theta)^{-1}$ where T is the tridiagonal part of A. Table 5.1 gives the sequence of eigenvalue approximations obtained by the GD algorithm (starting with the same vector).

TABLE 5.1
Behavior of the GD *algorithm on Example* 1.

Step	θ	Residual norm
1	3.23529	5.274
2	2.58389	3.777
3	1.54362	1.286
4	1.49082	1.121
5	.38969	1.024
6	.22286	.0151
7	.22285	.1e-7
8	.22285	.6e-13

As can be seen from Table 5.1, the first four steps are very similar to the behavior of the original algorithm with convergence to the wrong eigenvalues. Once θ is closer to the desired eigenvalue than to the rest of the spectrum, convergence is almost immediate. As before the behavior of the algorithm can be understood by examining the spectrum of $N(\theta)=(T-\theta)^{-1}(A-\theta)$ for values of θ near the desired eigenvalue.

TABLE 5.2
Gap ratios for different values of θ *for* GD.

θ	Eigenvalue of N	Gap ratio	Sine
.9	1.0+.171i	1.0	.4761
.2	0.2388	1.0	.0165
.222846097	.494e-7	1.0	.839e-6

As before, for all values of θ near the desired eigenvalue there is a well separated eigenvalue of $N(\theta)$ with a corresponding eigenvector which is nearly parallel to the desired eigenvector of A. The extremely rapid convergence obtained with the tridiagonal preconditioner is due to more than just the increased gap ratio. For all values of θ near the desired eigenvalue, the spectrum of $N(\theta)$ with tridiagonal preconditioning consists of a cluster of 18 eigenvalues within 1e-16 of 1 and two other eigenvalues symmetrically located around 1. This distribution guarantees convergence after only 3 steps.

6. Global convergence. As seen above, both Davidson's method and GD do not converge rapidly to the desired eigenvalue as long as θ is far away. If the desired eigenvalue is specified as the eigenvalue of A closest to a given number σ, then it is possible to modify the algorithm to improve the global convergence. Instead of using $(M-\theta)^{-1}$ as the preconditioner, $(M-\sigma)^{-1}$ can be used until θ has started converging to the desired eigenvalue. In these examples the switch from σ to θ was made when the residual norm bounded θ away from σ. Tables 6.1, 6.2, and 6.3 give the behavior, for different values of σ, of both diagonal preconditioning and tridiagonal preconditioning on Example 1 when the preconditioners are fixed at σ until θ has settled down. As can be seen from the tables, this modification significantly improves the global convergence of the algorithm and does not require a particularly accurate value for σ.

TABLE 6.1
$\sigma = .9$.

	Diagonal		Tridiagonal	
Step	θ	Residual norm	θ	Residual norm
1	3.2352	5.2740	3.2352	5.2740
2	.9007	1.3130	.5190	1.5320
3	.3321*	.5493	.2276*	.2001
4	.2347	.2184	.2229	.0331
5	.2236	.0613	.2228	.0002
6	.2229	.0130	.2228	.3813e-11

* Switch from σ to θ.

TABLE 6.2
$\sigma = .5$.

	Diagonal		Tridiagonal	
Step	θ	Residual norm	θ	Residual norm
1	3.2352	5.2740	3.2352	5.2740
2	.7455	1.1730	.2911	.9275
3	.3055	.4406	.2229*	.0168
4	.2318*	.1978	.2228	.0022
5	.2234	.0494	.2228	.1154e-5
6	.2229	.0117	.2228	.3836e-13

* Switch from σ to θ.

TABLE 6.3
$\sigma = .2$.

	Diagonal		Tridiagonal	
Step	θ	Residual norm	θ	Residual norm
1	3.2352	5.2740	3.2352	5.2740
2	.7054	1.1160	.2493	.7077
3	.2987	.4254	.2230	.0294
4	.2308	.1854	.2228*	.7790e-4
5	.2233	.0462	.2228	.2244e-7
6	.2228*	.0109	.2228	.4518e-13

7. Conclusions. The success of Davidson's method on some types of eigenvalue problems shows the potential power of diagonal preconditioning. Generalizing Davidson's method allows for more powerful preconditioners to be used which makes the method effective for a much wider class of matrices. Using $(M-\sigma)^{-1}$ instead of $(M-\theta)^{-1}$ as the preconditioner shows that it is possible to force global convergence to a particular eigenvalue. If the formation of a matrix-vector product is very cheap then the overhead required in the GD algorithm will not be cost effective, but if the matrix-vector product is expensive then the GD algorithm will significantly reduce the number of matrix-vector products required and thus will be significantly cheaper than the Lanczos algorithm.

Acknowledgments. The second author would like to thank Beresford Parlett for helpful discussions during early phases of this work. We would both like to thank the referee for many helpful comments.

REFERENCES

[1] O. AXELSSON, *On preconditioning and convergence acceleration in sparse matrix problems*, Technical Report CERN 74-10, CERN, Geneva, Switzerland, 1974.

[2] E. R. DAVIDSON, *The iterative calculation of a few of the lowest eigenvalues and corresponding eigenvectors of large real-symmetric matrices*, J. Comp. Phys., 17 (1975), pp. 87-94.

[3] I. GUSTAFSSON, *A class of first order factorization methods*, BIT, 18 (1978), pp. 142-156.

[4] T. Z. KALAMBOUKIS, *Davidson's Algorithm with and without perturbation corrections*, J. Phys. A: Math. Gen., 13 (1980), pp. 57-62.

[5] J. A. MEIJERINK AND H. A. VAN DER VORST, *An iterative solution method for linear systems of which the coefficient matrix is a symmetric M-matrix*, Math. Comp., 31 (1977), pp. 148-162.

[6] C. C. PAIGE AND M. A. SAUNDERS, *Solution of sparse indefinite systems of linear equations*, SIAM J. Numer. Anal., 12 (1975), pp. 617-629.

[7] B. N. PARLETT, *The Symmetric Eigenvalue Problem*, Prentice-Hall, Englewood Cliffs, NJ, 1980.

[8] H. L. STONE, *Iterative solution of implicit approximations of multidimensional partial differential equations*, SIAM J. Numer. Anal., 5 (1968), pp. 530-558.

[9] D. B. SZYLD, *A two-level iterative method for large sparse generalized eigenvalue calculations*, Ph.D. thesis, Courant Institute, New York Univ., New York, October, 1983.

SIAM J. SCI. STAT. COMPUT.
Vol. 7, No. 3, July 1986

008

LINEAR LEAST SQUARES WITH BOUNDS AND LINEAR CONSTRAINTS*

RICHARD J. HANSON†

Abstract. An algorithm is given for solving linear least squares systems of algebraic equations subject to simple bounds on the unknowns and (more general) linear equality and inequality constraints.

The method used is a penalty function approach wherein the linear constraints are (effectively) heavily weighted. The resulting system is then solved as an ordinary bounded least squares system except for some important numerical and algorithmic details.

This report is a revision of an earlier work. It reflects some hard-won experience gained while using the resulting software to solve nonlinear constrained least squares problems.

Key words. linear least squares, linear systems, bounds, equality constraints, inequality constraints

1. Introduction.

Solving the least squares algebraic system

$$A\mathbf{x} \cong \mathbf{b}, \qquad A_{\text{MROWS by NCOLS}}, \tag{1}$$

is fundamental in the solution of many problems in applied mathematics. Excellent numerical methods exist for solving (1), such as Householder's method (employing elementary orthogonal matrices), [1, pp. 9-10], Givens' method (employing plane rotations), [1, pp. 10-11], and the singular value decomposition, [1, pp. 18-21].

Frequently the solution of the system of (1) has additional requirements or constraints that might derive from physical meanings for the unknowns, $\mathbf{x}$. Here we consider a special class of constraints which we write in the form of simple bounds

$$\alpha_j \leqq x_j \leqq \beta_j, \qquad j = 1, \cdots, \text{NCOLS},$$

and linear equality and inequality constraints $\mathbf{c}_i^T\mathbf{x} = f_i$, $\mathbf{c}_i^T\mathbf{x} \geqq g_i$, $\mathbf{c}_i^T\mathbf{x} \leqq h_i$ or $g_i \leqq \mathbf{c}_i^T\mathbf{x} \leqq h_i$. We write these general linear constraints in the compact form $\mathbf{y} = C\mathbf{x}$, $C_{\text{MCON by NCOLS}}$, with bounds on the components y_i:

$$\alpha_i' \leqq y_i \leqq \beta_i', \qquad i = 1, \cdots, \text{MCON}.$$

(For example, the constraints $\mathbf{c}_i^T\mathbf{x} = f_i$ and $\mathbf{c}_j^T\mathbf{x} \geqq g_j$ amount to the respective choices $\alpha_i' = \beta_i' = f_i$ and $\alpha_j' = g_j$, $\beta_j' = +\infty$.) This method of writing the linear constraints is more concise than describing each constraint individually. It also provides a useful framework for replacing an infeasible problem (in which no values of $\mathbf{x}$ exist that satisfy all constraints) by a nearby feasible problem. We summarize the statement of the linearly constrained problem:

Problem BNDCLS. Solve

$$A\mathbf{x} \cong \mathbf{b}, \qquad A_{\text{MROWS by NCOLS}},$$

subject to constraints in the form of simple bounds

$$\alpha_j \leqq x_j \leqq \beta_j, \qquad j = 1, \cdots, \text{NCOLS},$$

and linear constraints

$$C\mathbf{x} = \mathbf{y}, \qquad C_{\text{MCON by NCOLS}},$$

subject to simple bounds

$$\alpha_i' \leqq y_i \leqq \beta_i', \qquad i = 1, \cdots, \text{MCON}. \tag{2}$$

The simple bounds on the x_j are a special type of linear constraint that could be included in the constraints $C\mathbf{x} = \mathbf{y}$. For reasons of computational efficiency and economy of storage it is important to consider such constraints separately.

* Received by the editors April 12, 1983, and in final form September 20, 1984.

† Applied Mathematics Division, Sandia National Laboratories, Albuquerque, New Mexico 87185. This work was supported by the U.S. Department of Energy.

Other related algorithmic work on the linearly constrained least squares problem is that of [7] based on the work of Stoer and Schittkowski [8]. A quite different approach is taken in our work—a penalty function method (also see [8]) as opposed to a feasible direction method. The work of Zimmermann [6] is concerned with the least squares problem constrained by simple bounds only. There is a restrictive assumption in [6] that the least squares matrix is of full column rank. This restriction is apparently present in the work of [7] also, but it is not explicitly stated. The present work has no such restriction.

This paper gives some of the required background for the development of a robust computer program to solve Problem BNDCLS. The software package developed by the author will be published elsewhere.

An early version of the software had two deficiencies. One was numerical and another was conceptual. The numerical deficiency involved translations of bounds. At first glance, it seems reasonable to normalize the bounds of Problem BNDCLS so that variables are translated and possibly reflected so that all lower bounds are zero. Numerically, this can be a disaster. Consider an example with a single least squares equation $x = a$ and bounds $-b \leqq x \leqq b$. Assume that the quantity b is large relative to a, so that $b + a$ is computed as b. If we translate by letting $y - b = x$, the working equation for y is $y = b$. All dependence on a is now gone, so the solution for x can have no accuracy at all. Thus the algorithm given in § 3 avoids translating variables that include the value zero as an interior point in their bounds. This fact also complicates the algorithm because there is a need to distinguish variables which have hit a bound during the course of the solution process and those that have not hit a bound.

The conceptual deficiency arose because it seemed important initially to note whether general linear constraints are inconsistent. In fact, the subroutine package of [4] contains a code *LSEI* that solves a constrained least squares problem with linear constraints. If the problem is infeasible, only an error flag is returned. That seems to be an undesirable feature in the design of the software. Suppose, for example, that two constraints $\sum_{j=1}^{\mathrm{NCOLS}} x_j = 1$ and $\sum_{j=1}^{\mathrm{NCOLS}} x_j \leqq 0.99$, $(x_j \geqq 0)$, are specified by a user. That user may not want to distinguish between 0.99 and 1.00, but the code *LSEI* surely will. This, and other reasons not stated here, forced the author to take a fresh approach regarding the inconsistent linearly constrained problem. In fact, the method presented below for resolving such constraints will replace the above example by the feasible region $\sum_{j=1}^{\mathrm{NCOLS}} x_j \leqq 0.995$, $0.995 \leqq \sum_{j=1}^{\mathrm{NCOLS}} x_j \leqq 1.0$, $(x_j \geqq 0)$.

There are two important differences between the present work and the work of [3] and its published algorithm [4]. The most significant one is the new capability for upper and lower bounds without writing these bounds as rows of C. This new capability combines conveniently with the technique of expressing the other linear constraints as $C\mathbf{x} = \mathbf{y}$, where bounds are present on the components $\mathbf{y}$. The second primary difference is the technique for dealing with infeasible constraints.

2. Reduction of the linearly constrained problem to a bounded least squares problem. In this section we consider the general case where the number of linear constraints, MCON > 0. The methods presented here are based on a two-phased approach.

Phase 1. Solve the constraint equation $C\mathbf{x} - \mathbf{y} = \mathbf{0}$ in a least squares sense subject to the bounds on $\mathbf{x}$ and $\mathbf{y}$ of (2).

In case the constraints are infeasible we "cooperatively" make them feasible by perturbing values of α_i' and β_i'. Specifically we compute $\hat{\mathbf{y}} = C\hat{\mathbf{x}}$, where $\hat{\mathbf{x}}$ is the first NCOLS components of the solution vector $(\mathbf{x}^T, \mathbf{y}^T)^T$. Then we enlarge the bounds on the y_i:

$$\alpha_i'' = \min(\alpha_i', \hat{y}_i), \qquad \beta_i'' = \max(\beta_i', \hat{y}_i).$$

Note that $\alpha_i'' = \alpha_i'$ and $\beta_i'' = \beta_i'$, $i = 1, \cdots,$ MCON, if the constraints are feasible. In the alternate (infeasible) case this algorithm provides one particular solution to the problem of enlarging the bounds on $\mathbf{y}$ so that solutions to the equations $C\mathbf{x} - \mathbf{y} = \mathbf{0}$ will exist. One could solve this least squares system for its unique minimum length solution and then enlarge the bounds on $\mathbf{y}$ as described. Obtaining this minimum length solution could be done as in [3]. The simpler method outlined above is often to be preferred because the computation of the minimum length solution is relatively expensive. Also, uniqueness of the new constraint region is not an issue for many applications.

It also seems that the approximate solution of linear constraints, as shown here, can be motivated just as the approximate solutions of linear algebraic equations. For example, the least squares solution of a linear system can be regarded as replacing the right-hand side vector by its projection into the column space of the matrix.

Phase 2. Solve the weighted least squares problem

$$\begin{bmatrix} C & -I \\ \tau A & 0 \end{bmatrix} \begin{bmatrix} \mathbf{x} \\ \mathbf{y} \end{bmatrix} = \begin{bmatrix} \mathbf{0} \\ \tau \mathbf{b} \end{bmatrix} \tag{3}$$

with bounds

$$\alpha_j \leqq x_j \leqq \beta_j, \qquad j = 1, \cdots, \text{NCOLS},$$
$$\alpha_j'' \leqq y_i \leqq \beta_i'', \qquad i = 1, \cdots, \text{MCON}.$$

The small weighting parameter τ is chosen so that the scaling of the resulting problem is balanced:

$$\tau = \begin{cases} \eta(\|C\|/\|A\|), & C \neq 0 \text{ and } A \neq 0, \\ \eta(1/\|A\|), & C = 0 \text{ and } A \neq 0, \\ 1, & \text{if } A = 0. \end{cases}$$

Here η = relative arithmetic precision, i.e., the largest machine number such that $1 + \eta$ has the value 1. The use of η in the definition of τ is arbitrary; in fact, τ has no positive lower bound. An upper bound is (effectively) $\eta^{1/2}$. See [1, Chap. 22] for further details, and also [3].

Use of the penalty function was made for two reasons. The first reason involved the fact that the problem with simple bounds but without general linear constraints is straightforward. In fact, C. L. Lawson developed a code (unpublished) about 1976 for this problem. With simple bounds, feasibility of the bounds is resolved simply by checking whether $\alpha_j \leqq \beta_j$. With general linear constraints the issue of feasibility is more complicated. The author chose to bypass this question by the preprocessing steps previously outlined. Thanks to this preprocessing step the constraints are made consistent. The second reason the penalty function was used is that it has been known for some time [1, pp. 148–149] that penalty or weighting methods for equality constrained linear least squares (no simple bounds present) are effective. Existing least squares algorithms can often be used as long as row and column pivoting is used. One must avoid mixing weighted and nonweighted rows in such a way that essential information about the problem is lost. Because penalty function methods work well in this context and existing algorithms had been developed for the case of simple bounds, the author was led to investigate the combined approach to solving Problems BNDCLS presented below.

It is helpful to precede the detailed description of algorithm NNLSB of § 3 with an informal discussion of how a solution is computed.

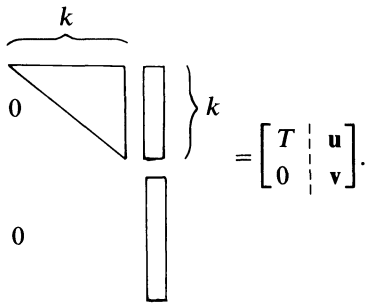
The basic idea amounts to a systematic choice of subsets of the columns of A. Using these columns, an unconstrained least squares solution is computed by triangularizing the matrix with an orthogonal decomposition, [1, pp. 5-7]. The variables not in the subset are either at their bounds or at the value zero. The choice of the subset (called B in the algorithm) is made so that the corresponding unconstrained least squares residual vector length is decreasing with each step. (Systematic examination of *all* subsets is out of the question for anything but small problems.) To be robust, the algorithm for solving the unconstrained least squares problem must test for linear independence of the columns of that subset.

Our suggested test for linear independence uses a norm that effectively removes the small weight applied to rows MCON+1, $\cdots$, MCON+MROWS of the matrix in (3). This weighted norm is defined as

$$\|\cdot\| \equiv \underbrace{[(\cdot)^2+\cdots+(\cdot)^2}_{\text{MCON}} + \underbrace{\tau^{-2}(\cdot)^2+\cdots+\tau^{-2}(\cdot)^2]^{1/2}}_{\text{MROWS}} .$$

The columns of A corresponding to indices in set B are triangularized. The variables in B are chosen so that the columns are linearly independent and the unconstrained least squares problem (with these columns) yields a solution that satisfies the bounds in the strict sense.

In the choice of a new subset, a candidate column is adjoined to this upper triangle to form a new partitioned matrix:



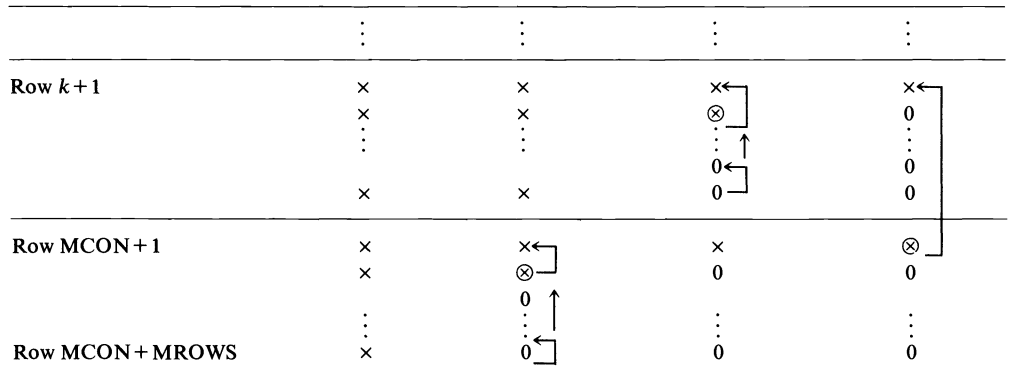
Using the weighted norm, we define the new columns as *linearly dependent* with respect to a parameter ε_1 if $\|v\| \leqq \varepsilon_1 \|u\|$. (The value of the parameter ε_1 can be varied by the user of the software; $\varepsilon_1 = \eta^{1/2}$ is the default value.) Otherwise the candidate column $[\mathbf{u}^T \,\vdots\, \mathbf{v}^T]^T$ is linearly independent. Mathematically the test for linear dependence of the candidate column is simply "$\|\mathbf{v}\| = 0$." In computations one replaces this test by "$\|\mathbf{v}\|$ is small." A particular (inexpensive but reliable) choice is the one given above. Note that this test is independent of column scaling of the data matrix. The test requires that the weighted norm be used because a norm with unit weights would always imply that $\|\mathbf{v}\|$ was small whenever $k \geqq$ MCON. This is due to the fact that the small weight τ applied to the least squares equations in (3) persists throughout the algorithm.

Completing the triangularization of the new column is done by means of plane rotations to eliminate the entries below the main diagonal. A second important implementation detail associated with the use of the small weight τ involves a type of *pivoting* step when eliminating (to entry $k+1$) the components of column $k+1$ in any of the entries $k+2, \cdots,$ MCON+MROWS. One must avoid a numerically unstable situation where an elimination is performed on two entries $\binom{u_i}{v_j}$ where u_i is small (relative to the norm of the column) and v_j corresponds to a row weighted by τ. This would yield an essentially random plane rotation that mixes the weighted and unweighted

rows and might result in the loss of essential information about the problem. The pivoting step is accomplished, in our implementation, by eliminating the entries in the order MCON+MROWS, $\cdots$, MCON+1; MCON, $\cdots$, $k+2$; and finally eliminating between the planes MCON+1 and $k+1$. In order to guarantee that entry $k+1$ is not small, prior column interchanges are performed (the pivoting choice is discussed in [3, p. 103]). Table 1 may help the reader follow the sequence of eliminations. For $k \geqq$ MCON only the "first sweep" is performed. For $k <$ MCON all the steps are required.

We close this section with comments about the working storage required for this algorithm. To highest order terms MCON$\times$(MCON+MROWS) additional memory locations are needed beyond that required for the data matrix $[C^T : A^T]^T$. There are options provided by the software so that the matrix A can be pretriangularized, a few rows at a time, using the process of sequential accumulation [1, p. 210]. Thus the user can organize the computation so that (in terms of storage required), MROWS $\doteq$ NCOLS. This is particularly important when the number of rows is large compared to the number of variables. Since we solve a bounded variable least squares problem followed by a computation of a reachable point $\hat{\mathbf{y}} = C\hat{\mathbf{x}}$, it is necessary to store (or regenerate) the matrix C. In fact (the transpose of) C is stored in the space indicated by the block of zeros in (3) before any processing is performed. This block of zeros may fill in completely after the eliminations are performed. The point of this discussion is that the amount of extra storage can be kept near MCON$\times$(MCON+NCOLS); it does not have to depend on MROWS.

TABLE 1



3. An algorithm for the heavily weighted least squares problem with simple bounds. In this section we present a method for solving Problem BNDCLS of (1)-(2), by performing the computations described in Phase 1 and Phase 2 of § 2. The core algorithm is, therefore, a linear least squares problem with simple bounds on the variables, $\alpha_j \leqq x_j \leqq \beta_j$, $j = 1, \cdots,$ NCOLS. The algorithm must be supplied with the value of MCON together with the data for the weighted least squares problem.

The user of the software specifies bounds for each variable according to one of the four cases of (2). If the variable is classified Case 1, 2 or 3, it is *restricted.*

Case 1. $\alpha_j \leqq x_j$, $\alpha_j > -\infty$.

Case 2. $x_j \leqq \beta_j$, $\beta_j < +\infty$.

Case 3. $\alpha_j \leqq x_j \leqq \beta_j, \quad -\infty < \alpha_j \leqq \beta_j < +\infty.$

Case 4. x_j is *unrestricted* or *free.*

By adjoining missing bounds to Cases 1, 2 and 4, we normalize the problem so that for each j, $-b \leqq \alpha_j \leqq x_j \leqq \beta_j \leqq b$, where b = largest floating point number in the machine.

We further normalize the problem so that if $\alpha_j < 0 < \beta_j$, then $|\alpha_j| \leqq \beta_j$, or $0 \leqq \alpha_j \leqq \beta_j$. (This may require a change of sign for x_j. The normalization step may change the values of α_j and β_j, but this should present no confusion for the reader.) These conditions partition the variables into two disjoint sets L and T: $j \in T$ if $0 \leqq \alpha_j \leqq \beta_j$; otherwise $j \in L$. The sets L and T change during the algorithm. The elements of L move to T whenever a variable moves to a bound. One can think of T as designating those variables that have ever "touched" a bound. The variables in L are those that have remained between bounds.

For those variables in T we translate the lower bound so that it becomes zero, $x'_j = x_j - \alpha_j$. (The normalization $0 \leqq \alpha_j \leqq \beta_j$, $j \in T$, makes this numerically safe to do.)

Note that column scaling $x_j = d_j w_j$, $d_j > 0$, does not alter the partitioning $\{T, L\}$. It does affect the choice of the variable that is chosen to become (possibly) unconstrained in Steps 4-5 of Algorithm NNLSB. It can also affect the particular solution chosen by the algorithm when the data matrix has rank less than NCOLS. Our software package performs column scaling so that the maximum magnitude norm of nonzero columns has the value one. An option is provided so that the user can override our particular choice of scaling. Steps 1-19 of the algorithm are comprised of a main loop or outer iteration consisting of Steps 2-16. This loop contains two inner loops 4-7 and 14-15. The algorithm begins with the feasible point $\mathbf{x} = \mathbf{0}$. In the main loop, elements of the set B are dropped and added in such a way that the residual vector length $\|\mathbf{b} - A\mathbf{x}\|$ is decreasing (in the weak sense). The proof of finite convergence of the algorithm consists of noting that the number of iterations between a strict decrease in the residual vector length, or termination of the algorithm, is at most NCOLS.

The integers $I = \{1, \cdots, \text{NCOLS}\}$ are naturally partitioned by the algorithm into two pairs of disjoint sets $\{Z, B\}$ and $\{T, L\}$. The partitioning $\{Z, B\}$ is determined by whether or not a variable is at the value zero. The array IP(*) effectively contains counts of the number of times a variable has hit an upper bound. This information is used in Step 3 to determine if allowing a variable to decrease from an upper bound would achieve a decrease in the residual norm. It is also required to reflect variables from an upper bound when computing the solution in Steps 17-19. In summary, the algorithm is a feasible direction, descent method. In fact, Steps 10-12 amount to choosing a step length and defining a new feasible point in this descent direction, and Steps 14-15 involve updating the Set B so that the corresponding unconstrained least squares problem has a solution that satisfies the bounds but is not at a bound.

ALGORITHM NNLSB.

1. Set $Z := \{1, \cdots, \text{NCOLS}\}$, $B :=$ NULL, $\mathbf{x} := \mathbf{0}$, $\text{IP}(j) := 1$, $j = 1, \cdots, \text{NCOLS}$.
2. Compute the NCOLS vector $\mathbf{w} = A^T(b - A\mathbf{x})$.
3. For $j = 1, \cdots, \text{NCOLS}$ set $w_j := -w_j$ if $\text{IP}(j) \equiv 0 \pmod 2$.
4. Find an index $t \in Z$ such that $w_t = \max\{w_j : j \in Z \cap T,\ w_j > 0,\ x_j \text{ is restricted}\}$.
5. Find an index $s \in Z$ such that $|w_s| = \max\{|w_j| : j \in Z \cap L\}$.
6. Find the larger of w_t, $|w_s|$ and the corresponding subscript u. (If neither t nor s is defined by Steps 4 and 5, exit to Step 17.).
7. If column u is linearly independent of the columns in set B then move the index u from set Z to set B. Otherwise set $w_u := 0$ and go to Step 4.
8. Let $A_B = [\hat{\mathbf{a}}_1, \cdots, \hat{\mathbf{a}}_{\text{NCOLS}}]$, where $\hat{\mathbf{a}}_j =$ column j of A if $j \in B$; $\hat{\mathbf{a}}_j = \mathbf{0}$ otherwise.

9. Compute the NCOLS vector $\mathbf{z}$ as the least squares solution of the system $A_B\mathbf{z} \cong \mathbf{b}$. (Only z_j, $j \in B$, are defined; define $z_j = 0$ for $j \in Z$.)
10. Set $\alpha := 1$, $\beta := 1$.
11. For each j define $\gamma_j = 0$ if $j \in T$. Otherwise $\gamma_j = \alpha_j$. If $z_j \leqq \gamma_j$ for $j \in B$, find an index $q \in B$ such that $\alpha = (x_q - \gamma_q)/(x_q - z_q) = \min((x_j - \gamma_j)/(x_j - z_j)\colon j \in B,\ z_j \leqq \gamma_j)$.
12. If $z_j \geqq \beta_j$ for some $j \in B$, then find an index $p \in B$ such that $\beta = (\beta_p - x_p)/(z_p - x_p) = \min((\beta_j - x_j)/(z_j - x_j)\colon j \in B,\ z_j \geqq \beta_j)$.
13. Update the vector $\mathbf{x} := \mathbf{x} + \min(\alpha, \beta)(\mathbf{z} - \mathbf{x})$.
14. Move from set B to set Z all indices j such that $x_j \leqq \gamma_j$. If $j \in L$, then update the bound $\beta_j := \beta_j - \alpha_j$ and move j from set L to set T.
15. Move from set B to set Z all indices j such that $x_j \geqq \beta_j$. If $j \in T$, reflect the variables by the update step $x_j := \beta_j - x_j$, and update the polarity of these x_j, $\mathrm{IP}(j) := \mathrm{IP}(j) + 1$. Otherwise change the sign of x_j, $\beta_j' := \beta_j - \alpha_j$, $\alpha_j := -\beta_j$, and move j from set L to set T.
16. Loop on Step 2.
17. For $j = 1, \cdots,$ NCOLS, set $x_j := \beta_j - x_j$ if $\mathrm{IP}(j) \equiv 0 \pmod 2$.
18. For $j = 1, \cdots,$ NCOLS, set $x_j := x_j + \alpha_j$ if $j \in T$.
19. For $j = 1, \cdots,$ NCOLS, set $x_j := -x_j$ for each j corresponding to a variable with a sign changed during the algorithm.

Remarks on Algorithm NNLSB.

The algorithm converges. This is proved by noting that the residual norm $\|\mathbf{b} - \mathbf{A}x\|$ is weakly monotone decreasing, with respect to the number of iterations of the loop between Steps 2 and 16. As in the discussion of Algorithm NNLS [1, Chap. 23], the norm strictly decreases or the Set B loses members at each iteration. Suppose the Set B becomes empty. At that point, if a column is moved from Set Z to Set B in Steps 4–5, then the residual norm strictly decreases, because $\min(\alpha, \beta) > 0$ in step 13, or $\min(\alpha, \beta) = 0$. From Step 11 and the fact that $\alpha > 0$, it follows that $\min(\alpha, \beta) = 0$ implies that $\beta = 0$. In that case the corresponding $w_j > 0$ chosen in Step 4 will now be negative and thus will not be chosen again in Step 4 while $\beta = 0$. Thus, in this most extreme case, the number of positive w_j decreases. This completes the proof of finite convergence of the algorithm.

At Steps 14 and 15, the variables should mathematically satisfy the bounds $x_j \geqq \gamma_j$ and $x_j \leqq \beta_j$ respectively. Due to the finite precision arithmetic, there may be an x_j that violates the bounds and it is essential that such indices be moved from Set B to Set Z. (The index corresponding to the choice of α or β in Steps 11–12 is always so moved.) There remains the prospect that variables satisfying $x_j > \gamma_j$ or $x_j < \beta_j$ are misclassified; they should really be classified as $x_j = \alpha_j$ or $x_j = \beta_j$. Fortunately this is not serious due to the fact that members of Set B satisfy $\alpha_j < x_j < \beta_j$ at Step 2 because of the action taken in Steps 14 and 15. Thus $x_j - z_j > 0$ in step 11 and $z_j - x_j > 0$ in Step 12 at later outer iterations of the algorithm. This implies that misclassified variables will be ultimately moved from Set B to Set Z if required.

The test for linear independence in Step 7 plays a crucial role in the success of the algorithm. This matter was more thoroughly considered in § 2.

The least squares problem of Step 9 is solved by applying plane rotations (Givens transformations) to the data matrix, $[A : \mathbf{b}]$. These transformations are applied in a certain order discussed in § 2. Those columns corresponding to indices in B are moved to the left. Thus when a new column is added to B the processed matrix A_B is upper triangular with a spike in the last column. Plane rotations are then applied to reduce this matrix to upper triangular form. When indices are dropped from B in steps 14 and 15 the resulting matrix A_B is upper Hessenberg, [2, p. 23]. This matrix is again

reduced to upper triangular form using plane rotations. When eliminating the entry between rows MCON and MCON+1, column pivoting may be required to avoid mixing the weighted and nonweighted rows. This pivoting step may result in the loss of upper Hessenberg form, so that reducing the system to triangular form once again requires triangularizing a full rectangular matrix, since only premultiplying plane rotations are allowed in this algorithm. (This large amount of work is the price for numerical stability; fortunately, this seems to occur rarely.)

An important computational matter, especially for the problem of general linear constraints, is the selection of the order of the planes of rotation. The specifics were given in § 2.

4. An example. A problem considered in [5] and [6] has problem dimensions

$$\text{NCOLS} = 5,$$

$$\text{MCON} = 4,$$

$$\text{MROWS} = 6.$$

This problem is used as a test case here because it is easy to describe and has been treated by others. The general linear constraints were used in [5] and the bounds were used in [6]. Here we have both the linear constraints and the bounds in our problem statement. The constraint matrix is given by

$$C = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 10 & 10 & -3 & 5 & 4 \\ 4 & 2 & -3 & 5 & -1 \\ 8 & -1 & 2 & 5 & 3 \end{bmatrix}.$$

The bounds on $\mathbf{y} = C\mathbf{x}$ are $y_1 \leqq 5$, $y_2 \geqq 2$, $y_3 \leqq 30$, $11 \leqq y_4 \leqq 30$. (Note that bounds on y_4 require two inequalities in [5].) The least squares matrix and two separate right-hand side vectors are

$$A = \begin{bmatrix} -74 & 80 & 18 & -11 & -4 \\ 14 & -69 & 21 & 28 & 0 \\ 66 & -72 & -5 & 7 & 1 \\ -12 & 66 & -30 & -23 & 3 \\ 3 & 8 & -7 & -4 & 1 \\ 4 & -12 & 4 & 4 & 0 \end{bmatrix}, \quad \mathbf{b}_1 = \begin{bmatrix} 51 \\ -61 \\ -56 \\ 69 \\ 10 \\ -12 \end{bmatrix}, \quad \mathbf{b}_2 = \begin{bmatrix} -5 \\ -9 \\ 708 \\ 4{,}165 \\ -13{,}266 \\ 8{,}409 \end{bmatrix}.$$

We also use two sets of upper and lower bounds, from the work of [6], on the components of $\mathbf{x}$:

$$\alpha_1^{(1)} = -1, \quad \alpha_2^{(1)} = 0, \quad \alpha_3^{(1)} = -3, \quad \alpha_4^{(1)} = 1, \quad \alpha_5^{(1)} = -6,$$

$$\beta_1^{(1)} = 3, \quad \beta_2^{(1)} = 4, \quad \beta_3^{(1)} = 1, \quad \beta_4^{(1)} = 5, \quad \beta_5^{(1)} = -2,$$

$$\alpha_1^{(2)} = 1, \quad \alpha_2^{(2)} = 0, \quad \alpha_3^{(2)} = -1, \quad \alpha_4^{(2)} = 1, \quad \alpha_5^{(2)} = -4,$$

$$\beta_1^{(2)} = 3, \quad \beta_2^{(2)} = 2, \quad \beta_3^{(2)} = 1, \quad \beta_4^{(2)} = 3, \quad \beta_5^{(2)} = -2.$$

In each of the four problems with bounds $\alpha^{(l)} \leqq x \leqq \beta^{(l)}$, $l = 1, 2$, and right-hand sides $\mathbf{b}_m$, $m = 1, 2$, the true solution is $\hat{\mathbf{x}} = (1, 2, -1, 3, -4)^T$, $\hat{\mathbf{y}} = (1, 32, 30, 31)^T$. Table 2 shows the accuracy obtained with each of the four problems on a CDC CYBER 172/855 computer using $\eta \doteq 7.1 \times 10^{-15}$ relative precision. Table 3 shows the computed residual vector length.

TABLE 2
Maximum error vector length for Stoer-Zimmermann problems.

$l=$	1	2
$m=1$	2.0×10^{-12}	2.5×10^{-11}
2	2.7×10^{-10}	4.2×10^{-9}

TABLE 3
Residual vector length for Stoer-Zimmermann problems.

$l=$	1	2
$m=1$	9.7×10^{-13}	2.3×10^{-13}
2	$1.6264444933658\times10^{4}$	*ditto* ←

To illustrate a result where the constraints are infeasible we (arbitrarily) added an additional constraint that is equivalent to $y_1 \geqq 6$. Then the minimum residual vector length for the constraints is $2^{1/2}$. The residual vector length for the least squares equations is 83.170006612968 and an approximate solution is

$$\hat{\mathbf{x}} = (1.5,\ 2,\ 1,\ 3,\ -2)_T, \qquad \hat{\mathbf{y}} = (5.5,\ 39,\ 24,\ 33,\ 5.5)^T.$$

Thus the first and last of the constraints have been resolved as $y_1 \geqq 5.5$ and $y_5 \leqq 5.5$, or $y_1 = y_5 = 5.5$.

Acknowledgments. The author wishes to thank Drs. C. L. Lawson and F. T. Krogh of Jet Propulsion Laboratory for valuable suggestions that improved the algorithm over its original version, [9]. Lawson pointed out the potential numerical disaster associated with a normalization step that made the lower bounds all equal to the value zero. He also noted that one must be prepared to do column pivoting during the retriangularization of the upper Hessenberg matrix. Krogh pointed out that one could use column pivoting, in the heavily weighted problem, to maximize the resulting size of the matrix entry where the plane rotation will be applied.

REFERENCES

[1] C. L. LAWSON AND R. J. HANSON, *Solving Least Squares Problems*, Prentice-Hall, Englewood Cliffs, NJ, 1974.

[2] G. W. STEWART, *Introduction to Matrix Computations*, Academic Press, New York, 1973.

[3] K. HASKELL AND R. J. HANSON, *An algorithm for linear least squares problem with equality and nonnegativity constraints*, Math. Prog., 21 (1981), pp. 98-118.

[4] R. J. HANSON AND K. HASKELL, *Algorithm 587. Two algorithms for the linearly constrained least squares problem*, ACM Trans. Math. Software, 8 (Sept. 1982), No. 3, pp. 323-333.

[5] J. STOER, *On the numerical solution of constrained least squares problems*, SIAM J. Numer. Anal., 8 (1971), pp. 382-411.

[6] P. ZIMMERMANN, *An algorithm for solving linear least squares problems with lower and upper bounds as constraints*, Diplomarbeit, Univ. Würzburg, March 1977.

[7] R. CRANE, B. GARBOW, K. HILLSTROM AND M. MINKOFF, LCLSQ: *An implementation of an algorithm for linearly constrained linear least squares problems*, Argonne Natl. Labs. Rept. ANL-80-116, (1980).

[8] J. STOER AND K. SCHITTKOWSKI, *A factorization method for the solution of constrained least squares problems*, Numer. Math., 31 (1979), pp. 431-463.

[9] R. J. HANSON, *Linear least squares with bounds and linear constraints*, Sandia Labs. Rept. SAND82-1517, Albuquerque, NM, August 1982.

SIAM J. SCI. STAT. COMPUT.
Vol. 7, No. 3, July 1986

009

A GENERALIZATION OF THE FRANK MATRIX*

J. M. VARAH†

Dedicated to James H. Wilkinson on the occasion of his 65th birthday.

Abstract. In this paper, we give a generalization of the well-known Frank matrix and show how to compute its eigensystem accurately. As well, we attempt to explain the ill-condition of its eigenvalues by treating it as a perturbation of a defective matrix.

Key words. eigenvalues, condition, sensitivity

1. Introduction. Over twenty-five years ago, Frank (1958) introduced two matrix examples as tests for an eigenvalue routine. The first, with

$$a_{ij} = n+1-\max(i,j),$$

was the discrete Green's function matrix arising from the standard three-point difference approximation to d^2y/dx^2. The inverse of this matrix is tridiagonal and its eigenvalues are known explicitly, so the computed eigenvalues could be checked.

Frank had little trouble getting good approximations to the eigenvalues, so he tried a variant of this matrix by truncating it to upper Hessenberg form, producing

$$F_n = \begin{pmatrix} n & n-1 & n-2 & \cdots & 1 \\ n-1 & n-1 & n-2 & \cdots & 1 \\ 0 & n-2 & n-2 & \cdots & 1 \\ | & \ddots & \ddots & \ddots & | \\ 0 & \cdots & 0 & 1 & 1 \end{pmatrix}.$$

He had much more difficulty with this matrix (with $n=12$), and in the years since, this matrix has come to be recognized as a good test case for eigenvalue routines: the eigenvalues are real and positive, are extremely ill-conditioned for moderate values of n, yet can be calculated accurately by other means. For example, F_{12} has eigenvalues ranging from 32.2 down to 0.031 (approximately) with condition numbers $1/|s(\lambda)|$ increasing to more than 10^{+7} for the smallest few eigenvalues. Here we define, for a real distinct eigenvalue λ with left and right eigenvectors $\mathbf{w}$ and $\mathbf{z}$, the sensitivity

$$s(\lambda) = \frac{\mathbf{w}^T\mathbf{z}}{\|\mathbf{w}\|_2\|\mathbf{z}\|_2}.$$

As is well known (see e.g. Wilkinson (1965, p. 68)), $\|B\|/|s(\lambda)|$ is a bound for the first order perturbation coefficient in the expansion of the eigenvalue $\lambda(\varepsilon)$ in a power series in ε, when the original matrix A is subjected to a perturbation $A+\varepsilon B$. Thus $|s(\lambda)|<10^{-7}$ means that a change in an element of F_{12} of order 10^{-7} can result in a change in the eigenvalue λ of order 1.

In this paper, we shall give a generalization of this matrix, show how to accurately compute its eigenvalues and eigenvectors, estimate the condition numbers of its eigenvalues, and attempt to explain the seemingly pathological ill-condition which ensues. We feel it is important to fully understand this matrix, as it illuminates the difficulties inherent in computing the eigenstructure, or invariant subspace structure, of general unsymmetric matrices.

* Received by the editors September 3, 1984, and in revised form March 4, 1985.

† Computer Science Department, University of British Columbia, Vancouver, British Columbia, Canada V6T 1W5.

2. The generalization. Start with the symmetric tridiagonal matrix

$$S=\begin{pmatrix} 0 & b_1 & & & \\ b_1 & 0 & b_2 & & \\ & \ddots & \ddots & \ddots & \\ & & b_{n-1} & 0 & \end{pmatrix} \tag{2.1}$$

with $b_i > 0$. Then S has real eigenvalues $\mu_1 \geqq \cdots \geqq \mu_n$ with $\mu_{n+1-i} = -\mu_i$, and these are all well conditioned. Let the corresponding eigenvectors be $\mathbf{x}^{(i)}$.

First form $T = \bar{D}^{-1} S \bar{D}$, with

$$\bar{D} = \operatorname{diag}\left(1, \frac{1}{b_1}, \frac{1}{b_1 b_2}, \cdots, \frac{1}{b_1 b_2 \cdots b_{n-1}}\right).$$

This gives

$$T=\begin{pmatrix} 0 & 1 & & & \\ b_1^2 & 0 & 1 & & \\ & b_2^2 & \ddots & \ddots & \\ & & b_{n-1}^2 & 0 & \end{pmatrix}. \tag{2.2}$$

Now consider $T - \mu I$ for each μ, and relate μ and $\lambda \geqq 0$ by

$$\mu = \frac{\lambda - a}{\sqrt{\lambda}} \tag{2.3}$$

for $a \geqq 0$. Then $\lambda - \mu\sqrt{\lambda} - a = 0$ and hence

$$\lambda(\mu) = \left(\frac{\mu}{2} \pm \sqrt{\left(\frac{\mu}{2}\right)^2 + a}\right)^2. \tag{2.4}$$

Each μ thus gives two λ's, say $\lambda_+(\mu)$, $\lambda_-(\mu)$. However since $(-\mu)$ generates the same pair of λ's, we can make the identification for positive μ, of $+\mu$ with $\lambda_+(\mu)$ and $-\mu$ with $\lambda_-(\mu)$. Now form $G(\lambda) = \sqrt{\lambda} D^{-1}(T - \mu I) D$, where $D = \operatorname{diag}(1, \lambda^{1/2}, \lambda, \lambda^{3/2}, \cdots)$. Then

$$G(\lambda)=\begin{pmatrix} a-\lambda & \lambda & & \\ b_1^2 & a-\lambda & \lambda & \\ & \ddots & \ddots & \ddots \end{pmatrix}$$

and hence $G = L(F - \lambda I)$, with

$$L=\begin{pmatrix} 1 & -1 & & & \\ & 1 & -1 & & \\ & & \ddots & \ddots & \\ & & & -1 & \\ & & & 1 & \end{pmatrix} \quad \text{and} \quad F=\begin{pmatrix} a+b_1^2 & a+b_2^2 & & & a+b_{n-1}^2 & a \\ b_1^2 & a+b_2^2 & & & & \\ & b_2^2 & \ddots & & & \\ & & \ddots & \ddots & & \\ & & & a+b_{n-1}^2 & a & \\ & & & b_{n-1}^2 & a & \end{pmatrix}. \tag{2.5}$$

Since $F - \lambda I = \sqrt{\lambda} L^{-1} D^{-1} (T - \mu I) D$, then (except possibly at $\lambda = 0$) the eigenvalues $\{\lambda_i\}$ of F and $\{\mu_i\}$ of S are related as λ and μ are related above. Thus if $S\mathbf{x} = \mu\mathbf{x}$, then

$$F\mathbf{z} = \lambda\mathbf{z}, \qquad \mathbf{z} = D^{-1}\bar{D}^{-1}\mathbf{x},$$

$$\mathbf{w}^T F = \lambda \mathbf{w}^T, \qquad \mathbf{w}^T = \mathbf{x}^T \bar{D} D L.$$

Thus given a matrix F as above, with any a, $\{b_i\}$, we can find its eigenvalues (λ_i) and eigenvectors $(\mathbf{z}^{(i)}, \mathbf{w}^{(i)})$ using the above technique. The $\{\lambda_i\}$ will be accurate, and so will the vectors as long as the $\{\mu_i\}$ are well-separated. Notice that for $a>0$, we get all $\lambda_i>0$ and in pairs $(\lambda_+, a^2/\lambda_+)$, with an extra root $\lambda = a$ $(\mu = 0)$ for n odd. And for $a=0$, we get pairs $\lambda_- = 0$, $\lambda_+ = \mu^2$, with an extra root $\lambda = 0$ $(\mu = 0)$ for n odd. In this latter case, the formulas above for $\mathbf{z}$ and $\mathbf{w}$ do not hold for $\lambda = 0$. The following special cases are of particular interest:

1. $b_i = \sqrt{n-i}$, $a=1$, giving $F = F_n$ (the Frank matrix). In this case, the matrix

$$S = \begin{pmatrix} 0 & \sqrt{n-1} & & & \\ \sqrt{n-1} & 0 & \sqrt{n-2} & & \\ & \ddots & \ddots & \ddots & 1 \\ & & 1 & 0 & \\ & & & 1 & 0 \end{pmatrix}$$

which is (apart from a constant factor $\sqrt{2}$) the tridiagonal matrix arising using the recurrence relations for the Hermite polynomials $H_n(x)$. Thus the eigenvalues $\{\mu_i\}$ of S are $\sqrt{2}$ times the roots of $H_n(x)$.

2. $b_i = \sqrt{n}$, $a=1$. This gives S with eigenvalues $\mu_j = 2\sqrt{n}\cos(j\pi/(n+1))$, and

$$F \equiv \tilde{F}_n = \begin{pmatrix} n+1 & n+1 & \cdots & n+1 & 1 \\ n & n+1 & \cdots & n+1 & 1 \\ & n & \ddots & \vdots & \vdots \\ & & & n+1 & 1 \\ & & & n & 1 \end{pmatrix}. \tag{2.6}$$

3. Estimating $s(\lambda)$. Given a matrix F as in (2.5), and corresponding tridiagonal matrix S with eigenvalues $\{\mu_i\}$, we can find F's eigenvalues $\{\lambda_i\}$ using (2.4), and for each λ the corresponding eigenvectors $\mathbf{z}$ and $\mathbf{w}$ by forming $\mu = (\lambda - a)/\sqrt{\lambda}$ and the corresponding eigenvector $\mathbf{x}$ of S. Then

$$z_i = \frac{x_i(\prod_1^{i-1} b_j)}{\lambda^{(i-1)/2}}, \qquad w_i = \frac{x_i \lambda^{(i-1)/2}}{\prod_1^{i-1} b_j} - \frac{x_{i-1}\lambda^{(i-2)/2}}{\prod_1^{i-2} b_j}.$$

We are interested here in the sensitivity $s(\lambda) = \mathbf{w}^T\mathbf{z}/\|\mathbf{w}\|_2\|\mathbf{z}\|_2$, for the smaller eigenvalues λ. First, $\mathbf{w}^T\mathbf{z} = \mathbf{x}^T\bar{D}DLD^{-1}\bar{D}^{-1}\mathbf{x} = \mathbf{x}^T\tilde{L}\mathbf{x}$, where

$$\tilde{L} = \begin{pmatrix} 1 & -b_1/\sqrt{\lambda} & & \\ & 1 & \ddots & \\ & & \ddots & -b_2/\sqrt{\lambda} \\ & & & 1 \end{pmatrix}.$$

Thus $\mathbf{w}^T\mathbf{z} = \mathbf{x}^T((\tilde{L}+\tilde{L}^T)/2)\mathbf{x}$, and since $\tilde{L}+\tilde{L}^T = 2I - S/\sqrt{\lambda}$,

$$\mathbf{w}^T\mathbf{z} = \mathbf{x}^T\mathbf{x} - \frac{\mathbf{x}^T S\mathbf{x}}{2\sqrt{\lambda}} = 1 - \frac{\mu}{2\sqrt{\lambda}}.$$

Also,

$$\|\mathbf{z}\|_2 \geqq |z_n| = \frac{|x_n|(\prod_1^{n-1} b_i)}{\lambda^{(n-1)/2}}$$

and $\|\mathbf{w}\|_2 \geqq |w_1| = |x_1|$. This gives

$$|s(\lambda)| \leqq \frac{|1 - \mu/2\sqrt{\lambda}|\lambda^{(n-1)/2}}{|x_1||x_n|(\prod_1^{n-1} b_i)} \leqq \frac{\mu\lambda^{n/2-1}}{2|x_1||x_n|(\prod_1^{n-1} b_i)}. \tag{3.1}$$

For the Frank matrix F_{12}, this bound gives $|s(\lambda)| \leqq 1.2 \times 10^{-7}$ for the smallest λ. For the matrix $\tilde{F}_{12}$ (with $a = 1$ and $b_i = \sqrt{12}$), this gives $|s(\lambda)| \leqq 2 \times 10^{-12}$ for the smallest λ. In fact, the smallest few eigenvalues of F_{12} and $\tilde{F}_{12}$ are as follow in Table 1:

TABLE 1

$\lambda(F_{12})$	$s(\lambda)$	$\lambda(\tilde{F}_{12})$	$s(\lambda)$
$.08122\cdots$	3.8×10^{-8}	$.03465\cdots$	1.6×10^{-12}
$.04950\cdots$	2.6×10^{-8}	$.02524\cdots$	7.8×10^{-13}
$.03102\cdots$	5.5×10^{-8}	$.02117\cdots$	1.3×10^{-12}

4. Understanding the poor condition. To see why these matrices F have such poorly conditioned (small) eigenvalues, one can consider F as the sum of two matrices:

$$F = H + aT, \tag{4.1}$$

$$H = \begin{pmatrix} b_1^2 & b_2^2 & \cdots & b_{n-1}^2 & 0 \\ b_1^2 & b_2^2 & \cdots & b_{n-1}^2 & 0 \\ & b_2^2 & \ddots & \vdots & \vdots \\ & 0 & & b_{n-1}^2 & 0 \end{pmatrix}, \qquad T = \begin{pmatrix} 1 & 1 & \cdots & 1 \\ & 1 & \cdots & 1 \\ & & \ddots & \vdots \\ & & & 1 \end{pmatrix},$$

and treat a as a small parameter. Note: in the examples used, $a = 1$ does not appear to be small, but both F_n and $\tilde{F}_n$ can be scaled by $1/n$ so the elements are $O(1)$ rather than $O(n)$, and the eigenvalues will be scaled by the same amount. This is equivalent to using, for F_n, $b_i = \sqrt{(n-i)/n}$, $a = 1/n$, and for $\tilde{F}_n$, $b_i = 1$, $a = 1/n$.

With a as a small parameter, F can be viewed as a *special* perturbation of H, and H has a well defined eigenstructure: if we assume n is even, then H has $n/2$ eigenvalues at $\lambda = 0$, corresponding to one Jordan block of order $n/2$, and the other $n/2$ eigenvalues at $\lambda = \mu_i^2$, where $\{\mu_i\}$ are the eigenvalues of S (recall these are in $(+,-)$ pairs).

Notice that F is a very special perturbation of H; the eigenvalues remain real, and look asymptotically as follows, using the formula (2.4) for $\lambda(\mu)$:

$$\lambda_+(\mu_i) = \mu_i^2 + 2a + O(a^2) \qquad \text{(perturbed from } \mu_i^2\text{)},$$

$$\lambda_-(\mu_i) = a^2/\mu_i^2 + O(a^3) \qquad \text{(perturbed from zero)}.$$

An arbitrary perturbation of H of order a, would result in generally complex eigenvalues, with $\lambda = O(a^{2/n})$ perturbation from zero, whereas these are $O(a^2)$. However, the sensitivities $|s(\lambda)|$ are still very small for these λ's: $s(\lambda) = O(a^{n-1})$ again using the formulas given earlier.

For the special examples, $a = 1/n$ implies $s(\lambda) = O(1/n^n)$, and this is independent of any constant scaling. These very small values for $s(\lambda)$ can be seen in the smaller eigenvalues of $\tilde{F}_{12}$; for the larger eigenvalues, and for F_{12}, the other constants involved serve to increase the actual values obtained.

We feel this matrix F serves as a very instructive example in connection with the general computational problem of resolving eigenspaces, or more generally invariant subspaces, of unsymmetric matrices. Although all the eigenvalues of F are distinct, as long as $a > 0$, the smaller ones are remarkably ill-conditioned, and the corresponding eigenvectors are not well-determined. Thus one should treat F as a matrix with an invariant subspace of order $n/2$, corresponding to the $n/2$ eigenvalues perturbed from zero in H. However it is not easy to discern this, given only F, since the amount of the perturbation (a) from H is much larger than the machine precision η. Indeed, all

those matrices which differ from F by $O(\eta)$ look much like F, and the canonical matrix H which is strongly influencing F, is a much greater distance away.

As a final comment on the ill-conditioning of F's eigenvalues, it should be mentioned that Wilkinson (1983) shows that *any* λ, $0 \leqq \lambda \leqq 0.1$, is an eigenvalue of $F_n + E_n(\lambda)$ with $\|E_n(\lambda)\|$ very small, even for moderate n.

5. A note on computing eigenvectors. In the course of checking the eigenvectors of these matrices, the author had occasion to return to the standard "inverse iteration" algorithm for computing eigenvectors. Even with a matrix like F, with poorly conditioned eigenvalues and thus poorly determined eigenvectors, we should at least expect each computed eigenvectors $\bar{\mathbf{z}}$ (corresponding to computed eigenvector $\bar{\lambda}$ to give a small residual: $\|F\bar{\mathbf{z}} - \bar{\lambda}\bar{\mathbf{z}}\| = O(\eta)$). This in fact occurred using the EISPACK routines, which however do not use inverse iteration explicitly, but instead perform a back-substitution, using the triangularized form of the matrix.

If inverse iteration is used (normally with the Hessenberg form) for F and computed eigenvalue $\bar{\lambda}$, one solves $(F - \bar{\lambda}I)\mathbf{y} = \mathbf{v}$ for some vector $\mathbf{v}$. Almost always, using a reasonably clever choice of $\mathbf{v}$, $\|\mathbf{y}\| \cong 1/\eta_1$ so that the normalized vector $\bar{\mathbf{z}} = \mathbf{y}/\|\mathbf{y}\|$ gives a small residual. However it can happen that $\|\mathbf{y}\|$ is not "large enough", and one is tempted to do another inverse iteration, solving $(F - \bar{\lambda}I)\mathbf{z} = \mathbf{y}$. For badly conditioned eigenvalues, this is *not* a good idea: the computed $\mathbf{z}$ vector will not give a small residual, because the vector $\mathbf{y}$ is not a good choice of "initial" vector for the inverse iteration.

Because of this phenomenon, it is much safer (although a little more expensive) to iterate with the matrix $(F - \bar{\lambda}I)^T(F - \bar{\lambda}I)$ (with transpose replaced by conjugate transpose in the complex case), as mentioned in Wilkinson (1979). The enforced symmetry avoids the possible embarrassment of large residuals when more iterations are used, essentially because we are now dealing with the *orthogonal* singular vectors not the eigenvectors of F. The algorithm is:

1. decompose $F - \bar{\lambda}I = PLU$.
   If F is upper Hessenberg, $\|L\|$ and $\|L^{-1}\|$ are not large, so the near-singularity of $(F - \bar{\lambda}I)$ is reflected in U; so
2. perform a (double) inverse iteration: solve $U^T U\mathbf{z} = \mathbf{v}$, for $\mathbf{v}$ your favorite initial vector; in the very unlikely case that $\|\mathbf{z}\|/\|\mathbf{v}\| \ll \eta^{-1}$, do another iteration.
3. get the left-hand eigenvector $\mathbf{w}^T$ as well: solve $UU^T\mathbf{y} = \mathbf{v}$ (and again if necessary as in 2) with finally $\mathbf{w}^T(PL) = \mathbf{y}^T$.
4. finally, compute $s(\lambda)$ using the computed $\mathbf{w}^T$ and $\mathbf{z}$.

We cannot emphasize the last step enough; most eigenvalue routines do not compute $s(\lambda)$, and it really is a reliable estimate of the accuracy of the computed eigenvalue.

REFERENCES

W. L. FRANK (1958), *Computing eigenvalues of complex matrices by determinant evaluation and by methods of Danilewski and Wielandt.*, SIAM J. Appl. Math., 6, pp. 378-392.

G. PETERS AND J. H. WILKINSON (1979), *Inverse iteration, ill-conditioned equations, and Newton's method*, SIAM Rev., 21, pp. 339-360.

J. H. WILKINSON (1965), *The Algebraic Eigenvalue Problem*, Oxford Univ. Press, Cambridge.

———, (1983), *Lecture notes for* CS 238b, Computer Science Department, Stanford Univ., Stanford, CA.

SIAM J. SCI. STAT. COMPUT.
Vol. 7, No. 3, July 1986

010

A HYBRID CHEBYSHEV KRYLOV SUBSPACE ALGORITHM FOR SOLVING NONSYMMETRIC SYSTEMS OF LINEAR EQUATIONS*

HOWARD C. ELMAN†, YOUCEF SAAD‡ AND PAUL E. SAYLOR§

Abstract. We present an iterative method for solving large sparse nonsymmetric linear systems of equations that enhances Manteuffel's adaptive Chebyshev method with a conjugate gradient-like method. The new method replaces the modified power method for computing needed eigenvalue estimates with Arnoldi's method, which can be used to simultaneously compute eigenvalues and to improve the approximate solution. Convergence analysis and numerical experiments suggest that the method is more efficient than the original adaptive Chebyshev algorithm.

Key words. iterative methods, Chebyshev methods, conjugate gradient methods, adaptive methods, nonsymmetric matrices, sparse matrices

1. Introduction. The adaptive Chebyshev algorithm of Manteuffel [11], [13] is an iterative method for solving large sparse real nonsymmetric systems of linear equations of the form

$$Ax = b, \tag{1}$$

where the coefficient matrix A has positive-definite symmetric part. Starting from an initial guess, x_0, the method generates a sequence of iterates $\{x_j\}$ whose residuals $\{r_j = b - Ax_j\}$ satisfy

$$r_j = P_j(A)r_0, \tag{2}$$

where

$$P_j(z) = T_j\left(\frac{d-z}{c}\right) \Big/ T_j\left(\frac{d}{c}\right). \tag{3}$$

T_j is the jth Chebyshev polynomial of the first kind

$$T_j(z) = \cosh\,(j \cosh^{-1}(z)),$$

and c and d are *iteration parameters* that depend on the convex hull of the spectrum of A. Two properties of the Chebyshev polynomials make this algorithm effective. First, for an appropriate choice of the iteration parameters, the residual polynomials $P_j(A)$ decrease rapidly in norm, so that the algorithm is rapidly convergent [13]. Second, the three-term recurrence for Chebyshev polynomials induces an inexpensive recurrence for the computation of each iterate x_j.

Because the iteration parameters depend on the convex hull of the spectrum of A, estimates of the extreme eigenvalues of A are needed. Manteuffel's algorithm computes these estimates dynamically [11]. It starts with a (possibly arbitrary) guess for the required parameters and monitors the convergence of the iterates generated. If convergence is deemed unsatisfactory, then information produced during the iteration

* Received by the editors February 28, 1984, and in revised form, March 1, 1985. This work was supported by the U.S. Office of Naval Research under grant N00014-82-K-0184, and by the U.S. Army Research Office under grant DAAG-83-0177.

† Department of Computer Science, University of Maryland, College Park, Maryland 20742. This work was performed while this author was at Yale University.

‡ Computer Science Department, Yale University, New Haven, Connecticut 06520.

§ Computer Science Department, University of Illinois at Urbana-Champaign, Urbana, Illinois 61801.

is used to compute eigenvalue estimates. These, in turn, are used to compute new iteration parameters, and the Chebyshev iteration is restarted with the new parameters. This adaptive procedure is repeated until good iteration parameters are found, after which the Chebyshev method can proceed with no further adaptive steps.

The eigenvalue computation makes use of the residuals generated by the previous Chebyshev iteration. The underlying numerical method is a modified version of the power method. If the values of the iteration parameters used by the Chebyshev iteration are inaccurate, then the residuals generated may diverge. Although divergent residuals may enhance the ability of the adaptive procedure to obtain accurate eigenvalue estimates and iteration parameters, the residual norms may increase by several orders of magnitude before good iteration parameters are found [5], [7]. Thus, the adaptive Chebyshev method may do a considerable amount of work to compute iteration parameters before it makes any improvement in the accuracy of the approximate solution of the linear system.

In this paper, we present an alternative to the eigenvalue computation part of the Manteuffel algorithm that decreases the sensitivity of the Chebyshev method to iteration parameters. We replace the modified power method for computing eigenvalues with Arnoldi's method [1], [18], a generalization of the Lanczos method [16] that estimates the eigenvalues of a nonsymmetric matrix A by reducing it to upper-Hessenberg form. An advantage of this method comes from its relationship to conjugate gradient-like iterative methods for solving nonsymmetric linear systems [5], [17], [20]. At relatively little extra expense, information provided by Arnoldi's method can be used to perform several steps of an iterative method that improves the quality of the solution iterate prior to restarting the Chebyshev iteration with new parameters. The *hybrid method* combines the basic Chebyshev method with this conjugate gradient-like iteration, which is performed whenever new eigenvalue estimates are computed.

In § 2, we briefly describe the original adaptive Chebyshev method. In § 3, we describe Arnoldi's method and its relationship to conjugate gradient-like iterative methods for nonsymmetric linear systems, and we present a convergence result for one of these iterative methods. In § 4, we define the hybrid method and discuss its advantages, and in § 5, we present the results of some numerical experiments comparing the performances of the hybrid method, the adaptive Chebyshev method and the CG-like method Orthomin [4], [5], [23], [25] in solving some discretized non-self-adjoint elliptic partial differential equations.

2. The adaptive Chebyshev method. In this section, we give a brief overview of Manteuffel's adaptive Chebyshev method. For given iteration parameters c and d, the basic Chebyshev iteration is [13].

ALGORITHM 1: The Chebyshev method.

1. *Start*: Choose an initial guess x_0, compute $r_0 = b - Ax_0$ and $p_0 = (1/d)r_0$.
2. *Iterate*: FOR $j = 0$ STEP 1 UNTIL convergence DO:

$$x_{j+1} = x_j + p_j$$

$$r_{j+1} = b - Ax_{j+1}$$

$$\alpha_{j+1} = \begin{cases} 2d/(2d^2 - c^2), & j = 0 \\ [d - (c/2)^2 \alpha_j]^{-1}, & j \geqq 1 \end{cases}$$

$$\beta_{j+1} = d\alpha_{j+1} - 1$$

$$p_{j+1} = \alpha_{j+1} r_{j+1} + \beta_{j+1} p_j.$$

The cost is one matrix-vector product plus $2N$ multiplications per step. The storage required is $4N$ words for x_j, Ax_j, r_j, and p_j. The residuals $\{r_j\}$ satisfy (2) and (3), and P_j is a member of

$$\mathbf{P}_j \equiv \{\text{real polynomials of degree } j \text{ such that } P_j(0) = 1\}. \tag{4}$$

The parameters c and d define the center, d, and foci, $d \pm c$, of a family of confocal ellipses in the complex plane. There is a smallest member of this family, the *smallest ellipse*, that contains the spectrum of A. If the closure of the smallest ellipse does not contain the origin, then Algorithm 1 converges. Moreover, convergence is nearly optimum in the sense that as j increases, P_j rapidly approaches the polynomial in $\mathbf{P}_j$ with minimum uniform norm over the smallest ellipse.

If the spectrum of A lies in the right half plane, then there is an infinite number of smallest ellipses, each of which uniquely corresponds to a set of Chebyshev iteration parameters. For any particular choice of parameters, the rate of convergence of the Chebyshev iteration is [13], [22], [24]

$$-\log\left(\max_{\lambda \in \sigma(A)} S(\lambda)\right), \tag{5}$$

where

$$S(z) = S_{c,d}(z) = \frac{d - z + [(d-z)^2 - c^2]^{1/2}}{d + [d^2 - c^2]^{1/2}}. \tag{6}$$

The iteration count for convergence is (approximately) proportional to the reciprocal of the rate of convergence. Hence, the *best ellipse* is defined to be that smallest ellipse for which the rate of convergence is greatest. The adaptive Chebyshev method starts with (possibly arbitrary) initial values for the iteration parameters and monitors the convergence of the Chebyshev iteration (Algorithm 1). If convergence is deemed unsatisfactory (i.e. the residuals are diverging or converging less rapidly than (5) suggests) after step s, then the adaptive procedure

1. estimates eigenvalues on the convex hull of the spectrum of A [11], and
2. computes the iteration parameters for the best ellipse containing these eigenvalue estimates [13].

The Chebyshev iteration is then restarted with the new parameters. The adaptive procedure is repeated as often as is deemed necessary, until good parameters are found, after which the Chebyshev iteration is performed until convergence.

The second step of the adaptive procedure, the computation of iteration parameters, requires negligible machine resources, and we omit a discussion of it here.

The eigenvalue estimates are computed by a *modified power method*, which is based on the fact that, asymptotically,

$$P_j(z) \approx S(z)^j,$$

so that

$$r_j \approx S(A)^j r_0,$$

where $S(A)$ is the linear operator induced by $S(z)$. That is, the residuals resemble the vectors generated by the power method for $S(A)$. If, for given iteration parameters, some eigenvalue of $S(A)$ has modulus greater than one and r_0 has a component in the corresponding eigenvector, then the residuals will diverge but will eventually become rich in that eigenvector. If there are m such eigenvalues, then eventually the sequence of $m+1$ residuals

$$\{r_s, r_{s+1}, \cdots, r_{s+m}\}$$

will be nearly linearly dependent. Estimates for m eigenvalues of $S(A)$ are then given by the roots of the mth-degree polynomial

$$\alpha_1+\alpha_2 z+\cdots+\alpha_m z^{m-1}+z^m,$$

whose coefficients $\{\alpha_j\}_{j=1}^m$ are the solution to the least squares problem

$$\min \|[r_s, \cdots, r_{s+m-1}]\alpha + r_{s+m}\|_2, \tag{7}$$

where $[r_s, \cdots, r_{s+m-1}]$ denotes the matrix with columns $\{r_j\}_{j=s}^{s+m-1}$ and α denotes the vector whose jth component is α_j[11]. Estimates for eigenvalues of A can be computed from the relationship

$$\mu = S(\lambda)$$

between eigenvalues $\{\mu\}$ of $S(A)$ and $\{\lambda\}$ of A.

Hence, the modified power method consists of m Chebyshev steps to generate the residuals $\{r_j\}_{j=s+1}^{s+m}$, followed by the computation of the least squares solution to (7), and the computation of new eigenvalue estimates and iteration parameters. The Chebyshev steps require m matrix-vector products and $2mN$ multiplications. If (7) is solved using the normal equations, then $[(m^2+3m)/2]N$ multiplications are needed to compute the inner products

$$(r_{s+j}, r_{s+k}), \quad 0 \leqq j \leqq m-1, \quad 0 \leqq k \leqq m. \tag{8}$$

Therefore, the dominant cost is m matrix-vector products plus $[(m^2+7m)/2]N$ additional multiplications. The storage requirement (over that of the Chebyshev iteration) is mN words to save the vectors $\{r_{s+j}\}_{j=1}^m$.

Note that an "unmodified" power method could be used instead of the modified power method by replacing $\{r_{s+j}\}_{j=1}^m$ with $\{A^j r_s\}_{j=1}^m$ in (7) [11]. We will examine a technique that is mathematically equivalent to the unmodified power method in § 3.

3. Arnoldi's method and its relation to iterative linear solvers. In this section, we describe Arnoldi's method for computing eigenvalues of nonsymmetric matrices, show how it can be used as the basis for iterative methods for solving linear systems, and derive a convergence bound for one of these linear solvers.

Given an arbitrary vector v_1 such that $\|v_1\|_2 = 1$, Arnoldi's method [1], [18] is a Galerkin method on the Krylov subspace $K_m = \text{span}\{v_1, Av_1, \cdots, A^{m-1}v_1\}$ for approximating the eigenvalues of A. That is, it finds a set of eigenvalue estimates $\{\lambda_1, \cdots, \lambda_m\}$ such that there exist nonzero $u_i \in K_m$, $i = 1, \cdots, m$, for which

$$(Au_i - \lambda_i u_i, v) = 0, \quad i = 1, \cdots, m \tag{9}$$

for all $v \in K_m$. It accomplishes this by constructing an orthonormal matrix $V_m = [v_1, \cdots, v_m]$ whose columns $\{v_j\}_{j=1}^m$ span K_m, and then computing the eigenvalues of $V_m^T A V_m$.

ALGORITHM 2: Arnoldi's method.

1. *Start*: Choose an initial vector v_1 such that $\|v_1\|_2 = 1$, and a step number m.
2. *Iterate*: FOR $j = 1$ STEP 1 UNTIL m DO

$$h_{ij} = (Av_j, v_i), \quad i = 1, \cdots, j$$

$$\hat{v}_{j+1} = Av_j - \sum_{i=1}^{j} h_{ij} v_i$$

$$h_{j+1,j} = \|\hat{v}_{j+1}\|_2$$

$$v_{j+1} = \hat{v}_{j+1}/h_{j+1,j}.$$

Notice that this method is essentially a Gram-Schmidt process for orthonormalizing the Krylov sequence $\{v_1, Av_1, \cdots, A^{m-1}v_1\}$. In a practical implementation, it is usually more suitable to use a modified Gram-Schmidt process. The orthonormal matrix V_m is such that $V_m^T A V_m = H_m$, where H_m is the $m \times m$ upper-Hessenberg matrix whose (i, j) entry is the scalar h_{ij}. The method generalizes the symmetric Lanczos algorithm to nonsymmetric matrices. Recall that in the symmetric case, H_m is symmetric and tridiagonal [16].

In an implementation, it is not necessary to compute the normalized vectors $\{v_j\}$; it suffices to compute and save the norms $\{\|\hat{v}_j\|_2\}$. It is also not necessary to compute $\hat{v}_{m+1}$. With these conventions, the cost of Arnoldi's method is m matrix-vector products and $(m^2+m)N$ multiplications. The storage requirement is $(m+1)N$ words for $\{v_j\}_{j=1}^m$ and Av [20].

Suppose now that x_0 is a guess to the solution of (1), with residual $r_0 = b - Ax_0$. Let $K_m = \text{span}\{r_0, Ar_0, \cdots, A^{m-1}r_0\}$. One way to solve (1) iteratively is to compute an approximate solution $x_m \in x_0 + K_m$ such that the Galerkin condition

$$(r_m, v) = 0, \qquad v \in K_m \tag{10}$$

holds. But if $v_1 = r_0/\|r_0\|_2$, then the Arnoldi vectors $\{v_j\}_{j=1}^m$ span K_m, so that

$$x_m = x_0 + V_m y_m,$$
$$r_m = r_0 - Az_m = \|r_0\|_2 v_1 - AV_m y_m,$$

for some $y_m \in \mathbf{R}^m$. Since the Arnoldi vectors are orthonormal, (10) is imposed by computing

$$y_m = H_m^{-1}\|r_0\|_2 e_1,$$

where e_j is the jth unit vector in $\mathbf{R}^m$. Hence, the algorithm [17] follows.

ALGORITHM 3: The full orthogonalization method (FOM).

1. *Start*: Choose an initial guess x_0, compute $r_0 = b - Ax_0$ and $v_1 = r_0/\|r_0\|_2$.
2. *Iterate*: Perform m steps of Algorithm 2 starting with v_1.
3. *Form the solution*:

$$\text{Solve } H_m y_m = \|r_0\|_2 e_1,$$
$$\text{compute } x_m = x_0 + V_m y_m,$$

where V_m and H_m are determined by Arnoldi's method.

Algorithm 3 is also referred to as Arnoldi's method for solving linear systems. It is theoretically equivalent to the ORTHORES method developed by Young and Jea [25], which is modelled after a version of the conjugate gradient method described by Engeli et al. [8].

A drawback of Algorithm 3 is that the approximate solution x_m does not satisfy an optimality property. An alternative is the generalized minimal residual method (GMRES) developed by Saad and Schultz [20], which uses the Arnoldi basis to compute the point $x_m \in x_0 + K_m$ whose residual norm $\|b - Ax_m\|_2$ is minimum. Let $v_1 = r_0/\|r_0\|_2$, let $\beta = \|r_0\|_2$, and let $\tilde{H}_m$ denote the $(m+1) \times m$ matrix obtained by appending to H_m a row with single nonzero entry $h_{m+1,m}$ in column m. Then the Arnoldi basis matrices V_m and $V_{m+1} = [v_1, \cdots, v_{m+1}]$ satisfy

$$AV_m = V_{m+1}\tilde{H}_m. \tag{11}$$

The GMRES iterate is given by x_0+z, where z is the solution of the least squares problem

$$\min_{z\in K_m} \|b-A(x_0+z)\|_2 = \min_{z\in K_m} \|r_0-Az\|_2 = \min_{y\in R^m} \|\beta v_1 - AV_m y\|_2. \tag{12}$$

Using (11) and the fact that V_{m+1} is orthonormal, the last expression in (12) is equal to

$$\min_{y\in R^m} \|\beta V_{m+1}e_1 - V_{m+1}\tilde{H}_m y\|_2 = \min_{y\in R^m} \|\beta e_1 - \tilde{H}_m y\|_2. \tag{13}$$

Hence, the GMRES iterate is given by $x_0+V_m y_m$, where y_m is the solution to the upper-Hessenberg least squares problem on the right-hand side of (13).

ALGORITHM 4: The generalized minimal residual method.

1. *Start*: Choose an initial guess x_0, compute $r_0=b-Ax_0$ and $v_1=r_0/\|r_0\|_2$, set $\beta=\|r_0\|_2$.
2. *Iterate*: Perform m steps of Algorithm 2 starting with v_1.
3. *Form the solution*: Find y_m minimizing $\|\beta e_1-\tilde{H}_m y\|_2$ and compute $x_m=x_0+V_m y_m$, where V_m and $\tilde{H}_m$ are determined by Arnoldi's method.

GMRES is a generalization of the MINRES algorithm presented by Paige and Saunders [15]. It is mathematically equivalent to Young and Jea's ORTHODIR [25], for arbitrary nonsingular matrices A. For matrices with positive-definite symmetric part, it is also equivalent to the generalized conjugate residual method [4], [5] and a method of Axelsson [2]. For large step numbers, it requires one third the multiplications and one half the storage of these methods [20].

For both FOM and GMRES, once $\{v_j\}_{j=1}^m$ and H_m are computed, the dominant cost of computing x_m is mN multiplications. Hence, the cost of both methods is m matrix-vector products plus $(m^2+2m)N$ multiplications. In addition to storage for x_j, $(m+1)N$ words are needed for the Arnoldi computation. We remark that for both methods, the residual norm $\|b-Ax_m\|_2$ can be monitored during Step 2 without explicitly computing x_m, so that Step 2 can be stopped as soon as the approximate solution is sufficiently accurate [20].

An error analysis of GMRES can be found in [20]. We derive a new result here that will demonstrate its effectiveness in the hybrid method. Note that the residual $r_m=b-Ax_m$ satisfies

$$\|r_m\|_2 = \min_{P_m\in P_m} \|P_m(A)r_0\|_2,$$

where $\mathbf{P}_m$ is defined by (4). Assume that A is diagonalizable,

$$A = U\Lambda U^{-1}, \tag{14}$$

where Λ is the diagonal matrix of eigenvalues $\{\lambda_j\}_{j=1}^N$ and $U=[u_1,\cdots,u_N]$ is the matrix of eigenvectors of A. Note that U and Λ may be complex. Suppose that the initial residual is dominated by m eigenvectors, i.e.

$$r_0 = \sum_{j=1}^m \alpha_j u_j + e, \tag{15}$$

where $\|e\|_2$ is small in comparison to $\|\sum_{j=1}^m \alpha_j u_j\|_2$, and that, moreover the sum in (15) satisfies

(16) if some complex u_k appears in $\sum_{j=1}^m \alpha_j u_j$, then its conjugate $\bar{u}_k$ appears also.

(In general, this might require including small components in the sum, with a corresponding increase in m.)

THEOREM 1. *If A is diagonalizable and the initial residual satisfies* (15)-(16), *then the residual norm after m steps of* GMRES *satisfies*

$$\|r_m\|_2 \leqq \|U\|_2 \|U^{-1}\|_2 c_m \|e\|_2,$$

where $c_m = \max_{k>m} \prod_{j=1}^{m} |(\lambda_k - \lambda_j)/\lambda_j|$.

Proof. Let $U_m = [u_1, \cdots, u_m]$, $\Lambda_m = \operatorname{diag}(\lambda_1, \cdots, \lambda_m)$, and $a = (\alpha_1, \cdots, \alpha_m)^T$, so that (15) is equivalent to

$$r_0 = U_m a + e.$$

Consider the polynomial

$$\tilde{P}_m(z) = \prod_{j=1}^{m} \frac{z - \lambda_j}{-\lambda_j},$$

which satisfies $\tilde{P}_m(\lambda_j) = 0$, $1 \leqq j \leqq m$, $\tilde{P}_m(0) = 1$. Hence

$$\tilde{P}_m(A) U_m = U_m \tilde{P}_m(\Lambda_m) = 0,$$

so that

$$\tilde{P}_m(A) r_0 = \tilde{P}_m(A) U_m \alpha + \tilde{P}_m(A) e = \tilde{P}_m(A) e.$$

Moreover, by (16), $\tilde{P}_m$ has real coefficients so that

$$\|r_m\|_2 = \min_{P_m \in \mathcal{P}_m} \|P_m(A) r_0\|_2 \leqq \|\tilde{P}_m(A) e\|_2$$

$$\leqq \|U\|_2 \|U^{-1}\|_2 \|\tilde{P}_m(\tilde{\Lambda}_m)\|_2 \|e\|_2,$$

where $\tilde{\Lambda}_m = \operatorname{diag}(\lambda_{m+1}, \cdots, \lambda_N)$. The assertion then follows with

$$c_m = \|\tilde{P}_m(\tilde{\Lambda}_m)\|_2 = \max_{k>m} |\tilde{P}_m(\lambda_k)|. \tag{17}$$

Q.E.D.

Note that the constant c_m *does* depend on $\{\lambda_j\}_{j=1}^m$, and it may not be small if, for example, these eigenvalues are small relative to the others. However, suppose $\{\lambda_j\}_{j=1}^m$ are the m dominant eigenvalues of A (i.e. $|\lambda_j| \geqq |\lambda_k|$ for $j \leqq m$, $k > m$). Then

$$|\lambda_k - \lambda_j| \leqq 2|\lambda_j|$$

for $k > m$, so that

$$c_m \leqq 2^m.$$

Moreover, if $\{\lambda_j\}_{j=1}^m$ are large relative to the remaining eigenvalues, then typically

$$|\lambda_k - \lambda_j| \leqq |\lambda_j|. \tag{18}$$

Similarly, let $\lambda_j = \mu_j + i\nu_j$, $\lambda_k = \mu_k + i\nu_k$. Then

$$\left|\frac{\lambda_k - \lambda_j}{\lambda_j}\right|^2 \leqq \left(1 - \frac{\mu_k}{\mu_j}\right)^2 + \frac{(\nu_j - \nu_k)^2}{\mu_j^2 + \nu_j^2},$$

which is less than one if λ_j and λ_k are real, and will typically be of order one if their imaginary parts ν_j, ν_k are small. In all these cases, c_m will be of order one, and the m steps of GMRES reduce the residual norm to the order of $\|e\|_2$ provided that the condition number of U is not too large. Finally, the possibility that c_m may be large is really only a problem if the bound of the theorem is sharp. But in that case, the polynomial used in the proof is nearly optimal, and therefore $\|e\|_2$ is very small.

4. The hybrid method. The hybrid method combines the approaches of the previous two sections. It uses the basic Chebyshev iteration of Algorithm 1, but replaces the modified power method for computing eigenvalues with Arnoldi's method, from which information is also used to improve the solution iterate. Either FOM or GMRES could be used for the solution update; we favor GMRES because of its minimization property. In the following implementation, the convergence of the Chebyshev iteration is monitored by examining the norms of the generated residuals, and the adaptive procedure is invoked if the residual norm exceeds a specified tolerance r relative to the norm of $r_{\min} = r_{\min}(c, d)$, the smallest residual encountered with the current iteration parameters. In addition, the adaptive procedure is invoked periodically, after at most s Chebyshev steps, and it is used to generate initial eigenvalue estimates from which initial iteration parameters are obtained.[1]

ALGORITHM 5: The hybrid method.
Choose x_0. Compute $r_0 = b - Ax_0$.
UNTIL Convergence DO
Adaptive Steps: Set v_1 = the current normalized residual, perform m Arnoldi/GMRES steps (Algorithm 4), and use the new eigenvalue estimates to update (or initialize) the iteration parameters.
Chebyshev Steps: Set $j_{\max} = j + s$.
WHILE ($\|r_j\|_2 / \|r_{\min}\|_2 \leqq r$ and $j+1 \leqq j_{\max}$)
Compute x_{j+1} by the Chebyshev iteration.

The Chebyshev step requires one matrix-vector product and $2N$ multiplications per iteration, and the adaptive step requires m matrix-vector products and $(m^2+2m)N$ multiplications. As with the modified power method, the eigenvalue estimates provided by Arnoldi's method lie in the field of values of A but not necessarily in the convex hull of the spectrum of A, so that the hybrid method is only rigorously applicable to linear systems with positive-definite symmetric part. The storage requirement for the adaptive step is mN words, the same as for the modified power method, since the first Arnoldi vector can share storage with the residual of the Chebyshev iteration.

There are two main differences between the original adaptive Chebyshev method and the hybrid method:

1. Different eigenvalue computations: the adaptive Chebyshev method uses the modified power method based on the operator $S(A)$, whereas the hybrid method uses Arnoldi's method, which is based on a Krylov subspace in A.

2. Purification: the hybrid method uses the GMRES steps to improve the approximate solution.

A third difference is that in the hybrid method, the initial eigenvalue estimates provided by Arnoldi's method can be used to compute initial iteration parameters; the original Chebyshev method requires an initial guess.

We do not know whether the use of Arnoldi's method alone offers any advantage, i.e. whether Arnoldi's method provides more accurate eigenvalue estimates than the modified power method. Arnoldi's method is mathematically equivalent to the "unmodified" power method discussed by Manteuffel [11], who observed no significant difference between the unmodified and modified methods. Numerical experiments comparing the two techniques are described in § 5.

[1] Manteuffel's method [13] for computing iteration parameters from eigenvalue estimates is still used.

The effect of the GMRES steps can be explained by a heuristic analysis based on Theorem 1. Assume that A is diagonalizable as in (14). If the initial residual for the hybrid method has the form

$$r_0 = \sum_{j=1}^{N} \gamma_j u_j,$$

then after s Chebyshev iterations, the residual is approximately equal to [13]

$$\tilde{r} = \sum_{j=1}^{N} \tau_j^s \gamma_j u_j, \tag{19}$$

where

$$\tau_j = S(\lambda_j) = \frac{d - \lambda_j + [(d-\lambda_j)^2 - c^2]^{1/2}}{d + [d^2 - c^2]^{1/2}},$$

and c, d are the iteration parameters used in the Chebyshev step. Suppose that these parameters are inaccurate, so that the components in the directions of some eigenvectors are not being damped out. This means that some of the $\{\tau_j\}$ satisfy $|\tau_j| > 1$, so that $|\tau_j^s| \gg 1$ and the terms with these coefficients dominate (19). Note that $|\tau_j| = |\bar{\tau}_j|$, so that if some complex eigenvector is not being damped out, then neither is its conjugate. For some m, therefore, $\tilde{r}$ satisfies (15) (with $\alpha_j = \tau_j^s \gamma_j$) and (16). If the corresponding eigenvalues $\{\lambda_j\}$ of A are the dominant ones, then Theorem 1 suggests that the m GMRES steps *purify* the residual of the eigenvectors whose coefficients had been growing during the Chebyshev iteration. Moreover, since $\tilde{r}$ is the starting vector for the Arnoldi computation and is presumably rich in these eigenvectors, the new eigenvalue estimates will be good approximations to the corresponding eigenvalues. Hence, the new iteration parameters will produce Chebyshev polynomials that continue to damp out these components.

Although the correct value of m to use in the adaptive step is not known in general, this analysis still shows that m GMRES steps will tend to damp out the m dominant components of (19). The analysis applies as well even if the iteration parameters are accurate but not optimal, i.e. the Chebyshev iteration is damping out all components but better parameters exist. In this case, some components will not be damped out as rapidly as others during the Chebyshev step, and these will eventually be dominant in (19).

Since the purification step seems to provide the important advantage of the hybrid method, it is natural to ask whether a similar idea can be implemented with the modified power method, which uses $\{r_{s+j} \approx S(A)^{s+j} r_0\}_{j=0}^{m}$ to compute eigenvalue estimates. One such procedure consists of computing

$$\tilde{x} \in x_{s+m} + \text{span}\{r_s, \cdots, r_{s+m-1}\} \tag{20}$$

for which $\|\tilde{r}\|_2 = \|b - A\tilde{x}\|_2$ is minimum. This requires the solution of the least squares problem

$$\min \|r_{s+m} - \sum_{j=0}^{m-1} \alpha_j A r_{s+j}\|_2. \tag{21}$$

To solve (21) using the normal equations, it is necessary to compute the inner products

$$(Ar_{s+j}, Ar_{s+k}), \qquad 0 \leqq j \leqq k \leqq m-1, \tag{22}$$

$$(r_{s+m}, Ar_{s+j}), \qquad 0 \leqq j \leqq m-1. \tag{23}$$

Note that the recurrence for the Chebyshev iteration induces a three-term residual recurrence

$$Ar_j = -\frac{\beta_j}{\alpha_j} r_{j-1} + \frac{1+\beta_j}{\alpha_j} r_j - \frac{1}{\alpha_j} r_{j+1}. \tag{24}$$

Therefore, except when $j=0$, all the quantities of (22) can be computed in terms of

$$(r_{s+t}, r_{s+u}), \qquad t = j-1, j, j+1, \quad u = k-1, k, k+1,$$

which are available from the modified power method (see (8) above). Similarly, except when $j=0$ and $j=m-1$, the terms of (23) are available from the modified power method. Moreover, the same trick can be used for $j=0$ in (22) if r_{s-1} is saved and $\{(r_{s-1}, r_{s+k})\}_{k=-1}^{m-1}$ are computed; and for $j=0$ and $j=m-1$ in (23) if (r_{s-1}, r_s) and (r_{s+m}, r_{s+m}) are computed. Hence (21) can be solved with a total of $m+3$ inner products. The computation of $\tilde{x}$ requires an additional mN multiplications, so that purification can be added to the modified power method with $(2m+3)N$ multiplications. Combining this with the $[(m^2+7m)/2]N$ multiplications and m matrix-vector products required for the modified power method, the cost of this adaptive procedure is m matrix-vector products plus $[(m^2+11m+6)/2]N$ multiplications. This contrasts with m matrix-vector products and $(m^2+2m)N$ multiplications for the hybrid method. Thus, the number of matrix-vector products is the same as for the hybrid method, but the number of additional operations is different. The coefficient of N for the additional operation counts of both methods, for several values of m, is shown in Table 4.1. The storage requirement is $(m+1)N$ words, for $\{r_{s+j}\}_{j=1}^{m}$ and r_{s-1}, which is N greater than for the hybrid method.[2]

TABLE 4.1
Coefficient of N in multiplication count of purification adaptive steps.

	m	2	4	6	8	10
Hybrid		8	24	48	80	120
Modified power with purification		16	33	54	79	108

Finally, we note that similar methods for annihilating eigencomponents have been developed in slightly different contexts by Saad and Sameh [19] and by Jesperson and Buning [10].

5. Numerical experiments. In this section, we describe the results of numerical experiments in which the methods discussed above are used to solve some nonsymmetric linear systems arising from the discretization of non-self-adjoint elliptic boundary value problems. We examine four methods based on four choices for the adaptive procedure:

(A) CHEB: the modified power method with no purification;
(B) HYBRID: Arnoldi's method with purification by GMRES;
(C) CHEB-MIN: the modified power method with purification added by solving (21);
(D) CHEB-ARNOLDI: Arnoldi's method *without* purification.

[2] Note that the space in (20) does not contain the most recent information available, since r_{s+m} is excluded. We exclude it to avoid the computation of Ar_m in (21). Also, a less expensive purification, with no reference to r_{s-1}, could be performed if r_s were excluded. The given method is a compromise between these two alternatives.

The experiments were run on a VAX11-780 in double precision (55 bit mantissa). The Chebyshev iterations were based on a slightly modified version of Manteuffel's Chebyshev code [12]. The eigenvalues of the upper-Hessenberg matrix H_m generated by Arnoldi's method were computed using EISPACK [21].

Table 5.1 summarizes the work and storage requirements for the adaptive procedures of each of the four methods. The matrix-vector products are denoted by Av.

TABLE 5.1
Work and storage requirements for the adaptive procedures.

	CHEB	HYBRID	CHEB-MIN	CHEB-ARNOLDI
Work	$mAv+(m^2+7m)N/2$	$mAv+(m^2+2m)N$	$mAv+(m^2+11m+6)N/2$	$mAv+(m^2+m)N$
Storage	mN	mN	$(m+1)N$	mN

As in Algorithm 5, the adaptive procedure of each method is invoked if

$$\|r_i\|_2 > \tau \|r_{\min}\|_2, \tag{25}$$

where $r_{\min}$ is the smallest residual encountered for the current parameters, and $\tau = 2$.[3] For HYBRID and CHEB-MIN, it is also invoked after at most $s = 20$ Chebyshev steps so that the purification step is performed periodically. Since no purification occurs in CHEB and CHEB-ARNOLDI, these techniques allow the Chebyshev iteration to proceed if the convergence seems to agree with the predicted rate of convergence.[4] HYBRID and CHEB-ARNOLDI compute initial values for the iteration parameters c and d from eigenvalue estimates provided by Arnoldi's method applied to the initial residual. CHEB and CHEB-MIN use $c = 0$ and $d = 1$ as the initial iteration parameters. Following [11], we use $m = 4$ as the size of the Arnoldi and modified power bases in an effort to identify the dominant and subdominant complex eigenvalue pair. Table 5.2 contains the work and storage costs of the adaptive procedures for this value of m.

TABLE 5.2
Costs of the adaptive procedures, $m = 4$.

	CHEB	HYBRID	CHEB-MIN	CHEB-ARNOLDI
Work	$4Av+22N$	$4Av+24N$	$4Av+33N$	$4Av+20N$
Storage	$4N$	$4N$	$5N$	$4N$

For the test problem, we use the elliptic partial differential equation

$$-(e^{-xy}u_x)_x-(e^{xy}u_y)_y+\gamma[(x+y)u_y+((x+y)u)_y]+[1/(1+x+y)]u=f, \tag{26}$$

where γ is a real scalar parameter and the right-hand side f is chosen so that the solution is

$$u(x, y) = x\, e^{xy} \sin(\pi x) \sin(\pi y).$$

[3] CHEB and CHEB-MIN make this test only if i is a multiple of $m = 4$. This convention is taken from the Chebyshev code and may slightly enhance the modified power method by allowing greater residual growth than indicated by (25).

[4] If j is the index of the first Chebyshev iterate corresponding to the current iteration parameters, then asymptotically $\|r_{j+t}\|_2/\|r_j\|_2$ is bounded by $\max S(|\lambda|)^t$ for $\lambda \in \sigma(A)$ [13]. The heuristic, built into the original code [12], is to compute new parameters only if $\|r_{j+t}\|_2/\|r_j\|_2 > 2S(d)^t$.

We pose (26) on the unit square $\{0 \leqq x, y \leqq 1\}$ with homogeneous Dirichlet boundary conditions and discretize using the five-point second order centered finite difference scheme on a uniform 47×47 grid, producing a linear system

$$Ax = b \tag{27}$$

or order $N = 2209$. We use the values $\gamma = 5$ and $\gamma = 50$. In addition, we *precondition* (27) with incomplete factorizations. We use both the incomplete LU (ILU) and modified incomplete LU (MILU) factorizations with no extra fill-in (see [3], [5], [9], [14] for the details concerning these techniques). The actual linear systems on which the various iterative methods are tested have the form

$$\tilde{A}\tilde{x} = [AQ^{-1}][Qx] = b = \tilde{b},$$

where Q is the *preconditioning matrix*. We thus have four test problems:

- Problem 1: $\gamma = 5$, ILU preconditioning,
- Problem 2: $\gamma = 5$, MILU preconditioning,
- Problem 3: $\gamma = 50$, ILU preconditioning,
- Problem 4: $\gamma = 50$, MILU preconditioning.

The eigenvalue estimates (computed by all four methods) are real in Problems 1 and 2, and have imaginary parts of order one in Problems 3 and 4. For all tests, the initial guess is $x_0 = 0$ and the stopping criterion is $\|r_i\|_2 / \|r_0\|_2 < 10^{-}$.

Table 5.3 shows the number of iterations required to satisfy the stopping criterion, where an iteration for the four adaptive Chebyshev methods is defined to be either a Chebyshev step or an Arnoldi step. Thus, one iteration does not correspond to a fixed amount of work, although each iteration contains one matrix-vector product.

TABLE 5.3
Iterations to convergence.

	CHEB	HYBRID	CHEB-MIN	CHEB-ARNOLDI	ORTHO-MIN(1)
Problem 1	90	60	64	77	78
Problem 2	35	27	34	44	32
Problem 3	36	42	31	59	32
Problem 4	33	27	31	34	21

Figures 5.1–5.4 show the performance of the methods on each of the four problems. The coordinates are residual norm $\|r_i\|_2$ (on a logarithmic scale) vs. multiplications. As a benchmark, for each problem we also include the performance of the conjugate gradient-like method Orthomin (1) [4], [5], [23], [25]. Note that numerical experiments indicating that Chebyshev methods (as well as Orthomin) are more effective than the conjugate gradient method applied to the normal equations and the biconjugate gradient method are presented in [5], [6], [7].

In examining this data, we consider three main issues:

1. the effect of the purification steps in HYBRID and CHEB-MIN;
2. the effect of the different eigenvalue estimators: Arnoldi's method in HYBRID and CHEB-ARNOLDI vs. the modified power method in CHEB and CHEB-MIN;

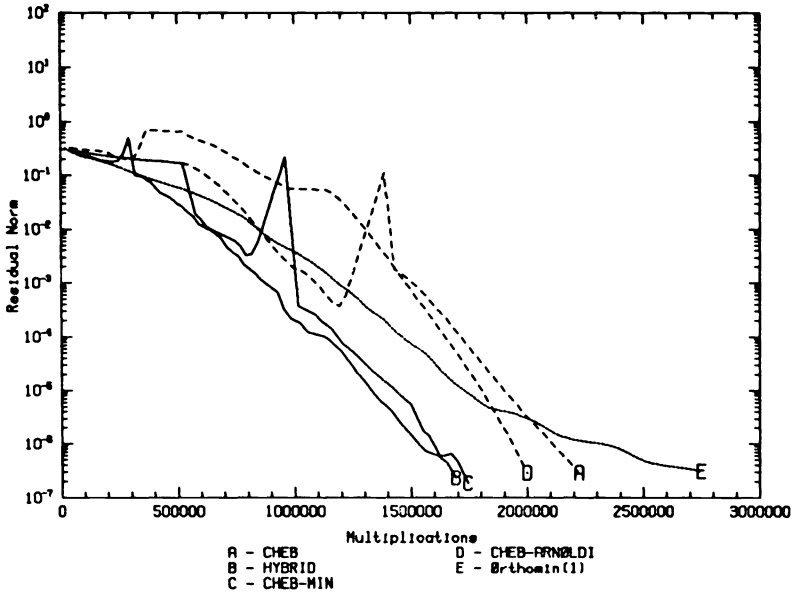


FIG. 5.1. *Problem* 1: $\gamma = 5$, ILU *preconditioning.*

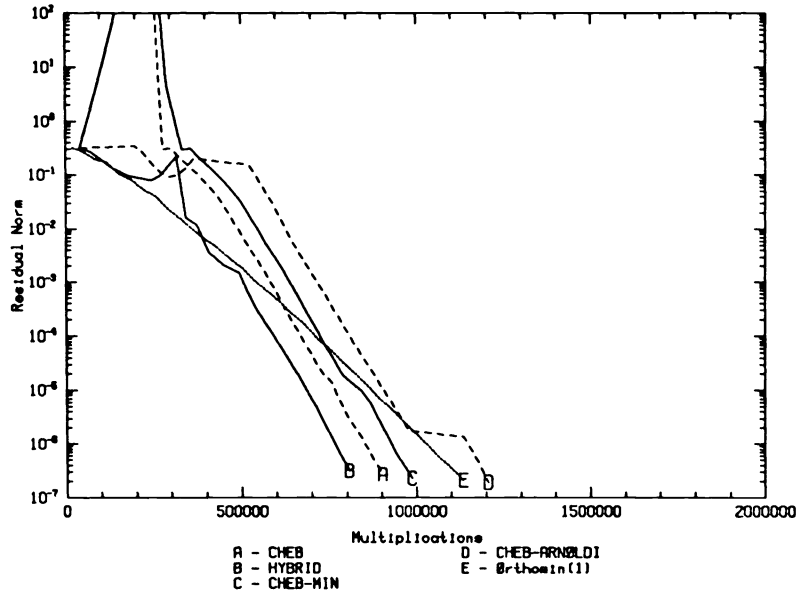


FIG. 5.2. *Problem* 2: $\gamma = 5$, MILU *preconditioning.*

3. the different choice of initial parameters: an initial Arnoldi computation in HYBRID and CHEB-ARNOLDI vs. initial guesses of $d = 1$, $c = 0$ in CHEB and CHEB-MIN.

The first issue is clearcut: for all four problems, the method with purification is superior to its analogue without purification. This is explained by the analysis of § 4: if the residuals from the Chebyshev steps are diverging, then the purification essentially annihilates the eigenvector components that are growing, at relatively little extra cost.

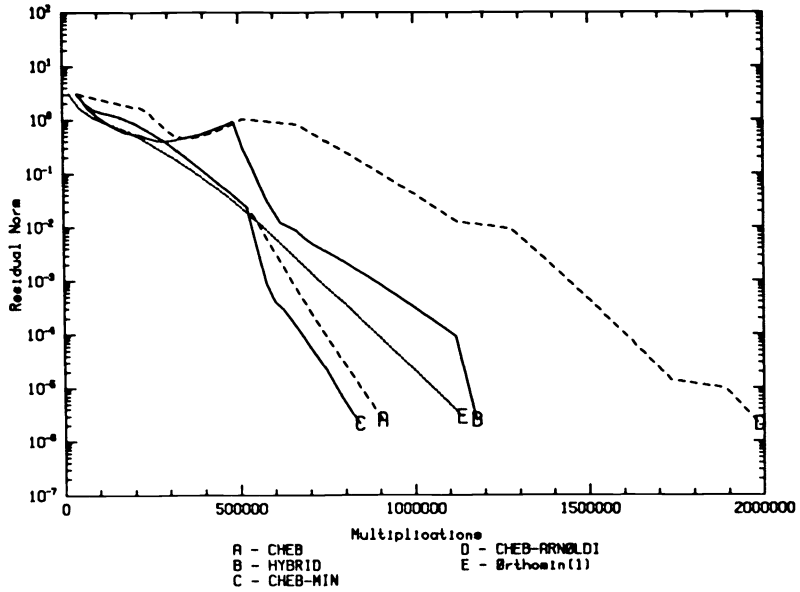


FIG. 5.3. *Problem* 3: $\gamma = 50$, ILU *preconditioning.*

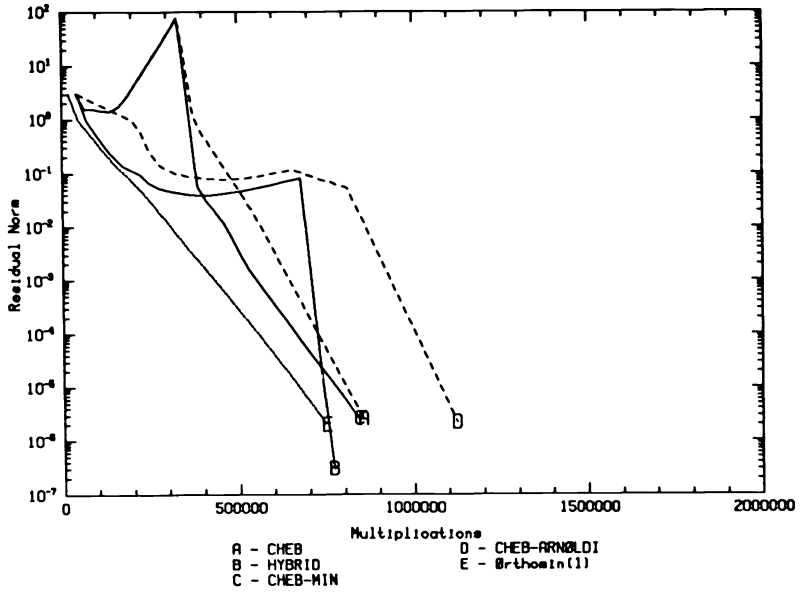


FIG. 5.4. *Problem* 4: $\gamma = 50$, MILU *preconditioning.*

A direct comparison between the two techniques for estimating eigenvalues is somewhat difficult because of the different roles of the growth tolerance parameter τ. In the modified power method, four Chebyshev iterations are performed *after* the condition (25) is violated, so that the residuals will become very rich in the needed eigenvectors. In contrast, Arnoldi's method is performed as soon as (25) is violated, so that the residuals will probably not be dominated as much by these eigenvectors. Without purification, Arnoldi's method (in CHEB-ARNOLDI) does not seem as

effective as the modified power method (in CHEB). However, the combined Arnoldi/GMRES step of HYBRID appears to be more effective than the purified modified power step of CHEB-MIN. It is both less expensive (for $m=4$), and it strongly limits the growth of the residual.

For the third issue, note that inaccurate initial iteration parameters cause the residuals generated by CHEB and CHEB-MIN to diverge by several orders of magnitude in Problems 1, 2 and 4 (the missing eigenvalues take some time to assert themselves in Problem 1). This difficulty is avoided by HYBRID in Problems 1 and 2, where fairly accurate initial eigenvalue estimates combine with the strict growth tolerance $\tau=2$ to prevent divergence. HYBRID does not handle Problem 4 as well. This is because the initial Arnoldi estimates determine a domain of convergence for the Chebyshev iteration that just misses one eigenvalue, and the next Chebyshev iteration diverges too slowly for the adaptive procedure to be invoked until the maximum number of 20 steps is performed. In Problem 3, the eigenvalues are clustered near 1 so that the initial parameters for CHEB and CHEB-MIN are accurate, whereas Arnoldi's method has some difficulty identifying them. The use of Arnoldi's method for initial eigenvalue estimates tends to make the overall performance somewhat smoother, although it may not be necessary if good initial parameters are available.

Finally, note that the performances of Orthomin (1) and the Chebyshev methods are very close. The slopes of the Chebyshev curves are steeper, reflecting their lower cost per step [4], [5], [13], but the overhead of the adaptive steps increases their total cost.

Acknowledgments. The authors wish to thank Tom Manteuffel for providing us with a copy of his Chebyshev code, without which this project would have been nearly impossible, and Martin Schultz and Stan Eisenstat for several helpful suggestions during the preparation of this paper.

REFERENCES

[1] W. E. ARNOLDI, *The principle of minimized iterations in the solution of the matrix eigenvalue problem*, Quart. Appl. Math., 9 (1951), pp. 17-29.

[2] O. AXELSSON, *Conjugate gradient type methods for unsymmetric and inconsistent systems of linear equations*, Linear Algebra Appl., 29 (1980), pp. 1-16.

[3] T. DUPONT, R. P. KENDALL AND H. H. RACHFORD JR., *An approximate factorization procedure for solving self-adjoint elliptic difference equations*, SIAM J. Numer. Anal., 5 (1968), pp. 559-573.

[4] S. C. EISENSTAT, H. C. ELMAN AND M. H. SCHULTZ, *Variational iterative methods for nonsymmetric systems of linear equations*, SIAM J. Numer. Anal., 20 (1983), pp. 345-357.

[5] H. C. ELMAN, *Iterative methods for large, sparse, nonsymmetric systems of linear equations*, Ph.D. Thesis, Dept. Computer Science, Yale Univ., New Haven, CT, 1982; also available as Technical Report 229.

[6] ———, *Iterative methods for non-self-adjoint elliptic problems*, Technical Report 266, Dept. Computer Science, Yale Univ., New Haven, CT, April 1983; Proc. Elliptic Problem Solvers Conference, Monterey, CA, Jan. 1983, G. Birkhoff and A. L. Schoenstadt, eds., Academic Press, New York, to appear.

[7] ———, *Preconditioned conjugate gradient methods for nonsymmetric systems of linear equations*, in Advances in Computer Methods for Partial Differential Equations-IV, R. Vichnevetsky and R. S. Stepleman, eds., IMACS, New Brunswick, NJ, 1981, pp. 409-417.

[8] M. ENGELI, T. GINSBERG, H. RUTISHAUSER AND E. STIEFEL, *Refined Iterative Methods for Computation of the Solution and the Eigenvalues of Self-Adjoint Boundary Value Problems*, Birkhauser Verlag, Basel, Stuttgart, 1959.

[9] I. GUSTAFSSON, *A class of first order factorizations*, BIT, 18 (1978), pp. 142-156.

[10] D. C. JESPERSEN AND P. G. BUNING, *Accelerating an iterative process by explicit annihilation*, Technical Report, NASA-Ames Research Center, 1983.

[11] T. A. MANTEUFFEL, *Adaptive procedure for estimation of parameters for the nonsymmetric Tchebychev iteration*, Numer. Math., 31 (1978), pp. 187-208.

[12] ———, *An iterative method for solving nonsymmetric linear systems with dynamic estimation of parameters*, Ph.D. Thesis, Dept. Computer Science, Univ. Illinois at Urbana-Champaign, 1975.
[13] ———, *The Tchebychev iteration for nonsymmetric linear systems*, Numer. Math., 28 (1977), pp. 307–327.
[14] J. A. MEIJERINK AND H. A. VAN DER VORST, *An iterative solution method for linear systems of which the coefficient matrix is a symmetric M-matrix*, Math. Comput., 31 (1977), pp. 148–162.
[15] C. C. PAIGE AND M. A. SAUNDERS, *Solution of sparse indefinite systems of linear equations*, SIAM J. Numer. Anal., 12 (1975), pp. 617–629.
[16] B. N. PARLETT, *The Symmetric Eigenvalue Problem*, Prentice-Hall, Englewood Cliffs, NJ, 1980.
[17] Y. SAAD, *Krylov subspace methods for solving large unsymmetric linear systems*, Math. Comput., 37 (1981), pp. 105–126.
[18] ———, *Variations on Arnoldi's method for computing eigenelements of large unsymmetric matrices*, Linear Algebra Appl., 34 (1980), pp. 269–295.
[19] Y. SAAD AND A. SAMEH, *Iterative methods for the solution of elliptic differential equations on multiprocessors*, in Proc. CONPAR 81 Conference, W. Handler, ed., Springer-Verlag, New York, 1981, pp. 395–411.
[20] Y. SAAD AND M. H. SCHULTZ, GMRES, *A generalized minimal residual algorithm for solving nonsymmetric linear systems*, Tech. Report 254, Dept. Computer Science, Yale Univ., New Haven, CT, 1983.
[21] B. T. SMITH, J. M. BOYLE, J. J. DONGARRA, B. S. GARBOW, Y. IKEBE, V. C. KLEMA AND C. B. MOLER, *Matrix Eigensystem Routines*-EISPACK *Guide*, Lecture Notes in Computer Science 6, Springer-Verlag, Heidelberg, 1976.
[22] R. S. VARGA, *Matrix Iterative Analysis*, Prentice-Hall, Englewood Cliffs, NJ, 1962.
[23] P. K. W. VINSOME, *Orthomin, an iterative method for solving sparse sets of simultaneous linear equations*, in Proc. Fourth Symposium on Reservoir Simulation, Society of Petroleum Engineers of AIME, 1976, pp. 149–159.
[24] D. M. YOUNG, *Iterative Solution of Large Linear Systems*, Academic Press, New York, 1971.
[25] D. M. YOUNG AND K. C. JEA, *Generalized conjugate gradient acceleration of nonsymmetrizable iterative methods*, Linear Algebra Appl., 34 (1980), pp. 159–194.

SIAM J. SCI. STAT. COMPUT.
Vol. 7, No. 3, July 1986

011

GMRES: A GENERALIZED MINIMAL RESIDUAL ALGORITHM FOR SOLVING NONSYMMETRIC LINEAR SYSTEMS*

YOUCEF SAAD† AND MARTIN H. SCHULTZ†

Abstract. We present an iterative method for solving linear systems, which has the property of minimizing at every step the norm of the residual vector over a Krylov subspace. The algorithm is derived from the Arnoldi process for constructing an l_2-orthogonal basis of Krylov subspaces. It can be considered as a generalization of Paige and Saunders' MINRES algorithm and is theoretically equivalent to the Generalized Conjugate Residual (GCR) method and to ORTHODIR. The new algorithm presents several advantages over GCR and ORTHODIR.

Key words. nonsymmetric systems, Krylor subspaces, conjugate gradient, descent methods, minimal residual methods

AMS(MOS) subject classification. 65F

1. Introduction. One of the most effective iterative methods for solving large sparse symmetric positive definite linear systems of equations is a combination of the conjugate gradient method with some preconditioning technique [3], [8]. Moreover, several different generalizations of the conjugate gradient method have been presented in the recent years to deal with nonsymmetric problems [2], [9], [5], [4], [13], [14] and symmetric indefinite problems [10], [3], [11], [14].

For solving indefinite symmetric systems, Paige and Saunders [10] proposed an approach which exploits the relationship between the conjugate gradient method and the Lanczos method. In particular, it is known that the Lanczos method for solving the eigenvalue problem for an $N \times N$ matrix A is a Galerkin method onto the Krylov subspace $K_k \equiv \text{span}\{v_1, Av_1, \cdots, A^{k-1}v_1\}$, while the conjugate gradient method is a Galerkin method for solving the linear system $Ax = f$, onto the Krylov subspace K_k with $v_1 = r_0/\|r_0\|$. Thus, the Lanczos method computes the matrix representation T_k of the linear operator $P_k A_{|K_k}$, the restriction of $P_k A$ to K_k, where P_k is the l_2-orthogonal projector onto K_k. The Galerkin method for $Ax = f$ in K_k leads to solving a linear system with the matrix T_k which is tridiagonal if A is symmetric. In general, T_k is indefinite when A is and some stable direct method must be used to solve the corresponding tridiagonal Galerkin system. The basis of Paige and Saunders' SYMMLQ algorithm is to use the stable LQ factorization of T_k. Paige and Saunders also showed that it is possible to formulate an algorithm called MINRES using the Lanczos basis to compute an approximate solution x_k which minimizes the residual norm over the Krylov subspace K_k.

In the present paper we introduce and analyse a generalization of the MINRES algorithm for solving nonsymmetric linear systems. This generalization is based on the Arnoldi process [1], [12] which is an analogue of the Lanczos algorithm for nonsymmetric matrices.

Instead of a tridiagonal matrix representing $P_k A_{|K_k}$, as is produced by the Lanczos method for symmetric matrices, Arnoldi's method produces an upper Hessenberg matrix. Using the l_2-orthonormal basis generated by the Arnoldi process, we will show that the approximate solution which minimizes the residual norm over K_k, is easily computed by a technique similar to that of Paige and Saunders. We call the resulting

* Received by the editors November 29, 1983, and in revised form May 8, 1985. This work was supported by the Office of Naval Research under grant N000014-82-K-0184 and by the National Science Foundation under grant MCS-8106181.

† Department of Computer Science, Yale University, New Haven, Connecticut 06520.

algorithm the Generalized Minimal Residual (GMRES) method. We will establish that GMRES is mathematically equivalent to the generalized conjugate residual method (GCR) [5], [16] and to ORTHODIR [9]. It is known that when A is positive real, i.e. when its symmetric part is positive definite, then the generalized conjugate residual method and the ORTHODIR method will produce a sequence of approximations x_k which converge to the exact solution. However, when A is not positive real GCR may break down. ORTHODIR on the other hand does not break down, but is known to be numerically less stable than GCR [5], although this seems to be a scaling difficulty.

Thus, systems in which the coefficient matrix is not positive real provide the main motivation for developing GMRES. For the purpose of illustration, consider the following 2×2 linear system $Ax=f$, where

$$A=\begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}, \quad f=\begin{bmatrix} 1 \\ 1 \end{bmatrix}, \quad x_0=0.$$

The GCR algorithm can be briefly described as follows:

1. *Start*: Set $p_0=r_0=f-Ax_0$
2. *Iterate*: For $i=0, 1, \cdots$ until convergence do:
   Compute $\alpha_i=(r_i, Ap_i)/(Ap_i, Ap_i)$,
   $x_{i+1}=x_i+\alpha_i p_i$,
   $r_{i+1}=r_i-\alpha_i Ap_i$,
   $p_{i+1}=r_{i+1}+\sum_{j=0}^{i}\beta_j^{(i)}p_j$
   where $\{\beta_j^{(i)}\}$ are chosen so that $(Ap_{i+1}, Ap_j)=0$, for $0\leqq j\leqq i$.

If one attempts to execute this algorithm for the above example one would obtain the following results:

1. At step $i=0$ we get $\alpha_0=0$ and therefore $x_1=x_0$, $r_1=r_0$. Moreover, the vector p_1 is zero.
2. At step $i=1$, a division by zero takes place when computing α_1 and the algorithm breaks down.

We will prove that GMRES *cannot* break down even for problems with indefinite symmetric parts unless it has already converged. Moreover, we will show that the GMRES method requires only half the storage required by the GCR method and $\frac{1}{3}$ fewer arithmetic operations than GCR.

In § 2 we will briefly recall Arnoldi's method for generating l_2-orthogonal basis vectors as it is described in [13]. In § 3, we will present the GMRES algorithm and its analysis. Finally, in § 4 we present some numerical experiments.

2. Arnoldi's method. Arnoldi's method [1] which uses the Gram-Schmidt method for computing an l_2-orthonormal basis $\{v_1, v_2, \cdots, v_k\}$ of the Krylov subspace $K_k \equiv$ span $\{v_1, Av_1, \cdots, A^{k-1}v_1\}$ can be described as follows.

ALGORITHM 1: Arnoldi.

1. *Start*: Choose an initial vector v_1 with $\|v_1\|=1$.
2. *Iterate*: For $j=1, 2, \cdots$, do:
   $h_{i,j}=(Av_j, v_i)$, $i=1, 2, \cdots, j$,
   $\hat{v}_{j+1}=Av_j-\sum_{i=1}^{j}h_{i,j}v_i$,
   $h_{j+1,j}=\|\hat{v}_{j+1}\|$, and
   $v_{j+1}=\hat{v}_{j+1}/h_{j+1,j}$.

In practical implementation it is usually more suitable to replace the Gram-Schmidt algorithm of step 2 by the modified Gram-Schmidt algorithm [15]. If V_k is the $N\times k$

matrix whose columns are the l_2-orthonormal basis $\{v_1, v_2, \cdots, v_k\}$, then $H_k \equiv V_k^T A V_k$, is the upper $k \times k$ Hessenberg matrix whose entries are the scalars $h_{i,j}$ generated by Algorithm 1. If we call P_k the l_2-orthogonal projector onto K_k, and denote by A_k the section of A in K_k, i.e. the operator $A_k = P_k A_{|K_k}$, we notice that H_k is nothing but the matrix representation of A_k in the basis $\{v_1, v_2, \cdots, v_k\}$. Thus Arnoldi's original method [1] was a Galerkin method for approximating the eigenvalues of A by those of H_k [1], [12].

In order to solve the linear system

$$Ax = f, \tag{1}$$

by the Galerkin method using the l_2-orthogonal basis V_k, we seek an approximate solution x_k of the form $x_k = x_0 + z_k$, where x_0 is some initial guess to the solution x, and z_k is a member of the Krylov subspace $K_k \equiv \text{span}\{r_0, Ar_0, \cdots, A^{k-1}r_0\}$, with $r_0 = f - Ax_0$. Suppose that k steps of Algorithm 1 are carried out starting with $v_1 = r_0/\|r_0\|$. Then it is easily seen that the Galerkin condition that the residual vector $r_k \equiv f - Ax_k$ be l_2-orthogonal to K_k yields

$$z_k = V_k y_k \quad \text{where } y_k = H_k^{-1}\|r_0\| e_1$$

and e_1 is the unit vector $e_1 \equiv (1, 0, 0, \cdots, 0)^T$ [13]. Hence we can define the following Algorithm [13].

ALGORITHM 2: Full orthogonalization method.

1. *Start*: Choose x_0 and compute $r_0 = f - Ax_0$ and $v_1 = r_0/\|r_0\|$.
2. *Iterate*: For $j = 1, 2, \cdots, k$ do:
   $h_{i,j} = (Av_j, v_i)$, $i = 1, 2, \cdots, j$,
   $\hat{v}_{j+1} = Av_j - \sum_{i=1}^{j} h_{i,j} v_i$,
   $h_{j+1,j} = \|\hat{v}_{j+1}\|$, and
   $v_{j+1} = \hat{v}_{j+1}/h_{j+1,j}$.
3. *Form the solution*:
   $x_k = x_0 + V_k y_k$, where $y_k = H_k^{-1}\|r_0\| e_1$.

In practice, the number k of iterations in step 2 is chosen so that the approximate solution x_k will be sufficiently accurate. Fortunately, it is simple to determine a posteriori when k is sufficiently large without having to explictly compute the approximate solution because we can compute the residual norm of x_k thanks to the relation [13], [14]:

$$\|f - Ax_k\| = h_{k+1,k}|e_k^T y_k|. \tag{2}$$

Note, that if the algorithm stops at step k, then clearly it is unnecessary to compute the vector v_{k+1}.

Algorithm 2 has a number of important properties [14]:

- Apart from a multiplicative constant, the residual vector r_k of x_k is nothing but the vector v_{k+1}. Hence, the residual vectors produced by Algorithm 2 are l_2-orthogonal to each other.
- Algorithm 2 does not break down if and only if the degree of the minimal polynomial of v_1 is at least k and the matrix H_k is nonsingular.
- The process terminates in at most N steps.

Algorithm 2 generalizes a method developed by Parlett [11] for the symmetric case. It is also known to be mathematically equivalent to the ORTHORES algorithm developed by Young and Jea [9].

A difficulty with the full orthogonalization method is that it becomes increasingly expensive as the step number k increases. There are two distinct ways of avoiding this difficulty. The first is simply to restart the algorithm every m steps. The second is to truncate the l_2-orthogonalization process, by insisting that the new vector v_{i+1} be l_2-orthogonal to only the previous l vectors where l is some integer parameter. The resulting Hessenberg matrix H_k is then banded and the algorithm can be implemented in such a way as to avoid storing all previous but only the l most recent v_i's. The details on this Incomplete l_2-orthogonalization Method (IOM (l)), can be found in [14]. A drawback of these truncation techniques is the lack of any theory concerning the global convergence of the resulting method. Such a theory is difficult because there is no optimality property similar to that of the conjugate gradient method. In the next section we derive a method which we call GMRES based on Algorithm 1 to provide an approximate solution which satisfies an optimality property.

3. The generalized minimal residual (GMRES) algorithm.

3.1. The algorithm. The approximate solution of the form x_0+z, which minimizes the residual norm over z in K_k, can in principle be obtained by several known algorithms:

- The ORTHODIR algorithm of Jea and Young [9];
- Axelsson's method [2];
- the generalized conjugate residual method [4], [5].

However, if the matrix is indefinite these algorithms may break down or have stability problems. Here we introduce a new algorithm to compute the same approximate solution by using the basis generated by Arnoldi's method, Algorithm 1.

To describe the algorithm we start by noticing that after k steps of Arnoldi's method we have an l_2-orthonormal system V_{k+1} and a $(k+1)\times k$ matrix $\bar{H}_k$ whose only nonzero entries are the elements h_{ij} generated by the method. Thus $\bar{H}_k$ is the same as H_k except for an additional row whose only nonzero element is $h_{k+1,k}$ in the $(k+1, k)$ position. The vectors v_i and the matrix $\bar{H}_k$ satisfy the important relation:

$$AV_k = V_{k+1}\bar{H}_k. \tag{3}$$

Now we would like to solve the least squares problem:

$$\min_{z\in K_k} \|f - A[x_0+z]\| = \min_{z\in K_k} \|r_0 - Az\|. \tag{4}$$

If we set $z = V_k y$, we can view the norm to be minimized as the following function of y:

$$J(y) = \|\beta v_1 - AV_k y\| \tag{5}$$

where we have let $\beta = \|r_0\|$ for convenience. Using (3) we obtain

$$J(y) = \|V_{k+1}[\beta e_1 - \bar{H}_k y]\|. \tag{6}$$

Here, the vector e_1 is the first column of the $(k+1)\times(k+1)$ identity matrix. Recalling that V_{k+1} is l_2-orthonormal, we see that

$$J(y) = \|\beta e_1 - \bar{H}_k y\|. \tag{7}$$

Hence the solution of the least squares problem (4) is given by

$$x_k = x_0 + V_k y_k \tag{8}$$

where y_k minimizes the function $J(y)$, defined by (7), over $y \in R^k$.

The resulting algorithm is similar to the Full Orthogonalization Method, Algorithm 2, described earlier, the only difference being that the vector y_k used in step 3 for computing x_k is now replaced by the minimizer of $J(y)$. Hence we define the following structure of the method.

ALGORITHM 3: The generalized minimal residual method (GMRES).

1. *Start*: Choose x_0 and compute $r_0 = f - Ax_0$ and $v_1 = r_0/\|r_0\|$.
2. *Iterate*: For $j = 1, 2, \cdots, k, \cdots$, until satisfied do:
   $h_{i,j} = (Av_j, v_i)$, $i = 1, 2, \cdots, j$,
   $\hat{v}_{j+1} = Av_j - \sum_{i=1}^{j} h_{i,j} v_i$,
   $h_{j+1,j} = \|\hat{v}_{j+1}\|$, and
   $v_{j+1} = \hat{v}_{j+1}/h_{j+1,j}$.
3. *Form the approximate solution*:
   $x_k = x_0 + V_k y_k$, where y_k minimizes (7).

When using the GMRES algorithm we can easily use the Arnoldi matrix H_k for estimating the eigenvalues of A. This is particularly useful in the hybrid Chebyshev procedure proposed in [6].

It is clear that we face the same practical difficulties with the above GMRES method as with the Full Orthogonalization Method. When k increases the number of vectors requiring storage increases like k and the number of multiplications like $\frac{1}{2}k^2N$. To remedy this difficulty, we can use the algorithm iteratively, i.e. we can restart the algorithm every m steps, where m is some fixed integer parameter. This restarted version of GMRES denoted by GMRES(m) is described below.

ALGORITHM 4: GMRES(m).

1. *Start*: Choose x_0 and compute $r_0 = f - Ax_0$ and $v_1 = r_0/\|r_0\|$.
2. *Iterate*: For $j = 1, 2, \cdots, m$ do:
   $h_{i,j} = (Av_j, v_i)$, $i = 1, 2, \cdots, j$,
   $\hat{v}_{j+1} = Av_j - \sum_{i=1}^{j} h_{i,j} v_i$,
   $h_{j+1,j} = \|\hat{v}_{j+1}\|$, and
   $v_{j+1} = \hat{v}_{j+1}/h_{j+1,j}$.
3. *Form the approximate solution*:
   $x_m = x_0 + V_m y_m$, where y_m minimizes $\|\beta e_1 - \bar{H}_m y\|$, $y \in R^m$.
4. *Restart*:
   Compute $r_m = f - Ax_m$; if satisfied then stop
   else compute $x_0 := x_m$, $v_1 := r_m/\|r_m\|$ and go to 2.

Note that in certain applications we will not restart GMRES. Such is the case for example in the solution of stiff ODE's [7] and in the hybrid adaptive Chebyshev method [6].

3.2. Practical implementation. We now describe a few important additional details concerning the practical implementation of GMRES. Consider the matrix $\bar{H}_k$, and let us suppose that we want to solve the least squares problem:

$$\min_y \|\beta e_1 - \bar{H}_k y\|.$$

A classical way of solving such problems is to factor $\bar{H}_k$ into $Q_k R_k$ using plane rotations. This is quite simple to implement because of the special structure of $\bar{H}_k$. However, it is desirable to be able to update the factorization of $\bar{H}_k$ progressively as each column appears, i.e. at every step of the Arnoldi process. This is important because, as will be seen, it enables us to obtain the residual norm of the approximate

solution *without computing* x_k thus allowing us to decide when to stop the process without wasting needless operations.

We now show in detail how such a factorization can be carried out. In what follows, we let F_j represent the rotation matrix which rotates the unit vectors e_j and e_{j+1}, by the angle θ_i:

$$F_j = \begin{bmatrix} 1 & & & & & & \\ & 1 & & & & & \\ & & \cdot & & & & \\ & & & c_j & -s_j & & \\ & & & s_j & c_j & & \\ & & & & & 1 & \\ & & & & & & \cdot \\ & & & & & & & 1 \end{bmatrix} \leftarrow \text{row } j+1$$

where $c_j \equiv \cos(\theta_j)$, $s_j \equiv \sin(\theta_j)$.

Assume that the rotations F_i, $i = 1, \cdots, j$ have been previously applied to $\bar{H}_j$ to produce the following upper triangular matrix of dimension $(j+1) \times j$:

$$R_j = \begin{bmatrix} x & x & x & x & x & x \\ & x & x & x & x & x \\ & & x & x & x & x \\ & & & x & x & x \\ & & & & x & x \\ & & & & & x \\ 0 & \cdot & \cdot & \cdot & \cdot & 0 \end{bmatrix}.$$

The letter x stands for a nonzero element. At the next step the last column and row of $\bar{H}_{j+1}$ appear and are appended to the above matrix. In order to obtain R_{j+1} we must start by premultiplying the new column by the previous rotations. Once this is done we obtain a $(j+2) \times (j+1)$ matrix of the form

$$\begin{bmatrix} x & x & x & x & x & x & : & x \\ & x & x & x & x & x & : & x \\ & & x & x & x & x & : & x \\ & & & x & x & x & : & x \\ & & & & x & x & : & x \\ & & & & & x & : & x \\ 0 & \cdot & \cdot & \cdot & \cdot & 0 & : & r \\ 0 & \cdot & \cdot & \cdot & \cdot & 0 & : & h \end{bmatrix}.$$

The principal upper $(j+1) \times j$ submatrix of the above matrix is nothing but R_j, and h stands for $h_{j+2,j+1}$ which is not affected by the previous rotations. The next rotation will then consist in eliminating that element h in position $j+2, j+1$. This is achieved by the rotation F_{j+1} defined by

$$c_{j+1} \equiv r/(r^2+h^2)^{1/2},$$
$$s_{j+1} \equiv -h/(r^2+h^2)^{1/2}.$$

Note that the successive rotations F_j must also simultaneously be applied to the right side βe_1.

Thus, after k steps of the above process, we have achieved the following decomposition of $\bar{H}_k$:

$$Q_k \bar{H}_k = R_k$$

where Q_k is $(k+1)\times(k+1)$ and is the accumulated product of the rotation matrices F_j, while R_k is an upper triangular matrix of dimension $(k+1)\times k$, whose last row is zero. Since Q_k is unitary, we have:

$$J(y) = \|\beta e_1 - \bar{H}_k y\| = \|Q_k[\beta e_1 - \bar{H}_k y]\| = \|g_k - R_k y\|, \tag{9}$$

where $g_k \equiv Q_k \beta e_1$ is the transformed right-hand side. Since the last row of R_k is a zero row, the minimization of (9) is achieved by solving the upper triangular linear system whch results from removing the last row of R_k and the last component of g_k. This provides y_k and the approximate solution x_k is then formed by the linear combination (8).

We claimed earlier that it is possible to obtain the residual norm of the approximate solution x_k while performing the above factorization, without explicitly computing x_k. Indeed, notice that from the definition of $J(y)$, the residual norm is nothing but $J(y_k)$ which, from (9), is in turn equal to $\|g_k - R_k y_k\|$. But by construction of y_k, this norm is the absolute value of the last component of g_k. We have proved the following.

PROPOSITION 1. *The residual norm of the approximate solution x_k is equal to the $(k+1)$st component of the right-hand side g_k obtained by premultiplying βe_1 by the k successive rotations transforming $\bar{H}_k$ into an upper triangular matrix.*

Therefore, since g_k is updated at each step, the residual norm is available at every step of the QR factorization at no extra cost. This is very useful in the practical implementation of the algorithm because it will prevent us from taking unnecessary iterations while allowing us to avoid the extra computation needed to obtain x_k explicitly.

Next we describe an efficient implementation of the last step of GMRES. If we can show that we can obtain the residual vector as a combination of the Arnoldi vectors $v_1, \cdots, v_m$ and Av_m, then after step m we do not need v_{m+1}. Note that computing $\hat{v}_{m+1}$ and its norm costs $(2m+1)N$ multiplications, so elimination of its computation is a significant saving. Assume that the first $m-1$ Arnoldi steps have already been performed, i.e. that the first $m-1$ columns of $\bar{H}_m$ are available as well as the first m vectors v_i, $i=1, \cdots, m$. Since we will not normalize v_i at every step, we do not have explicitly the vectors v_i but rather the vectors $w_i = \mu_i v_i$ where μ_i are some *known* scaling coefficients.

All we need in order to be able to compute x_m is the matrix $\bar{H}_m$ and the vectors $v_1, \cdots, v_m$. Since the vectors v_i, $i=1, \cdots, m$, are already known, we need compute only the coefficients $h_{i,m}$, $i=1, \cdots, m+1$. Noting that $h_{i,m} = (Av_m, v_i)$, for $i \leqq m$ we see that these first m coefficients can be obtained as follows:

1. Compute Av_m, and
2. Compute the m inner-products (Av_m, v_i), $i=1, \cdots, m$.

Clearly, the scaling coefficients μ_i must be used in the above computations as v_i, $i=1, \cdots, m$, are only available as $w_i = \mu_i v_i$, where $\mu_i = \|w_i\|$. This determines the mth column of $\bar{H}_m$ except for the element $h_{m+1,m}$. We wish to compute this coefficient without having to compute w_{m+1}. By definition and the orthogonality of the v_i's

$$h_{m+1,m}^2 = \left\| Av_m - \sum_{i=1}^{m} h_{i,m} v_i \right\|^2 = \|Av_m\|^2 - \sum_{i=1}^{m} h_{i,m}^2. \tag{10}$$

Hence the last coefficient can be obtained from the $h_{i,m}$'s, $i=1,\cdots,m$, and the norm of Av_m.

Now we will show how to compute the residual vector $r_m = f - Ax_m$ from the v_i's, $i=1,\cdots,m$ and Av_m. This computation is necessary only when restarting. From (6) the residual vector can be expressed as

$$r_m = V_{m+1}[\beta e_1 - \bar{H}_m y_m]. \tag{11}$$

If we define $t \equiv [t_1, t_2, \cdots, t_{m+1}]^T \equiv \beta e_1 - \bar{H}_m y_m$, then

$$r_m = \left(\sum_{i=1}^{m} t_i v_i\right) + t_{m+1} v_{m+1} = \left(\sum_{i=1}^{m} t_i v_i\right) + t_{m+1} \frac{1}{h_{m+1,m}} \left[Av_m - \sum_{i=1}^{m} h_{i,m} v_i\right]$$

$$= \frac{t_{m+1}}{h_{m+1,m}} Av_m + \sum_{i=1}^{m} (t_i - t_{m+1} h_{i,m}/h_{m+1,m}) v_i.$$

It is to be expected that for large m, the alternative expression (10) for $h_{m+1,m}$ would be inaccurate as the orthogonality of the vectors v_i, on which it is based, is likely to be lost [11]. Moreover, in the restarted GMRES, the computation of r_m by (11) may be more time consuming than the explicit use of $r_m = f - Ax_m$. Therefore, it is not recommended to use the above implementation when m is large.

3.3. Comparison with other methods. From the previous description of GMRES, it is not clear whether or not this algorithm is more effective than GCR or ORTHODIR. Let us examine the computational costs of these three methods. We will denote by NZ the number of nonzero elements in A. We will evaluate the cost of computing the approximation x_k by GMRES. There are several possible implementations but we will refer to the one described in the previous section. If we neglect the cost of computing y_k, which is the solution of a least squares problem of size k, where k is usually much less than N, the total cost of computing x_k by GMRES can be divided in two parts:

- The computation of the Arnoldi vectors v_{j+1}, for $j=1,2,\cdots,k$. The jth step in this loop requires $(2j+1)N + \mathrm{NZ}$ multiplications, assuming that the vectors v_i are not normalized but that their norms are only computed and saved. The last step requires only $(k+1)N$ multiplications instead of $(2k+1)N$, i.e. kN fewer multiplications than the regular cost, as was shown in the previous section. Hence, the total number of mutiplications for this part is approximately $k(k+2)N + k\mathrm{NZ} - kN = k(k+1)N + k\mathrm{NZ}$.
- The formation of the approximate solution $x_0 + V_k y_k$, in step 3 requires kN multiplications.

The k steps of GMRES therefore require $k(k+2)N + k\mathrm{NZ}$ multiplications. Dividing by the total number of steps k, we see that each step requires $(k+2)N + \mathrm{NZ}$ multiplications on the average. In [5], it was shown that both GCR and ORTHODIR require on the average $\frac{1}{2}(3k+5)N + \mathrm{NZ}$ multiplications per step to produce the same approximation x_k. Therefore with the above implementation GMRES is always less expensive than either GCR or ORTHODIR. For large k savings will be nearly $\frac{1}{3}$.

The above comparison concerns the *nonrestarted* GMRES *algorithm.* Note that the notation adopted in [5] for the restarted versions of GCR and ORTHODIR differs slightly from ours in that GCR(m) has $m+1$ steps in each innerloop, while GMRES(m) has only m steps. Hence GMRES(m) is mathematically equivalent to GCR($m-1$). When we restart GMRES, we will need the residual vector after the m steps are completed. The residual vector can be obtained either explicitly as $f - Ax_m$ or, as will be described later, as a linear combination of Av_m and the v_i's, $i=1,\cdots,m$. Assuming

the latter, we will perform $(m+1)N$ extra multiplications. This will increase the average cost per step by $(1+1/m)N$ to $(m+3+1/m)N+\text{NZ}$. The corresponding cost per step of the restarted GCR and ORTHODIR is $\frac{1}{2}(3m+5)N+\text{NZ}$. Thus GMRES($m$) is more economical than GCR($m-1$) for $m>1$. Note that the above operation count for GCR($m-1$) and ORTHODIR($m-1$) does not include the computation of the norm of the residual vector which is required in the stopping criterion while for GMRES, we have shown earlier that this norm is available at every step at no extra cost. This remark shows that in fact the algorithms require the same number of operations when $m=1$.

For GMRES(m), it is clear that all we need to store is the v_i's, the approximate solution, and vector for Av_i, which means $(m+2)N$ storage locations. For large m, this is nearly half the $(2m+1)N$ storage required by both GCR and ORTHODIR. The comparison of costs is summarized in the following table in which GCR($m-1$) and GMRES(m) are the restarted versions of GCR and GMRES, using m steps in each innerloop. Note that the operation count of ORTHODIR($m-1$) is identical with that of GCR($m-1$) [5].

TABLE 1

Method	Multiplications	Storage
GCR($m-1$)	$[(3m+5)/2])N+\text{NZ}$	$(2m+1)N$
GMRES(m)	$(m+3+1/m)N+\text{NZ}$	$(m+2)N$

3.4. Theoretical aspects of GMRES. A question often raised in assessing iterative algorithms is whether they may break down. As we showed in the introduction, GCR can break down when A is not positive real, i.e. when its symmetric part is not positive definite. In this section we will show that GMRES cannot break down, regardless of the positiveness of A.

Initially, we assume that the first m Arnoldi vectors can be constructed. This will be the case if $h_{j+1,j}\neq 0$, $j=1,2,\cdots,m$. In fact if $h_{j+2,j+1}\neq 0$, the diagonal element $r_{j+1,j+1}$ of R_{j+1} obtained from the above algorithm satisfies:

$$r_{j+1,j+1}=(c_{j+1}r-s_{j+1}h_{j+2,j+1})=(r^2+h_{j+2,j+1}^2)^{1/2}>0.$$

Hence, the diagonal elements of R_m do not vanish and therefore the least squares problem (9) can always be solved, establishing that the algorithm *cannot break down* if $h_{j+1,j}\neq 0$, $j=1,\cdots,m$.

Thus the only possible potential difficulty is that during the Arnoldi process we encounter an element $h_{j+1,j}$ equal to zero. Assume that this actually happens at the jth step. Then since $h_{j+1,j}=0$ the vector v_{j+1} cannot be constructed. However, from Arnoldi's algorithm it is easily seen that we have the relation $AV_j=V_jH_j$ which means that the subspace K_j spanned by V_j is invariant. Notice that if A is nonsingular then H_j whose spectrum is a part of the spectrum of A is also nonsingular. The quadratic form (5) at the jth step becomes

$$J(y)=\|\beta v_1-AV_jy\|=\|\beta v_1-V_jH_jy\|=\|V_j[\beta e_1-H_jy]\|=\|\beta e_1-H_jy\|.$$

Since H_j is nonsingular the above function is minimum for $y=H_j^{-1}\beta e_1$ and the corresponding minimum norm is zero, i.e., the solution x_j is exact.

To prove that the converse is also true assume that x_j is the exact solution and that x_i, $i=1,2,\cdots,j-1$ are not, i.e. $r_j=0$ but $r_i\neq 0$ for $i=0,1,\cdots,j-1$. Then $r_j=0$ and from Proposition 1 we know that the residual norm is nothing but $s_je_{j-1}^Tg_{j-1}$, i.e.

the previous residual norm times s_j. Since the previous residual norm is nonzero by assumption, we must have $s_j = 0$ which implies $h_{j+1,j} = 0$, i.e. the algorithm breaks down and $\hat{v}_{j+1} = 0$ which proves the result.

Moreover, it is possible to show that $\hat{v}_{j+1} = 0$ and $\hat{v}_i \neq 0$ $i = 1, 2, \cdots, j$ is equivalent to the property that the degree of the minimal polynomial of the initial residual vector $r_0 = v_1$ is equal to j. Indeed assume the degree of the minimal polynomial of v_1 is j. This means that there exists a polynomial p_j of degree j, such that $p_j(A)v_1 = 0$, and p_j is the polynomial of lowest degree for which this is true. Therefore $K_{j+1} = \text{span}\{v_1, Av_1, \ldots, A^j v_1\}$ is equal to K_j. Hence the vector $\hat{v}_{j+1}$ which is a member of $K_{j+1} = K_j$ and is orthogonal to K_j is necessarily a zero vector. Moreover, if $\hat{v}_i = 0$ for $i \leqq j$ then there exists a polynomial p_i of degree i such that $p_i(A)v_1 = 0$ which contradicts the minimality of p_i.

To prove the converse assume that $\hat{v}_{j+1} = 0$ and $\hat{v}_i \neq 0$ $i = 1, 2, \cdots, j$. Then there exists a polynomial p_j of degree j such that $p_j(A)v_1 = 0$. Moreover, p_j is the polynomial of lowest degree for which this is true, otherwise we would have $\hat{v}_{i+1} = 0$, for some $i < j$ by the first part of this proof which is a contradiction.

PROPOSITION 2. *The solution x_j produced by* GMRES *at step j is exact if and only if the following four equivalent conditions hold*:

- (1) *The algorithm breaks down at step j.*
- (2) $\hat{v}_{j+1} = 0$.
- (3) $h_{j+1,j} = 0$.
- (4) *The degree of the minimal polynomial of the initial residual vector r_0 is equal to j.*

This uncommon type of breakdown is sometimes referred to as a "lucky" breakdown in the context of the Lanczos algorithm. Because the degree of the minimal polynomial of v_1 cannot exceed N for an N-dimensional problem, an immediate corollary follows.

COROLLARY 3. *For an $N \times N$ problem* GMRES *terminates in at most N steps.*

A consequence of Proposition 2 is that the restarted algorithm GMRES (m) does not break down. GMRES (m) would therefore constitute a very reliable algorithm if it always converged. Unfortunately this is not always the case, i.e. there are instances where the residual norms produced by the algorithm, although nonincreasing, do not converge to zero. In [5] it was shown that the GCR ($m-1$) method converges under the condition that A is positive real and so the same result is true for GMRES (m). It is easy to construct a counter-example showing that this result does not extend to indefinite problems, i.e. that the method may not converge if the symmetric part of A is not positive definite. In fact it is possible to show that the restarted GMRES method may be stationary. Consider GMRES (1) for the problem $Ax = f$, where

$$A = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}, \quad f = \begin{bmatrix} 1 \\ 1 \end{bmatrix}, \quad x_0 = 0,$$

which we considered in the introduction. The approximate solution x_1 minimizes the residual norm $\|f - Az\|$ where z is a vector of the form $z = \alpha f$. It is easily seen that $x_1 = 0$. Therefore the algorithm will provide a stationary sequence. Note that this is independent from the problem of breakdown. In fact GMRES will produce the solution in two steps but GMRES (1) never will.

Since the residual norm is minimized at every step of the method it is clear that it is nonincreasing. Intuitively, for m large enough the residual norm will be reduced by a sufficiently small ratio as to ensure convergence. Thus we would expect GMRES (m) to be convergent for sufficiently large m. However, note that ultimately

when $m = N$, the result is trivial, i.e. the method converges in one step. Thus, we will not attempt to show that the method GMRES (m) converges for sufficiently large m. On the other hand it is useful to show that if A is nearly positive real, i.e. when it has a small number of eigenvalues on the left half plane, then m need not be too large for convergence to take place.

In order to analyse this convergence, we let P_m be the space of all polynomials of degree $\leqq m$ and let σ represent the spectrum of A. The following result was established in [5] for the GCR algorithm and is a simple consequence of the optimality property.

PROPOSITION 4. *Suppose that A is diagonalizable so that $A = XDX^{-1}$ and let*

$$\varepsilon^{(m)} = \min_{p \in P_m, p(0)=1} \max_{\lambda_i \in \sigma} |p(\lambda_i)|. \tag{12}$$

Then the residual norm provided at the mth step of GMRES *satisfies*

$$\|r_{m+1}\| \leqq \kappa(X)\varepsilon^{(m)}\|r_0\|,$$

where $\kappa(X) = \|X\|\,\|X^{-1}\|$.

When A is positive real with symmetric part M, the following error bound can be derived from the proposition, see [5]:

$$\|r_m\| \leqq [1 - \alpha/\beta]^{m/2}\|r_0\|,$$

with $\alpha = (\lambda_{\min}(M))^2$, $\beta = \lambda_{\max}(A^TA)$. This proves the convergence of the GMRES (m) for all m when A is positive real [5].

When A is not positive real the above result is no longer true but we can establish the following explicit upper bound for $\varepsilon^{(m)}$.

THEOREM 5. *Assume that there are ν eigenvalues $\lambda_1, \lambda_2, \cdots, \lambda_\nu$ of A with nonpositive real parts and let the other eigenvalues be enclosed in a circle centered at C with $C > 0$ and having radius R with $C > R$. Then*

$$\varepsilon^{(m)} \leqq \left[\frac{R}{C}\right]^{m-\nu} \max_{j=\nu+1,N} \prod_{i=1}^{\nu} \frac{|\lambda_i - \lambda_j|}{|\lambda_i|} \leqq \left[\frac{D}{d}\right]^{\nu} \left[\frac{R}{C}\right]^{m-\nu} \tag{13}$$

where

$$D = \max_{i=1,\nu; j=\nu+1,N} |\lambda_i - \lambda_j| \quad \text{and} \quad d = \min_{i=1,\nu} |\lambda_i|.$$

Proof. Consider the particular class of polynomials defined by $p(z) = r(z)q(z)$ where $r(z) = (1 - z/\lambda_1)(1 - z/\lambda_2)\cdots(1 - z/\lambda_\nu)$ and $q(z)$ is an arbitrary polynomial of degree $\leqq m - \nu$, such that $q(0) = 1$. Clearly, since $p(0) = 1$ and $p(\lambda_i) = 0$, $i = 1, \cdots, \nu$, we have

$$\varepsilon^{(m)} \leqq \max_{j=\nu+1,N} |p(\lambda_j)| \leqq \max_{j=\nu+1,N} |r(\lambda_j)| \max_{j=\nu+1,N} |q(\lambda_j)|.$$

It is easily seen that

$$\max_{j=\nu+1,N} |r(\lambda_j)| = \max_{j=\nu+1,N} \prod_{i=1}^{\nu} \frac{|\lambda_i - \lambda_j|}{|\lambda_i|} \leqq (D/d)^{\nu}.$$

Moreover, by the maximum principle, the maximum of $|q(z)|$ for z belonging to the set $\{\lambda_j\}_{j=\nu+1,N}$ is no larger than its maximum over the circle that encloses that set. Taking the polynomial $q(z) = [(C - z)/C]^{m-\nu}$ whose maximum modulus on the circle is $(R/C)^{m-\nu}$ yields the desired result. □

A similar result was shown by Chandra [3] for the symmetric indefinite case. Note that when the eigenvalues of A are all real then the maximum of the product term in

the second part of inequality (13) satisfies

$$\max_{j=\nu+1,N} \prod_{i=1}^{\nu} \frac{|\lambda_i - \lambda_j|}{|\lambda_i|} = \prod_{i=1}^{\nu} \frac{|\lambda_i - \lambda_N|}{|\lambda_i|}$$

where λ_N is the largest eigenvalue of A. A simple consequence of the above theorem is the following corollary.

COROLLARY 6. *Under the assumptions of Proposition* 4 *and Theorem* 5, GMRES (m) *converges for any initial vector* x_0 *if*

$$m > \nu \operatorname{Log}\left[\frac{DC}{dR}\kappa(X)^{1/\nu}\right] \Big/ \operatorname{Log}\left[\frac{C}{R}\right].$$

A few comments are in order. First note that, in general, the upper bound (13) is not likely to be sharp, and so convergence may take place for m much smaller than would be predicted by the result. Second, observe that the minimal m that ensures convergence is related only to the eigenvalue distribution and the condition number of X. In particular, it is independent of the problem-size N. Third, it may very well happen that the minimal m would be larger than N, in which case the information provided by the corollary would be trivial since the method is exact for $m = N$.

4. Numerical experiments. In this section we report a few numerical experiments comparing the performances of GMRES with other conjugate gradient-like methods. The tests were performed on a VAX-11/780 using double precision corresponding to a unit round off of nearly 6.93×10^{-18}. The GMRES (k) algorithm used in the following tests computes explicitly the last vector v_{k+1} of each outer iteration, i.e. it does not implement the modification described at the end of § 3.2.

The test problem was derived from the five point discretization of the following partial differential equation which was described in H. Elman's thesis [5]:

$$-(bu_x)_x - (cu_x)_x + du_x + (du)_x + eu_y + (eu)_y + fu = g$$

on the unit square, where

$$b(x, y) = e^{-xy}, \qquad c(x, y) = e^{xy} d(x, y) = \beta(x+y),$$
$$e(x, y) = \gamma(x+y) \quad \text{and} \quad f(x, y) = 1./(1+x+y)$$

subject to the Dirichlet boundary conditions $u = 0$ on the boundary. The right-hand side g was chosen so that the solution was known to be $xe^{xy} \sin(\pi x) \sin(\pi y)$. The parameters β and γ are useful for changing the degree of symmetry of the resulting linear systems. Note that the matrix A resulting from the discretization remains positive real independent of these parameters.

We will denote by n the number of interior nodes on each side of the square and by $h = 1/(n+1)$ the mesh size. In the first example we took $n = 48$, $\gamma = 50$ and $\beta = 1$. This yielded a matrix of dimension $N = 2304$. The system was preconditioned by the *MILU* preconditioning applied on the right, i.e. we solved $AM^{-1}(Mx) = f$ where M was some approximation to A^{-1} provided by an approximate LU factorization of A see [5]. The process was stopped as soon as the residual norm was reduced by a factor of $\varepsilon = 10^{-6}$. The following plot compares the results obtained for GCR (k), GMRES (k), and ORTHOMIN (k) for some representative values of k.

The plot shows that ORTHOMIN (k) did not converge for $k = 1$ and $k = 5$ on this example. In fact, we observed that it exhibited the same nonconverging behaviour for all values of k between 1 and 5. Another interesting observation is that GMRES (5)

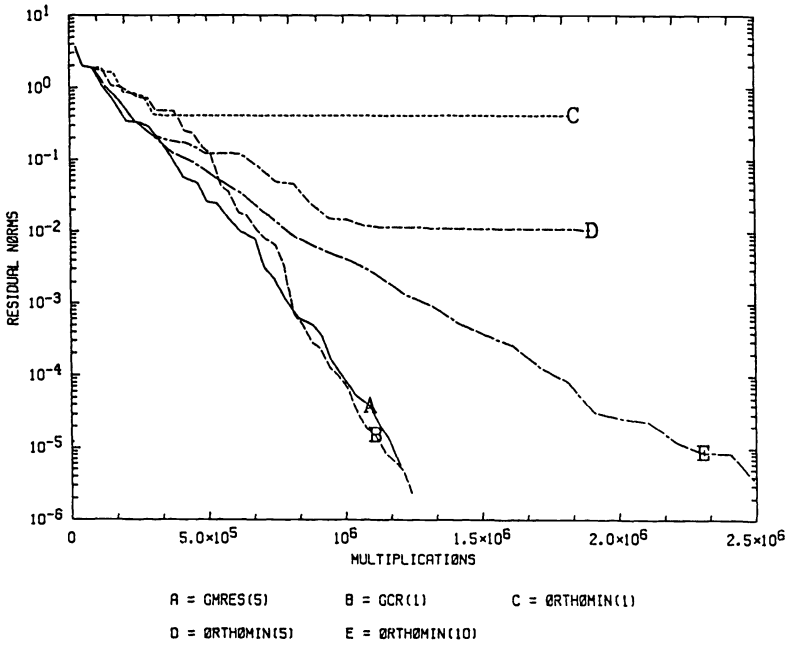


FIG. 4.1. $n = 48$, *MILU preconditioning*, $\gamma = 50.$, $\beta = 1.$

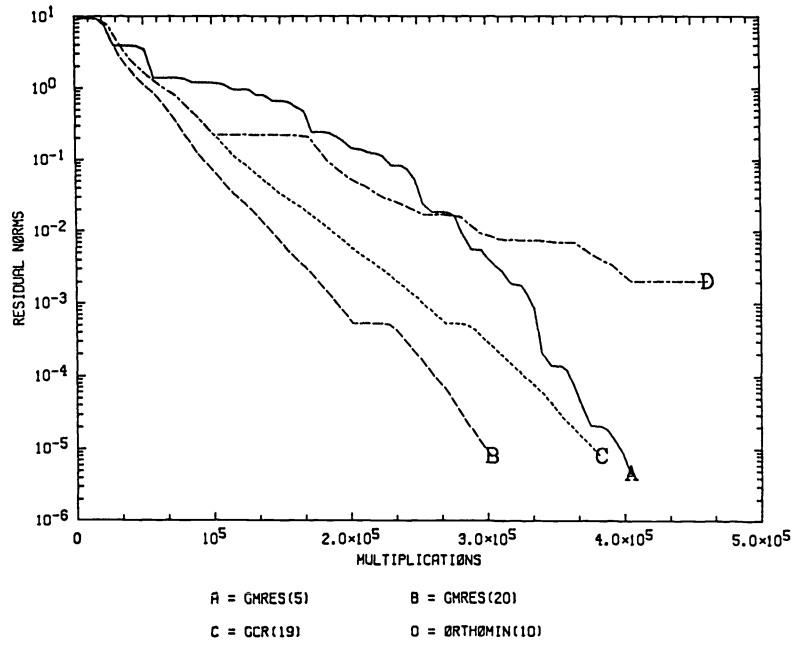


FIG. 4.2. $n = 18$, *MILU preconditioning*, $\gamma = 50.$, $\beta = -20.$

performed almost as well as GCR (1). Note that the value $k = 5$ yielded the best possible result that was obtained for all reasonable choices of k and similarly GCR (1) corresponded to the best possible performance for GCR (k).

It is worth pointing out that for moderate accuracy ($\varepsilon \geqq 10^{-2}$), GMRES (5) was slightly better than GCR (1). Finally, we should indicate that the reason why ORTHOMIN performed so badly in this example is that the *preconditioned system* is not positive real. The *MILU* preconditioning seems to be more prone to such peculiarities than the simpler *ILU* preconditioning. In fact for this example ORTHOMIN (1) performed very well when the *ILU* preconditioning was used.

In the next test we took $n = 18$ which yielded a matrix of smaller dimension $N = 324$, and $\gamma = 50.$, $\beta = -20$. The main purpose of this experiment was to show that there are instances where using a large parameter m is important. Here again we used the *MILU* preconditioning and the stopping tolerance was $\varepsilon = 10^{-6}$. This example was more difficult to treat. ORTHOMIN (k) diverged for all values of k between 1 and 10. Also GCR (1), GCR (2) and GCR (3) diverged as well as their equivalent versions GMRES (k), $k = 2, 3, 4$. The process GMRES (k) started to converge with $k = 5$ and improved substantially as k increased. The best performance was realized for larger values of k. The following plot shows the results obtained for GMRES (5), GMRES (20) and ORTHOMIN (10). In order to be able to appreciate the gains made by GMRES (20) versus its equivalent version GCR (19), we also plotted the results for GCR (19). Note that we saved nearly 25% in the number of multiplications but also almost half the storage which was quite important here since we needed to keep 22 vectors in memory versus 39 for GCR (19).

REFERENCES

[1] W. E. ARNOLDI., *The principle of minimized iteration in the solution of the matrix eigenvalue problem*, Quart. Appl. Math., 9 (1951), pp. 17-29.

[2] O. AXELSSON, *Conjugate gradient type methods for unsymmetric and inconsistent systems of linear equations*, Lin. Alg. Appl., 29 (1980), pp. 1-16.

[3] R. CHANDRA, *Conjugate gradient methods for partial differential equations*, Ph.D. thesis, Computer Science Dept., Yale Univ., New Haven, CT, 1978.

[4] S. C. EISENSTAT, H. C. ELMAN AND M. H. SCHULTZ, *Variational iterative methods for nonsymmetric systems of linear equations*, SIAM J. Numer. Anal., 20 (1983), pp. 345-357.

[5] H. C. ELMAN, *Iterative methods for large sparse nonsymmetric systems of linear equations*, Ph.D. thesis, Computer Science Dept., Yale Univ., New Haven, CT, 1982.

[6] H. C. ELMAN, Y. SAAD AND P. SAYLOR, *A hybrid Chebyshev Krylov subspace algorithm for solving nonsymmetric systems of linear equations*, Technical Report YALU/DCS/TR-301, Yale Univ., New Haven, CT, 1984.

[7] W. C. GEAR AND Y. SAAD, *Iterative solution of linear equations in* ODE *codes*, this Journal, 4 (1983), pp. 583-601.

[8] A. L. HAGEMAN AND D. M. YOUNG, *Applied Iterative Methods*, Academic Press, New York, 1981.

[9] K. C. JEA AND D. M. YOUNG, *Generalized conjugate gradient acceleration of nonsymmetrizable iterative methods*, Lin. Alg. Appl., 34 (1980), pp. 159-194.

[10] C. C. PAIGE AND M. A. SAUNDERS, *Solution of sparse indefinite systems of linear equations*, SIAM J. Numer. Anal., 12 (1975), pp. 617-624.

[11] B. N. PARLETT, *A new look at the Lanczos algorithm for solving symmetric systems of linear equations*, Lin. Alg. Appl., 29 (1980), pp. 323-346.

[12] Y. SAAD, *Variations on Arnoldi's method for computing eigenelements of large unsymmetric matrices*, Lin. Alg. Appl., 34 (1980), pp. 269-295.

[13] ———, *Krylov subspace methods for solving large unsymmetric linear systems*, Math. Comput., 37 (1981), pp. 105-126.

[14] ———, *Practical use of some Krylov subspace methods for solving indefinite and unsymmetric linear systems*, this Journal, 5 (1984), pp. 203-228.

[15] G. W. STEWART, *Introduction to Matrix Computations*, Academic Press, New York, 1973.

[16] P. K. W. VINSOME, ORTHOMIN, *an iterative method for solving sparse sets of simultaneous linear equations*, in Proc. Fourth Symposium on Reservoir Simulation, Society of Petroleum Engineers of AIME, 1976, pp. 149-159.

SIAM J. SCI. STAT. COMPUT.
Vol. 7, No. 3, July 1986

012

A SECOND-ORDER ACCURATE PRESSURE-CORRECTION SCHEME FOR VISCOUS INCOMPRESSIBLE FLOW*

J. VAN KAN†

Abstract. A pressure correction method for (time-dependent) viscous incompressible flow is presented that is second order accurate in time and space. The order of accuracy is proved for a model scheme and demonstrated for a numerical example. A practical application is given.

Key words. viscous time-dependent flow, finite differences, fractional steps, ADI-methods.

AMS(MOS) subject classifications. 65M05, 76D05

1. Introduction. Pressure correction methods have been introduced as a useful way to significantly reduce the computational cost of explicit and implicit calculations of time-dependent incompressible viscous flow in the velocity-pressure formulation. (see [1], [3], [5], [8] and [12]); the method is called by various names, such as fractional step method or Chorin's method. So far, little attention has been paid to improving the accuracy of this method, which in earlier works is $O(\Delta t+(\Delta x)^2)$.

Goda [5] presents a method for 3D calculations that has this accuracy and only Braza [1] hints at the possibility of improving the accuracy in time to $O(\Delta t^2)$. The reason she does not succeed in achieving $O(\Delta t^2)$ accuracy is, that in the ADI scheme she developed, she did not approximate the convective terms to $O(\Delta t^2)$ accuracy.

In this paper we shall present an ADI scheme with pressure correction which is second order consistent in both space and time. Ordinarily one would expect $O(\Delta t^2+\Delta x^2)$ accuracy in the solution under these circumstances and we shall present practical calculations that make second order accuracy in time plausible. We shall show that our proposed pressure correction method in a system of "constrained" ODE's similar to the Navier-Stokes problem under consideration under reasonably weak assumptions lead to a solution with $O(\Delta t^2)$ accuracy. We also show that in a linearized simplified case pressure correction does not affect the unconditional stability of the underlying scheme. Finally we shall present a practical application of the presented schemes.

2. Development of a pressure correction scheme. We consider the 2D Navier-Stokes equations for incompressible viscous flow

$$\begin{aligned} &\frac{\partial u}{\partial t}+(u\cdot\nabla)u+\nabla p=\mathrm{Re}^{-1}\Delta u,\\ &\operatorname{div} u=0, \end{aligned} \tag{2.1}$$

in which $u=\binom{u_1}{u_2}$ is the (scaled) velocity, p the (scaled) pressure and $\mathrm{Re}=Ud/\nu$ the Reynolds number, with U a reference velocity, d a reference length and ν the kinematic viscosity.

In the first stage of the development we will take a time centered or Crank-Nicolson type scheme

$$\begin{aligned} &\frac{u^{n+1}-u^n}{\Delta t}+\frac{1}{2}[(u^{n+1}\cdot\nabla)u^{n+1}+(u^n\cdot\nabla)u^n+\nabla p^{n+1}+\nabla p^n]=\frac{1}{2}\mathrm{Re}^{-1}\Delta(u^{n+1}+u^n),\\ &\operatorname{div} u^{n+1}=0, \end{aligned} \tag{2.2}$$

in which superscripts denote levels in time.

* Received by the editors February 14, 1985.

† Technische Hogeschool Delft, Delft University of Technology, Delft, the Netherlands.

For the space discretization we use a staggered grid as introduced by Harlow and Welch [8]. For simplicity the mesh-size is taken as uniform. The velocities and pressures are calculated in positions indicated in Figs. 2.1 and 2.2. Define the following operators:

$$D_x^1\phi_{i,j}=\frac{\phi_{i+1,j}-\phi_{i-1,j}}{2\Delta x}, \qquad D_x^2\phi_{i,j}=\frac{\phi_{i+1,j}-2\phi_{i,j}+\phi_{i-1,j}}{(\Delta x)^2},$$

$$D_y^1\phi_{i,j}=\frac{\phi_{i,j+1}-\phi_{i,j-1}}{2\Delta y}, \qquad D_y^2\phi_{i,j}=\frac{\phi_{i,j+1}-2\phi_{i,j}+\phi_{i,j-1}}{(\Delta y)^2},$$

$$\nabla_x\phi_{i,j}=\frac{\phi_{i+1,j}-\phi_{i,j}}{\Delta x}, \qquad \Delta_x\phi_{i,j}=\frac{\phi_{i,j}-\phi_{i-1,j}}{\Delta x},$$

$$\nabla_y\phi_{i,j}=\frac{\phi_{i,j+1}-\phi_{i,j}}{\Delta y}, \qquad \Delta_y\phi_{i,j}=\frac{\phi_{i,j}-\phi_{i,j-1}}{\Delta y},$$

$$\bar{u}_{1,i,j}=\tfrac{1}{4}[u_{1,i,j+1}+u_{1,i,j}+u_{1,i-1,j}+u_{1,i-1,j+1}],$$

$$\bar{u}_{2,i,j}=\tfrac{1}{4}[u_{2,i,j}+u_{2,i,j-1}+u_{2,i+1,j}+u_{2,i+1,j-1}].$$

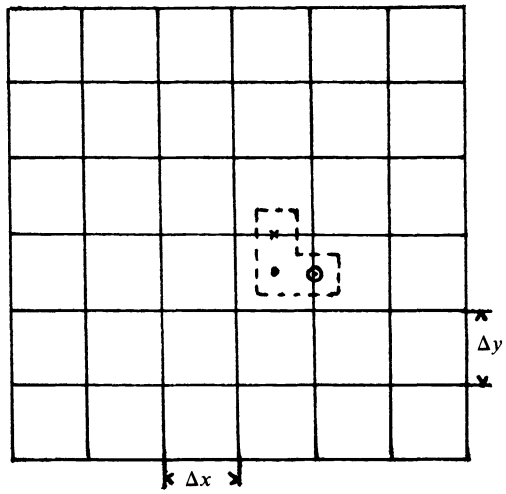


FIG. 2.1. *Staggered grid with* (i, j)*th tile.*

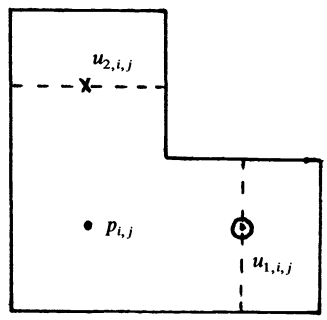


FIG. 2.2. (i, j)*th tile.*

A space discretization of (2.2) is, putting $\alpha = (i, j)$,

$$
\begin{aligned}
&\frac{u_{1,\alpha}^{n+1} - u_{1,\alpha}^{n}}{\Delta t} + \frac{1}{2}[u_{1,\alpha}^{n+1} D_x^1 u_{1,\alpha}^{n+1} + \bar{u}_{2,\alpha}^{n+1} D_y^1 u_{1,\alpha}^{n+1} + u_{1,\alpha}^{n} D_x^1 u_{1,\alpha}^{n} \\
&\qquad + \bar{u}_{2,\alpha}^{n} D_y^1 u_{1,\alpha}^{n} + \nabla_x p_\alpha^{n+1} + \nabla_x p_\alpha^{n}] \\
&\qquad = \frac{1}{2}\mathrm{Re}^{-1}[D_x^2(\bar{u}_{1,\alpha}^{n+1} + u_{1,\alpha}^{n}) + D_y^2(u_{1,\alpha}^{n+1} + u_{1,\alpha}^{n})], \\
&\frac{u_{2,\alpha}^{n+1} - u_{2,\alpha}^{n}}{\Delta t} + \frac{1}{2}[u_{1,\alpha}^{n+1} D_x^1 u_{2,\alpha}^{n+1} + u_{2,\alpha}^{n+1} D_y^1 u_{2,\alpha}^{n+1} + \bar{u}_{1,\alpha}^{n} D_x^1 u_{2,\alpha}^{n} \\
&\qquad + u_{2,\alpha}^{n} D_y^1 u_{2,\alpha}^{n} + \nabla_y p_\alpha^{n+1} + \nabla_y p_\alpha^{n}] \\
&\qquad = \frac{1}{2}\mathrm{Re}^{-1}[D_x^2(u_{2,\alpha}^{n+1} + u_{2,\alpha}^{n}) + D_y^2(u_{2,\alpha}^{n+1} + u_{2,\alpha}^{n})], \\
&\Delta_x u_{1,\alpha}^{n+1} + \Delta_y u_{2,\alpha}^{n+1} = 0.
\end{aligned} \tag{2.3}
$$

It is a straightforward exercise to show that this scheme has $O((\Delta t)^2 + (\Delta x)^2 + (\Delta y)^2)$ consistency.

We introduce the following operator notation for brevity:

$$
\left.
\begin{aligned}
A_{1,\alpha}^{n} &= u_{1,\alpha}^{n} D_x^1 + \bar{u}_{2,\alpha}^{n} D_y^1, \\
A_{2,\alpha}^{n} &= \bar{u}_{1,\alpha}^{n} D_x^1 + u_{2,\alpha}^{n} D_y^1,
\end{aligned}
\right\} \text{the convection operators,}
$$

$$
B = \mathrm{Re}^{-1}[D_x^2 + D_y^2], \qquad \text{the diffusion operator,}
$$

and rewrite (2.3) to obtain

$$
\begin{aligned}
&u_{1,\alpha}^{n+1} + \tfrac{1}{2}\Delta t\{[A_{1,\alpha}^{n+1} - B]u_{1,\alpha}^{n+1} + \nabla_x p_\alpha^{n+1}\} = u_{1,\alpha}^{n} - \tfrac{1}{2}\Delta t\{[A_{1,\alpha}^{n} - B]u_{1,\alpha}^{n} + \nabla_x p_\alpha^{n}\}, \\
&u_{2,\alpha}^{n+1} + \tfrac{1}{2}\Delta t\{[A_{2,\alpha}^{n+1} - B]u_{2,\alpha}^{n+1} + \nabla_x p_\alpha^{n+1}\} = u_{2,\alpha}^{n} - \tfrac{1}{2}\Delta t\{[A_{2,\alpha}^{n} - B]u_{2,\alpha}^{n} + \nabla_y p_\alpha^{n}\}, \\
&\Delta_x u_{1,\alpha}^{n+1} + \Delta_y u_{2,\alpha}^{n+1} = 0.
\end{aligned} \tag{2.4}
$$

Scheme (2.4) requires each time step the solution of a nonlinear system of a structure shown in Fig. 2.3.

A direct solution method (using some linearization for the convective term) would result in an enormous fill-in and is not feasible for this problem. We therefore use the

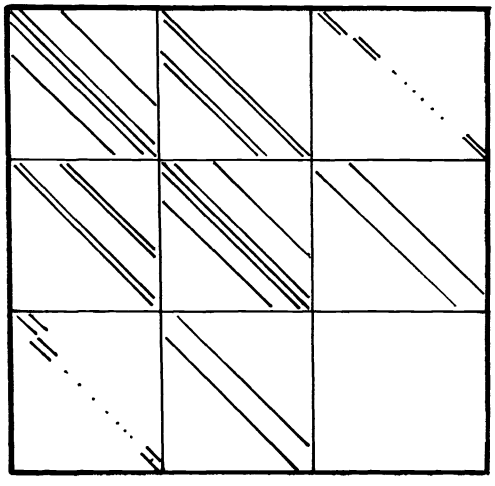


FIG. 2.3. *Sparsity pattern of scheme* (2.4).

pressure correction method. First the pressure at time t_{n+1} is replaced by the pressure at t_n and later on we correct the velocities, requiring that the discretized divergence will be zero. This reduces the complexity of the problem to that of Fig. 2.4.

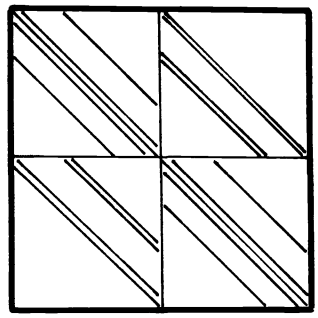


FIG. 2.4. *Sparsity pattern of remaining problem.*

Next, we construct an ADI type scheme for the remaining problem by splitting the operators $A_{1,\alpha}$, $A_{2,\alpha}$ and B (contrary to Braza [1], who only splits the operator B). This then will result in an ADI scheme with the accuracy claimed.

3. Approximation of boundary values. For simplicity we restrict ourselves to rectangular regions, and we suppose that at a boundary u_β or $\partial u_\beta/\partial n$ is prescribed for $\beta = 1, 2$. For the pressure there are no physical boundary conditions and the method we shall derive will not need them numerically either. As has been remarked in [2] this is easy, straightforward and correct, yet in the literature one often sees artificial boundary conditions for the pressure, which may easily degrade the accuracy. Fortunately, the effect on the velocities is an order lower, so that in practice one will often get away with it. If we are to obtain $O(\Delta t^2)$ accuracy however, we shall have no option but to let the boundary conditions for the pressure follow from those for the velocities.

If a prescribed velocity at a boundary has no point on that boundary (i.e. u_1 on horizontal boundaries and u_2 on vertical boundaries) the approximation is as in Fig. 3.1, using a virtual point

$$u_{2,\text{wall}} = \tfrac{1}{2}(u_{2,0,j} + u_{2,1,j}). \tag{3.1}$$

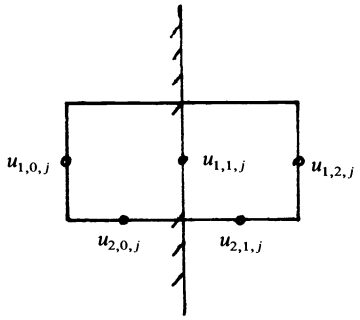


FIG. 3.1.

The normal derivatives are approximated as follows, again using virtual points,

$$\left.\frac{\partial u_1}{\partial x}\right|_{\text{wall}} = D_x^1 u_{1,1,j}, \qquad \left.\frac{\partial u_2}{\partial x}\right|_{\text{wall}} = \Delta_x u_{2,1,j}. \tag{3.2}$$

It will be noted that both approximations (3.1) and (3.2) are $O(\Delta x^2)$ consistent. The approximation of the boundary values is somewhat more complicated in the ADI schemes when the inflow conditions are time dependent. This problem shall be dealt with in due course.

4. A Crank–Nicolson scheme with pressure correction. The pressure correction idea is conveniently explained using as an analogue to the Navier-Stokes equations the following system of ordinary differential equations with constraints

$$\dot{x} = f(x) + L^T p, \quad Lx = g(t), \quad x(t_0) = x_0, \tag{4.1}$$

in which $x \in R^k, f: R^k \to R^k$ with appropriate smoothness properties, $p \in R^m$, $g: [0, \infty) \to R^m$ with appropriate smoothness properties, $L \in \mathcal{L}(R^k, R^m)$, $m < k$. If we discretize (2.1) in space but not in time (method of lines) we obtain a system of type (4.1). The term $g(t)$ is caused by the (time-dependent) boundary conditions.

Equation (4.1) is equivalent to

$$\dot{x} = Pf(x) + L^T(LL^T)^{-1}\dot{g}, \tag{4.2}$$

provided that $(LL^T)^{-1}$ exists, and if we define the projection operator P as follows:

$$P = I - L^T(LL^T)^{-1}L.$$

A Crank-Nicolson type scheme for (4.1) would be

$$\frac{x^{n+1} - x^n}{\Delta t} = \frac{1}{2}\{f(x^{n+1}) + f(x^n)\} + \frac{1}{2}L^T(p^{n+1} + p^n), \tag{4.3a}$$

$$Lx^{n+1} = g^{n+1}, \tag{4.3b}$$

$$x^0 = x(t_0), \qquad p^0 = (LL^T)^{-1}(\dot{g}(t_0) - f(x^0)). \tag{4.3c}$$

Elimination of the term $\frac{1}{2}L^T(p^{n+1} + p^n)$ yields

$$\frac{x^{n+1} - x^n}{\Delta t} = \frac{1}{2}P\{f(x^{n+1}) + f(x^n)\} + L^T(LL^T)^{-1}\frac{g^{n+1} - g^n}{\Delta t} \tag{4.4}$$

which is of course an $O(\Delta t^2)$ consistent approximation to (4.2).

The nonlinear equations (4.3) are similar to scheme (2.4) and we now describe the pressure correction idea with respect to scheme (4.3), because this will show how to graft a pressure correction step onto the Crank-Nicolson scheme (2.4).

We first calculate an approximation $\tilde{x}^{n+1}$ that does not necessarily satisfy (4.3b), using only p^n:

$$\frac{\tilde{x}^{n+1} - x^n}{\Delta t} = \frac{1}{2}\{f(\tilde{x}^{n+1}) + f(x^n)\} + L^T p^n. \tag{4.5a}$$

We calculate x^{n+1} and p^{n+1} as follows:

$$\frac{x^{n+1} - \tilde{x}^{n+1}}{\Delta t} = \frac{1}{2}L^T(p^{n+1} - p^n), \tag{4.5b}$$

$$Lx^{n+1} = g^{n+1}. \tag{4.5c}$$

If we multiply (4.5b) by L we obtain using (4.5c)

$$\frac{g^{n+1} - L\tilde{x}^{n+1}}{\Delta t} = \frac{1}{2}LL^T(p^{n+1} - p^n). \tag{4.6}$$

From this we can calculate p^{n+1} (that is if LL^T is invertible) and finally on substitution of this result into (4.5b) we obtain x^{n+1}. The first remark that is in order is that if (4.3)

represents a space discretization of the Navier–Stokes equation with boundary conditions for the velocities, then (4.6) represents a discretization of a Poisson equation for the pressure that already includes boundary conditions.

The second remark is that we have to investigate the effect of pressure correction on convergence and stability. For clarity we will treat those subjects for system (4.1). This will show the effect of pressure correction on the underlying scheme.

5. The effect of pressure correction on consistency and global accuracy. We consider the system of ODEs (4.1) and we call the numerical solution of the CN scheme (4.3) at time $t_0+n\Delta t$ v^n $(\in R^k)$ and q^n $(\in R^m)$ and the solution of the PC scheme (4.5) u^n $(\in R^k)$ and p^n $(\in R^m)$.

We shall give a presentation that will finally result in a global accuracy of the solution of the PC scheme of $O(\Delta t^2)$ at the same time resolving the consistency question. We adopt two hypotheses:

(5.1a) Uniform boundedness in finite time. We integrate up to T. Then $\|u^n\|$ and $\|v^n\|$ (and consequently $\|p^n\|$ and $\|q^n\|$) are assumed to be bounded uniformly in n and Δt with $n\Delta t \leqq T$. Say $\|u^n\|$ and $\|v^n\| < M$.

(5.1b) $f(u)$ is uniformly Lipschitz (with constant K) for $\|u\| < M$: $\|f(u)-f(v)\| < K\|u-v\|$ if $\|u\| < M$.

Hypothesis (a) is effectively a stability requirement. We shall prove stability in the linear case (see § 6). Unless otherwise stated the norms on R^k and R^m will be the Euclidean norm for vectors and the induced operator norm for matrices.

Before we state our results we first reformulate (4.5) slightly by adding (4.5b) to (4.5a) to obtain

$$\frac{u^{n+1}-u^n}{\Delta t}=\frac{1}{2}[f(\tilde{u}^{n+1})+f(u^n)]+\frac{1}{2}L^T[p^{n+1}+p^n]. \tag{5.2}$$

Next we define the difference between the CN and PC solutions

$$\varepsilon^n = u^n - v^n, \qquad \eta^n = p^n - q^n. \tag{5.3}$$

We can obtain the following global estimate for $\|\varepsilon^n\|$:

THEOREM 5.4. *There exists M_1 independent of Δt and n such that* $\|\varepsilon^n\| \leqq 2e^{3TM_1}\Delta t M_1 \sum_{j=0}^{n}\|u^j-\tilde{u}^j\|$, *$\forall n$ such that $n\Delta t < T$, $\forall \Delta t$ such that $\Delta t < 1/2M_1$.*

Proof. Subtracting the equation for the CN solution given by

$$\begin{aligned}&\frac{v^{n+1}-v^n}{\Delta t}=\frac{1}{2}[f(v^{n+1})+f(v^n)]+\frac{1}{2}L^T[q^{n+1}+q^n],\\ &Lv^{n+1}=g^{n+1},\end{aligned} \tag{5.5}$$

from (5.2) we obtain

$$\begin{aligned}&\varepsilon^{n+1}=\varepsilon^n+\tfrac{1}{2}\Delta t[f(\tilde{u}^{n+1})-f(v^{n+1})+f(u^n)-f(v^n)]+\tfrac{1}{2}\Delta t L^T(\eta^{n+1}+\eta^n),\\ &L\varepsilon^{n+1}=0.\end{aligned} \tag{5.6}$$

From (5.2), (5.5) and $Lu^{n+1}=g^{n+1}$ we may deduce

$$p^{n+1}+p^n=-(LL^T)^{-1}\left[L\{f(\tilde{u}^{n+1})+f(u^n)\}-2\frac{g^{n+1}-g^n}{\Delta t}\right], \tag{5.7a}$$

$$q^{n+1}+q^n=-(LL^T)^{-1}\left[L\{f(v^{n+1})+f(v^n)\}-2\frac{g^{n+1}-g^n}{\Delta t}\right]. \tag{5.7b}$$

From this we get

$$L^T(\eta^{n+1}+\eta^n) = -L^T(LL^T)^{-1}L[f(\tilde{u}^{n+1})+f(u^n)-f(v^{n+1})-f(v^n)], \tag{5.8}$$

hence, with $P = I - L^T(LL^T)^{-1}L$ we obtain by substitution of (5.8) into (5.6)

$$\begin{aligned}\varepsilon^{n+1} &= \varepsilon^n + \tfrac{1}{2}\Delta t P[f(\tilde{u}^{n+1})-f(v^{n+1})+f(u^n)-f(v^n)] \\ &= \varepsilon^n + \tfrac{1}{2}\Delta t P[f(u^{n+1})-f(v^{n+1})+f(u^n)-f(v^n)+f(\tilde{u}^{n+1})-f(u^{n+1})].\end{aligned}$$

By the Lipschitz condition this yields

$$\|\varepsilon^{n+1}\| \leqq \|\varepsilon^n\| + \tfrac{1}{2}\Delta t\|P\|K\{\|\varepsilon^{n+1}\| + \|\varepsilon^n\| + \|\tilde{u}^{n+1}-u^{n+1}\|\},$$

and since $\|P\| = 1$, we have for $\Delta t < 2/K$, putting $M_1 = \frac{1}{2}K$

$$(1-\Delta t M_1)\|\varepsilon^{n+1}\| \leqq (1+\Delta t M_1)\|\varepsilon^n\| + \Delta t M_1\|u^{n+1}-\tilde{u}^{n+1}\|. \tag{5.9}$$

We shall show by induction that

$$\|\varepsilon^n\| \leqq \frac{\Delta t M_1}{1-\Delta t M_1}\sum_{j=1}^{n} c^{n-j}\|u^j-\tilde{u}^j\|, \tag{5.10}$$

with $c = (1+\Delta t M_1)/(1-\Delta t M_1)$.

The assertion is obviously true for $n = 1$, since $\varepsilon^0 = 0$. Suppose it were true up to n; we would have by (5.9) that

$$\|\varepsilon^{n+1}\| \leqq c\|\varepsilon^n\| + \frac{\Delta t M_1}{1-\Delta t M_1}\|u^{n+1}-\tilde{u}^{n+1}\|,$$

and by substitution of (5.10) that

$$\begin{aligned}\|\varepsilon^{n+1}\| &\leqq \frac{\Delta t M_1}{1-\Delta t M_1}\left\{\sum_{j=1}^{n} c^{n-j+1}\|u^j-\tilde{u}^j\| + \|u^{n+1}-\tilde{u}^{n+1}\|\right\} \\ &= \frac{\Delta t M_1}{1-\Delta t M_1}\sum_{j=1}^{n+1} c^{n-j+1}\|u^j-\tilde{u}^j\|,\end{aligned} \tag{5.11}$$

which proves (5.10).

Now observe, that

$$c^n = \left(\frac{1+\Delta t M_1}{1-\Delta t M_1}\right)^n < \left(\frac{1+\Delta t M_1}{1-\Delta t M_1}\right)^{T/\Delta t},$$

and if $\Delta t < 1/2M_1$, c^n can be bounded by e^{3TM_1}. ($(1+\Delta t M_1)^{T/\Delta t} < e^{TM}$, $(1-\Delta t M_1)^{-T/\Delta t} < e^{2TM_1}$.)

This concludes the proof of Theorem 5.4. To conclude anything from this result, we need an estimate of $\|u^n-\tilde{u}^n\|$. This estimate will be the subject of the next two lemmas. The difference $u^{n+1}-\tilde{u}^{n+1}$ is given by (4.5b) hence it can be expressed in the difference $p^{n+1}-p^n$. If we can prove that this last difference is $O(\Delta t)$ then by (4.5b) $u^n-\tilde{u}^n$ will be $O(\Delta t^2)$ and Theorem 5.4 then gives a global estimate of the error of $O(\Delta t^2)$. Unfortunately it is only possible to estimate the difference $p^{n+1}-p^n$ in terms of $u^n-\tilde{u}^n$, which leads to a circular argument. This complication makes the following two steps necessary.

1. Estimate the quantity

$$LL^T p^n + Lf(\tilde{u}^n) - \dot{g}^n$$

which can be done in terms of $u^j-\tilde{u}^j$, $j<n$ plus a term of $O(\Delta t^2)$.

2. Deduce from an estimate of $u^n - \tilde{u}^n$ in terms of $u^j - \tilde{u}^j$, $j < n$ (obtained from 1) a global estimate of $u^n - \tilde{u}^n$. This final estimate then will yield a global estimate for ε^n. Steps 1 and 2 will be the subject of the next two lemmas.

LEMMA 5.12. *If* $g \in C^4[0, T]$ *then*

$$LL^T p^n = -Lf(\tilde{u}^n) - L\sum_{j=1}^{n-1}(-1)^{n-j-1}(f(u^j) - f(\tilde{u}^j)) + \dot{g}^n + O(\Delta t^2), \qquad n \geqq 1.$$

Proof. We have $LL^T p^0 = -Lf(u^0) + \dot{g}^0$ and

$$\frac{1}{2}LL^T(p^1 + p^0) = -\frac{1}{2}L(f(\tilde{u}^1) + f(u^0)) + \frac{g^1 - g^0}{\Delta t}$$

$$= -\frac{1}{2}L(f(\tilde{u}^1) + f(u^0)) + \frac{1}{2}(\dot{g}^1 + \dot{g}^0) + \frac{\Delta t^2}{24}h^1$$

in which h^1 is a vector $\in R^m$ that is given componentwise by

$$h_j^1 = \frac{d^3}{dt^3}g_j(t_0 + \theta_j^1 \Delta t), \qquad 0 < \theta_j^1 < 1,$$

as is easily verified by Taylor expansion of g^1 and g^0.

We then have

$$LL^T p^1 = -Lf(\tilde{u}^1) + \dot{g}^1 + \frac{\Delta t^2}{12}h^1$$

which proves the formula for $n = 1$.

Suppose the following formula holds up to n

$$LL^T p^n = -Lf(\tilde{u}^n) - L\sum_{j=1}^{n-1}(-1)^{n-j-1}[f(u^j) - f(\tilde{u}^j)] + \dot{g}^n + \frac{\Delta t^2}{12}\sum_{j=1}^{n}(-1)^{n-j}h^j, \tag{5.13}$$

in which the kth component of h^j is given by

$$h_k^j = \frac{d^3}{dt^3}g_k(t_{j-1} + \theta_k^j \Delta t), \qquad 0 < \theta_k^j < 1.$$

We have by Taylor expansion

$$\frac{1}{2}LL^T(p^{n+1} + p^n) = -\frac{1}{2}L(f(\tilde{u}^{n+1}) + f(u^n)) + \frac{g^{n+1} - g^n}{\Delta t}$$

$$= -\frac{1}{2}L(f(\tilde{u}^{n+1}) + f(u^n)) + \frac{1}{2}(\dot{g}^{n+1} + \dot{g}^n) + \frac{\Delta t^2}{24}h^{n+1}. \tag{5.14}$$

Induction hypothesis (5.13) implies

$$\frac{1}{2}LL^T p^{n+1} = -\frac{1}{2}L(f(\tilde{u}^{n+1}) + f(u^n)) + \frac{1}{2}Lf(\tilde{u}^n)$$

$$+\frac{1}{2}L\sum_{j=1}^{n-1}(-1)^{n-j-1}[f(u^j) - f(\tilde{u}^j)] - \frac{1}{2}\dot{g}^n - \frac{1}{2}\frac{\Delta t^2}{12}\sum_{j=1}^{n}(-1)^{n-j}h^j$$

$$+\frac{1}{2}(\dot{g}^{n+1} + \dot{g}^n) + \frac{\Delta t^2}{24}h^{n+1}$$

$$= -\frac{1}{2} Lf(\tilde{u}^{n+1}) - \frac{1}{2} L \sum_{j=1}^{n} (-1)^{n-j} [f(u^j) - f(\tilde{u}^j)]$$

$$+ \frac{1}{2} \dot{g}^{n+1} + \frac{1}{2} \frac{\Delta t^2}{12} \sum_{j=1}^{n+1} (-1)^{n-j+1} h^j.$$

Hence (5.13) holds for all n.

We claim that the term $\sum_{j=1}^{n+1} (-1)^{n-j+1} h^j$ is uniformly bounded as $n \to \infty$ for $n\Delta t < T$. To see this note that two subsequent terms have opposite sign and componentwise typically look like

$$\frac{d^3}{dt^3} g_k(t_{j-1} + \theta_k^j \Delta t) - \frac{d^3}{dt^3} g_k(t_j + \theta_k^{j+1} \Delta t), \qquad 0 < \theta_k^j, \theta_k^{j+1} < 1,$$

which is equal to

$$-\Delta t(1 + \theta_k^{j+1} - \theta_k^j) \frac{d^4}{dt^4} g(\tau),$$

in which τ is some intermediate point.

Since $g \in C^4[0, T]$, g^{iv} is uniformly bounded by a constant G_4, say, and component wise the absolute value of the sum may be overestimated by $(n+1)\ \Delta t\ G_4$, if $n+1$ is even or by $n\ \Delta t\ G_4 + G_3$ if $n+1$ is odd, G_3 being a uniform bound on the third derivative of g. This proves the lemma.

COROLLARY 5.15.

$$\| LL^T p^{n+1} + Lf(\tilde{u}^{n+1}) - \dot{g}^{n+1} \| \leqq nK \sup_{j \leqq n} \| u^j - \tilde{u}^j \| + O(\Delta t^2),$$

$$\| LL^T p^n + Lf(u^n) - \dot{g}^n \| \leqq nK \sup_{j \leqq n} \| u^j - \tilde{u}^j \| + O(\Delta t^2).$$

K is the Lipschitz constant of f.

The first result follows immediately, the second follows from (5.14). The next lemma gives a global estimate for $u^n - \tilde{u}^n$:

LEMMA 5.16. *Let $\| u^0 - \tilde{u}^0 \| = 0$ and let there exist a constant $K_1 \in R^+$ independent of n and Δt such that*

$$\| u^n - \tilde{u}^n \| \leqq K_1 \left\{ (\Delta t)^2 + \Delta t \sum_{j=0}^{n-1} \| u^j - \tilde{u}^j \| \right\}$$

then

$$\| u^n - \tilde{u}^n \| \leqq K_1 (\Delta t)^2 (1 + K_1 \Delta t)^{n-1}.$$

Proof. The result is obviously true for $n = 1$. Suppose it is true up to n. Since

$$\| u^{n+1} - \tilde{u}^{n+1} \| \leqq K_1 \left(\Delta t^2 + \Delta t \sum_{j=1}^{n} \| u^j - \tilde{u}^j \| \right),$$

we have by the induction hypothesis

$$\| u^{n+1} - \tilde{u}^{n+1} \| \leqq K_1 \left\{ (\Delta t)^2 + \Delta t \sum_{j=1}^{n} K_1 (\Delta t)^2 (1 + K_1 \Delta t)^{j-1} \right\}$$

$$= K_1 (\Delta t)^2 \left(1 + K_1 \Delta t \sum_{j=1}^{n} (1 + K_1 \Delta t)^{j-1} \right)$$

$$= K_1(\Delta t)^2\left(1+K_1\Delta t\frac{(1+K_1\Delta t)^n-1}{1+K_1\Delta t-1}\right)$$

$$= K_1(\Delta t)^2(1+K_1\Delta t)^n,$$

proving the lemma.

These results lead to the following global error estimate:

THEOREM 5.17. *Let hypotheses* (5.1a) *and* (5.1b) *be satisfied and let* $g \in C^4[0, T]$. *Then the difference of the pressure correction solution* u^n *of* (5.2) *and the Crank-Nicolson solution* v^n *of* (5.5) *is* $O(\Delta t^2)$ *globally, whereas* p^n *and* q^n *differ by* $O(\Delta t)$ *globally.*

Proof. If we can show the existence of a constant K_2 independent of n and Δt such that $\|u^n-\tilde{u}^n\| \leqq K_2(\Delta t)^2$ for $n\Delta t \leqq T$ the theorem would follow from Theorem 5.4 and Corollary 5.15. By (4.5b) we have

$$\|u^n-\tilde{u}^n\| \leqq \frac{\Delta t}{2}\|L^T\|\,\|p^n-p^{n-1}\|.$$

Furthermore, by (5.13)

$$\|p^n-p^{n-1}\| \leqq \|(LL^T)^{-1}\|\,\|L\|\,\|f(\tilde{u}^n)-f(u^{n-1})$$

$$+2\sum_{j=1}^{n-1}(-1)^{n-j-1}[f(u^j)-f(\tilde{u}^j)]\| + \|\dot{g}^n-\dot{g}^{n-1}\| + O(\Delta t^2). \tag{5.18}$$

We also have using (4.5a) and the uniform boundedness of u^n that

$$\|f(\tilde{u}^n)-f(u^{n-1})\| \leqq K\|\tilde{u}^n-u^{n-1}\| \leqq K_3\Delta t$$

uniformly for $n\Delta t \leqq T$. (K is the Lipschitz constant, K_3 is a constant involving K, M and a constant bounding p^n+p^{n-1}, which follows essentially from (5.14)). Furthermore, $\|\dot{g}^n-\dot{g}^{n-1}\| \leqq K_4\Delta t$ by the uniform boundedness of d^2g/dt^2 and finally

$$\left\|2\sum_{j=1}^{n-1}(-1)^{n-j-1}[f(u^j)-f(\tilde{u}^j)]\right\| \leqq 2K\sum_{j=1}^{n-1}\|u^j-\tilde{u}^j\|$$

uniformly. Putting all this into (5.18) we have the existence of K_5 such that

$$\|u^n-\tilde{u}^n\| \leqq K_5\left[(\Delta t)^2+\Delta t\sum_{j=1}^{n-1}\|u^j-\tilde{u}^j\|\right].$$

By Lemma 5.16 this implies the inequality

$$\|u^n-\tilde{u}^n\| \leqq K_5(\Delta t)^2(1+K_5\Delta t)^{n-1},$$

and since $n\Delta t \leqq T$ we have

$$(1+K_5\Delta t)^{n-1} \leqq (1+K_5\Delta t)^{T/\Delta t} < e^{K_5T}$$

and this gives us

$$\|u^n-\tilde{u}^n\| \leqq K_2(\Delta t)^2 \quad \text{for all } n \text{ such that } n\Delta t \leqq T$$

($K_2 = K_5\, e^{K_5T}$). This proves the theorem.

Remark 5.19. By using somewhat more smoothness for f and using the fact that in expression (5.13) the signs alternate in the sum it is possible to sharpen the results in such a way that p^n also is $O(\Delta t^2)$ accurate.

Remark 5.20. When the solution v^n of (5.5) by (4.4) is an $O(\Delta t^2)$ approximation of the exact solution, so is the PC solution. Anyway the PC algorithm does not spoil the accuracy of the CN scheme.

Some predictor corrector schemes spoil stability and now we turn our attention to that question.

6. The effect of pressure correction on linear stability. In order to study the effects of pressure correction on linear stability we consider the linearized problem

$$\begin{aligned} &\dot{u} = Au + L^T p, \\ &Lu = 0, \\ &u \in R^k, \quad A \in \mathscr{L}(R^k, R^k), \quad p \in R^m, \quad L \in \mathscr{L}(R^k, R^m), \quad m < k. \end{aligned} \tag{6.1}$$

We shall make the assumption that A is dissipative, that is

$$(u, Au) \leqq 0 \quad \forall u \in R^k. \tag{6.2}$$

System (6.1) describes for example the error propagation in the so-called Oseen approximation with periodic boundary conditions and a space periodic convective velocity field. For this test problem we have the following theorem:

THEOREM 6.3. *Problem* (6.1) *with Assumption* (6.2) *integrated with the pressure correction method is unconditionally stable in the following sense*

$$\|u^{n+1}\|^2 + \tfrac{1}{4}\Delta t^2 \|L^T p^{n+1}\|^2 \leqq \|u^n\|^2 + \tfrac{1}{4}\Delta t^2 \|L^T p^n\|^2$$

independent of Δt.

Proof. Applying (4.5) to (6.1) we get

$$\frac{\tilde{u}^{n+1} - u^n}{\Delta t} = \frac{1}{2}[A\tilde{u}^{n+1} + Au^n] + L^T p^n, \tag{6.4a}$$

$$\frac{u^{n+1} - \tilde{u}^{n+1}}{\Delta t} = \frac{1}{2} L^T (p^{n+1} - p^n), \tag{6.4b}$$

and since $Lu^{n+1} = 0$

$$-\frac{L\tilde{u}^{n+1}}{\Delta t} = \frac{1}{2} LL^T (p^{n+1} - p^n). \tag{6.5}$$

We take the inner product of (6.4a) and $\tilde{u}^{n+1} + u^n$ and obtain

$$\|\tilde{u}^{n+1}\|^2 = \|u^n\|^2 + \tfrac{1}{2}\Delta t(\tilde{u}^{n+1} + u^n, A(\tilde{u}^{n+1} + u^n)) + \Delta t(\tilde{u}^{n+1} + u^n, L^T p^n).$$

Since $(x, By) = (B^T x, y)$ and $Lu^n = 0$, this yields by hypothesis (6.2),

$$\|\tilde{u}^{n+1}\|^2 \leqq \|u^n\|^2 + \Delta t(L\tilde{u}^{n+1}, p^n), \tag{6.6}$$

and by (6.5)

$$\|\tilde{u}^{n+1}\|^2 \leqq \|u^n\|^2 - \tfrac{1}{2}\Delta t^2 (L^T (p^{n+1} - p^n), L^T p^n).$$

The inner product of (6.4b) with $u^{n+1} + \tilde{u}^{n+1}$ gives

$$\|u^{n+1}\|^2 \leqq \|\tilde{u}^{n+1}\|^2 + \tfrac{1}{2}\Delta t(u^{n+1} + \tilde{u}^{n+1}, L^T (p^{n+1} - p^n)) \tag{6.7}$$

and since $Lu^{n+1} = 0$ this reduces to

$$\begin{aligned} \|u^{n+1}\|^2 &\leqq \|\tilde{u}^{n+1}\|^2 + \tfrac{1}{2}\Delta t(L\tilde{u}^{n+1}, p^{n+1} - p^n) \\ &= \|\tilde{u}^{n+1}\|^2 - \tfrac{1}{4}\Delta t^2 (L^T (p^{n+1} - p^n), L^T (p^{n+1} - p^n)). \end{aligned}$$

Substitution of this result into (6.6) yields

$$\|u^{n+1}\|^2 \leqq \|u^n\|^2 + \tfrac{1}{4}\Delta t^2 (L^T p^n, L^T p^n) - \tfrac{1}{4}\Delta t^2 (L^T p^{n+1}, L^T p^{n+1}),$$

whence

$$\|u^{n+1}\|^2+\tfrac{1}{4}\Delta t^2\|L^T p^{n+1}\|^2 \leqq \|u^n\|^2+\tfrac{1}{4}\Delta t^2\|L^T p^n\|^2. \tag{6.8}$$

This proves the theorem.

Remark. As can be seen the result of Theorem 6.3 implies Hypothesis (5.1a). So for the linear case we have shown that the Lax theorem ("stability implies convergence") also holds for PC methods. This is not entirely trivial: it needs introduction of the specific Lyapunov functional $\|u\|+\Delta t^2/4\|L^T p\|$.

7. Towards a second order pressure correction scheme for the Navier–Stokes equations. The first step towards a second order PC scheme for the Navier-Stokes equation is to graft formulae (4.5a)-(4.5c) onto scheme (2.3) to obtain ($\alpha=(i,j)$)

$$\text{a1}\quad \frac{\tilde{u}^{n+1}_{1,\alpha}-u^n_{1,\alpha}}{\Delta t}+\frac{1}{2}[\tilde{u}^{n+1}_{1,\alpha}D^1_x\tilde{u}^{n+1}_{1,\alpha}+\tilde{u}^{n+1}_{2,\alpha}D^1_y\tilde{u}^{n+1}_{1,\alpha}+u^n_{1,\alpha}D^1_x u^n_{1,\alpha}+\bar{u}^n_{2,\alpha}D^1_y u^n_{1,\alpha}]+\nabla_x p^n_\alpha$$

$$=\frac{1}{2}\mathrm{Re}^{-1}[D^2_x(\tilde{u}^{n+1}_{1,\alpha}+u^n_{1,\alpha})+D^2_y(\tilde{u}^{n+1}_{1,\alpha}+u^n_{1,\alpha})],$$

$$\text{a2}\quad \frac{\tilde{u}^{n+1}_{2,\alpha}-u^n_{2,\alpha}}{\Delta t}+\frac{1}{2}[\tilde{u}^{n+1}_{1,\alpha}D^1_x\tilde{u}^{n+1}_{2,\alpha}+\tilde{u}^{n+1}_{2,\alpha}D^1_y\tilde{u}^{n+1}_{2,\alpha}+\bar{u}^n_{1,\alpha}D^1_x u^n_{2,\alpha}+u^n_{2,\alpha}D^1_y u^n_{2,\alpha}]+\nabla_y p^n_\alpha$$

$$=\frac{1}{2}\mathrm{Re}^{-1}[D^2_x(\tilde{u}^{n+1}_{2,\alpha}+u^n_{2,\alpha})+D^2_y(\tilde{u}^{n+1}_{2,\alpha}+u^n_{2,\alpha})], \tag{7.1}$$

$$\text{b1}\quad \frac{u^{n+1}_{1,\alpha}-\tilde{u}^{n+1}_{1,\alpha}}{\Delta t}=\frac{1}{2}\nabla_x(p^{n+1}_\alpha-p^n_\alpha),$$

$$\text{b2}\quad \frac{u^{n+1}_{2,\alpha}-\tilde{u}^{n+1}_{2,\alpha}}{\Delta t}=\frac{1}{2}\nabla_y(p^{n+1}_\alpha-p^n_\alpha),$$

$$\text{c}\quad \Delta_x u^{n+1}_{1,\alpha}+\Delta_y u^{n+1}_{2,\alpha}=0.$$

In order to obtain an efficient method two modifications are applied to (7.1). The first modification in this scheme is to linearize the equations by approximating a term of the form $\int_{t_n}^{t_n+\Delta t}\phi\psi\,dt$ by $\frac{1}{2}\Delta t[\phi_n\psi_{n+1}+\psi_n\phi_{n+1}]+O(\Delta t^3)$ instead of $\frac{1}{2}\Delta t[\phi_{n+1}\psi_{n+1}+\psi_n\phi_n]+O(\Delta t^3)$, as in (7.1).

This enables us to calculate $\tilde{u}^{n+1}_1$ and $\tilde{u}^{n+1}_2$ solving only *linear* systems of equations. So we can calculate p^{n+1} and u^{n+1} by the procedure explained in § 4. Note that the set of equations c contains the boundary conditions for u_1 and u_2 and therefore in matrix form is something like

$$L_x u_1^{n+1}+L_y u_2^{n+1}=g^{n+1}.$$

Also note that substitution of c into b results in a Poisson equation for the pressure that already includes its boundary conditions. There is no checkerboarding thanks to the staggered grid. A word of warning is in order: the literature abounds with methods that solve this Poisson equation approximately ([2], [5] and [12]). This is good enough for an accuracy of $O(\Delta t)$ in the velocities, but for an accuracy of $O(\Delta t^2)$ one should solve this Poisson equation almost to machine accuracy by a fast Poisson solver like preconditioned conjugate gradients (PCG), multigrid (MGD) or discrete Fourier type methods (FTM). In our experiments we used FTM and MGD.

Scheme (7.1) even with the modification just introduced cannot be solved efficiently with a large number of nodes. We therefore introduce as a second modification an

ADI formulation using only tridiagonal systems. Let

$$A^n_{1x} = u^n_{1,\alpha} D^1_x - \mathrm{Re}^{-1} D^2_x, \qquad A^n_{1y} = \bar{u}^n_{2,\alpha} D^1_y - \mathrm{Re}^{-1} D^2_y,$$
$$A^n_{2x} = \bar{u}^n_{1,\alpha} D^1_x - \mathrm{Re}^{-1} D^2_x, \qquad A^n_{2y} = u^n_{2,\alpha} D^1_y - \mathrm{Re}^{-1} D^2_y,$$

and define A^{n+1} and A^* analogously.

We now give an ADI formulation of the modified version of (7.1)a.

Modified Crank-Nicolson:

$$\text{a1}\quad \frac{\tilde{u}^{n+1}_{1,\alpha} - u^n_{1,\alpha}}{\Delta t} + \frac{1}{2}[\tilde{A}^{n+1}_{1x} u^n_{1,\alpha} + A^n_{1x} \tilde{u}^{n+1}_{1,\alpha} + \tilde{A}^{n+1}_{1y} u^n_{1,\alpha} + A^n_{1y} \tilde{u}^{n+1}_{1,\alpha}] + \nabla_x p^n_\alpha = 0,$$
$$\text{a2}\quad \frac{\tilde{u}^{n+1}_{2,\alpha} - u^n_{2,\alpha}}{\Delta t} + \frac{1}{2}[\tilde{A}^{n+1}_{2x} u^n_{2,\alpha} + A^n_{2x} \tilde{u}^{n+1}_{2,\alpha} + \tilde{A}^{n+1}_{2y} u^n_{2,\alpha} + A^n_{2y} \tilde{u}^{n+1}_{2,\alpha}] + \nabla_y p^n_\alpha = 0. \tag{7.2}$$

The ADI version of this scheme is

$$\text{a11}\quad u^*_{1,\alpha} + \frac{\Delta t}{2}[A^n_{1x} u^*_{1,\alpha}] = u^n_{1,\alpha} - \frac{\Delta t}{2}[\tilde{A}^{n+1}_{1y} u^n_{1,\alpha} + \nabla_x p^n_\alpha],$$
$$\text{a12}\quad \tilde{u}^{n+1}_{1,\alpha} + \frac{\Delta t}{2}[A^n_{1y} \tilde{u}^{n+1}_{1,\alpha}] = u^*_{1,\alpha} - \frac{\Delta t}{2}[\tilde{A}^{n+1}_{1x} u^*_{1,\alpha} + \nabla_x p^n_\alpha],$$
$$\text{a21}\quad u^*_{2,\alpha} + \frac{\Delta t}{2}[A^n_{2x} u^*_{2,\alpha}] = u^n_{2,\alpha} - \frac{\Delta t}{2}[\tilde{A}^{n+1}_{2y} u^n_{2,\alpha} + \nabla_y p^n_\alpha], \tag{7.3}$$
$$\text{a22}\quad \tilde{u}^{n+1}_{2,\alpha} + \frac{\Delta t}{2}[A^n_{2y} \tilde{u}^{n+1}_{2,\alpha}] = u^*_{2,\alpha} - \frac{\Delta t}{2}[\tilde{A}^{n+1}_{2x} u^*_{2,\alpha} + \nabla_y p^n_\alpha].$$

In every step we only have to solve tridiagonal linear systems. However the operators $\tilde{A}^{n+1}_{1,y}$, $\tilde{A}^{n+1}_{1,x}$, $\tilde{A}^{n+1}_{2,y}$ and $\tilde{A}^{n+1}_{2,x}$ need the values $\tilde{u}^{n+1}_{1,\alpha}$ and $\tilde{u}^{n+1}_{2,\alpha}$ on moments that they are still not known. We shall return to this problem in a few moments. First we show that (7.3) has also $O(\Delta t^2)$ consistency if the solutions are sufficiently smooth.

We multiply a11 by $(1-(\Delta t/2)\tilde{A}^{n+1}_{1x})$ and a12 by $(1+(\Delta t/2)A^n_{1x})$ and get, using the fact that $\tilde{A}^{n+1} = A^n + O(\Delta t)$

$$\left(1 + \frac{\Delta t}{2}[A^n_{1x} + A^n_{1y}]\right) \tilde{u}^{n+1}_{1,\alpha} + \frac{\Delta t^2}{4} A^n_{1x} A^n_{1y} \tilde{u}^{n+1}_{1,\alpha}$$
$$= \left(1 - \frac{\Delta t}{2}[\tilde{A}^{n+1}_{1x} + \tilde{A}^{n+1}_{1y}]\right) u^n_{1,\alpha} + \frac{\Delta t^2}{4} \tilde{A}^{n+1}_{1x} \tilde{A}^{n+1}_{1y} u^n_{1,\alpha} - \Delta t \nabla_x p^n_\alpha + O(\Delta t^3), \tag{7.4}$$

and since $\tilde{u}_{1,\alpha} - u^n_{1,\alpha} = O(\Delta t)$,

$$\left(1 + \frac{\Delta t}{2}[A^n_{1x} + A^n_{1y}]\right) \tilde{u}^{n+1}_{1,\alpha} = \left(1 - \frac{\Delta t}{2}[\tilde{A}^{n+1}_{1x} + \tilde{A}^{n+1}_{1y}]\right) u^n_{1,\alpha} - \Delta t \nabla_x p^n_\alpha + O(\Delta t^3).$$

This proves our consistency claim.

With respect to the boundary conditions for $u^*_{1,\alpha}$ and $u^*_{2,\alpha}$ in scheme (7.3) it will be clear, that if the boundary conditions are time-independent one can take u as boundary condition for u^* (or $\partial u/\partial n$ for $\partial u^*/\partial n$). When the boundary conditions are time-dependent we have from (7.3) following, an idea of Fairweather and Mitchell [4],

$$2u^*_{1,\alpha} = \tilde{u}^{n+1}_{1,\alpha} + u^n_{1,\alpha} - \frac{\Delta t}{2}[\tilde{A}^{n+1}_{1y} u^n_{1,\alpha} - A^n_{1y} \tilde{u}^{n+1}_{1,\alpha} + \tilde{A}^{n+1}_{1x} u^*_{1,\alpha} - A^n_{1x} u^*_{1,\alpha}]. \tag{7.5}$$

The simpler boundary condition

$$u^*_{1,\alpha} = \frac{u^{n+1}_{1,\alpha} + u^n_{1,\alpha}}{2} \tag{7.6}$$

would result in an extra error of $O(\Delta t^2/\Delta x^2)$ on back substitution into 7.3. This effect can be noticed when Δx tends to zero while Δt remains constant. In order to make (7.3) workable we have to have an approximation to $\tilde{u}^{n+1}_{1,\alpha}$ and $u^{n+1}_{2,\alpha}$ when they are needed in the right-hand sides of equations a11–a22. To obtain such an approximation we first use a prediction step with the old values u^n_1 and u^n_2, obtain values u^p_1 and u^p_2, presumably off by $O(\Delta t^2)$ locally and using those in the correction step to obtain a final value $\tilde{u}^{n+1}_1$ and $\tilde{u}^{n+1}_2$.

Summing it all up we have:

Prediction step:

$$\text{a11p} \quad u^{p*}_{1,\alpha} + \frac{\Delta t}{2}[A^n_{1x} u^{p*}_{1,\alpha}] = u^n_{1,\alpha} - \frac{\Delta t}{2}[A^n_{1y} u^n_{1,\alpha} + \nabla_x p^n_\alpha],$$

$$\text{a12p} \quad u^p_{1,\alpha} + \frac{\Delta t}{2}[A^n_{1y} u^p_{1,\alpha}] = u^{p*}_{1,\alpha} - \frac{\Delta t}{2}[A^n_{1x} u^{p*}_{1,\alpha} + \nabla_x p^n_\alpha],$$

$$\text{a21p} \quad u^{p*}_{2,\alpha} + \frac{\Delta t}{2}[A^n_{2x} u^{p*}_{2,\alpha}] = u^n_{2,\alpha} - \frac{\Delta t}{2}[A^n_{2y} u^n_{2,\alpha} + \nabla_y p^n_\alpha],$$

$$\text{a22p} \quad u^p_{2,\alpha} + \frac{\Delta t}{2}[A^n_{2y} u^p_{2,\alpha}] = u^{p*}_{2,\alpha} - \frac{\Delta t}{2}[A^n_{2x} u^{p*}_{2,\alpha} + \nabla_y p^n_\alpha].$$

Correction step:

$$\text{a11c} \quad u^*_{1,\alpha} + \frac{\Delta t}{2}[A^n_{1x} u^*_{1,\alpha}] = u^n_{1,\alpha} - \frac{\Delta t}{2}[A^p_{1y} u^n_{1,\alpha} + \nabla_x p^n_\alpha],$$

$$\text{a12c} \quad \tilde{u}^{n+1}_{1,\alpha} + \frac{\Delta t}{2}[A^n_{1y} \tilde{u}^{n+1}_{1,\alpha}] = u^*_{1,\alpha} - \frac{\Delta t}{2}[A^p_{1x} u^*_{1,\alpha} + \nabla_x p^n_\alpha],$$

$$\text{a21c} \quad u^*_{2,\alpha} + \frac{\Delta t}{2}[A^n_{2x} u^*_{2,\alpha}] = u^n_{2,\alpha} - \frac{\Delta t}{2}[A^p_{2y} u^n_{2,\alpha} + \nabla_y p^n_\alpha], \tag{7.7}$$

$$\text{a22c} \quad \tilde{u}^{n+1}_{2,\alpha} + \frac{\Delta t}{2}[A^n_{2y} \tilde{u}^{n+1}_{2,\alpha}] = u^*_{2,\alpha} - \frac{\Delta t}{2}[A^p_{2x} u^*_{2,\alpha} + \nabla_y p^n_\alpha].$$

Pressure correction step:

$$\text{b1} \quad \frac{u^{n+1}_{1,\alpha} - \tilde{u}^{n+1}_{1,\alpha}}{\Delta t} = \frac{1}{2}\nabla_x(p^{n+1}_\alpha - p^n_\alpha),$$

$$\text{b2} \quad \frac{u^{n+1}_{2,\alpha} - \tilde{u}^{n+1}_{2,\alpha}}{\Delta t} = \frac{1}{2}\nabla_y(p^{n+1}_\alpha - p^n_\alpha).$$

Divergence freedom:

$$\text{c} \quad \Delta_x u^{n+1}_{1,\alpha} + \Delta_y u^{n+1}_{2,\alpha} = 0.$$

In the now familiar way c is substituted into b1 and b2 giving a discrete Poisson equation for the pressure correction. The solution of this Poisson equation is back substituted into b1 and b2 and this ends one time cycle.

One may ask the question whether we should not make (u_1^p, u_2^p) divergence free. It appears, however, that the correction for incompressibility is only $O(\Delta t^2)$ and (u_1^p, u_2^p) is no more accurate than that anyway.

Again we note that the boundary values for the discrete Poisson equation are already incorporated in the equations. This will be apparent when they are recast in vector form.

A significant improvement in terms of efficiency of scheme (7.7) is possible, and we actually put it to the test. For a comparison of the results see § 8. A different approximation of the convection terms is taken, and by use of the order in which the variables are calculated the predictor step is eliminated as follows:

$$\text{a11}\quad u^*_{1,\alpha}+\frac{\Delta t}{2}[A^n_{1x}u^*_{1,\alpha}]=u^n_{1,\alpha}-\frac{\Delta t}{2}[A^n_{1y}u^n_{1,\alpha}+\nabla_x p^n_\alpha],$$

$$\text{a21}\quad u^*_{2,\alpha}+\frac{\Delta t}{2}[A^*_{2x}u^*_{2,\alpha}+A^*_{2y}u^n_{2,\alpha}]=u^n_{2,\alpha}-\frac{\Delta t}{2}\nabla_y p^n_\alpha,$$

(7.8)

$$\text{a22}\quad \tilde{u}^{n+1}_{2,\alpha}+\frac{\Delta t}{2}[A^*_{2y}\tilde{u}^{n+1}_{2,\alpha}]=u^*_{2,\alpha}-\frac{\Delta t}{2}[A^*_{2x}u^*_{2,\alpha}+\nabla_y p^n_\alpha],$$

$$\text{a12}\quad \tilde{u}^{n+1}_{1,\alpha}+\frac{\Delta t}{2}[\tilde{A}^{n+1}_{1x}u^*_{1,\alpha}+\tilde{A}^{n+1}_{1y}\tilde{u}^{n+1}_{1,\alpha}]=u^*_{1,\alpha}-\frac{\Delta t}{2}\nabla_x p^n_\alpha,$$

followed by the pressure correction steps b1, b2 and c as in scheme (7.7).

Note that A^*_{2x} contains $u^*_{1,\alpha}$ which is known once (7.8)-a11 has been calculated and one may easily check, that (7.8)-a11, (7.8)-a21, (7.8)-a22, (7.8)-a12 in that order are sets of linear equations.

By elimination of $u^*_{1,\alpha}$ and $u^*_{2,\alpha}$ from (7.8) one may verify that (7.8) is an $O(\Delta t^3)$ approximation of the following equations:

$$\left(1+\frac{\Delta t}{2}A^n_{1x}+\frac{\Delta t}{2}\tilde{A}^{n+1}_{1y}\right)\tilde{u}^{n+1}_{1,\alpha}=\left(1-\frac{\Delta t}{2}\tilde{A}^{n+1}_{1x}-\frac{\Delta t}{2}A^n_{1y}\right)u^n_{1,\alpha}-\Delta t\nabla_x p^n_\alpha,$$

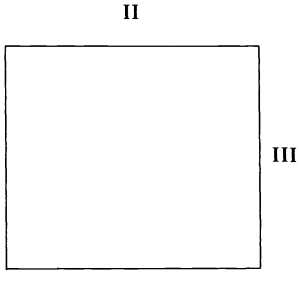
$$\left(1+\frac{\Delta t}{2}A^*_{2x}+\frac{\Delta t}{2}A^*_{2y}\right)\tilde{u}^{n+1}_{2,\alpha}=\left(1-\frac{\Delta t}{2}A^*_{2x}-\frac{\Delta t}{2}A^*_{2y}\right)u^n_{2,\alpha}-\Delta t\nabla_y p^n_\alpha,$$

which in itself is a local $O(\Delta t^3)$ approximation of (7.1).

8. Experimental verification of second order accuracy in time. The second order accuracy in time of schemes (7.7) and (7.8) was experimentally verified on the following test problem:

Navier-Stokes equation, $\mathrm{Re}=10$ on the square $(0,1)\times(0,1)$ with a 5×5 grid in space. The Poisson equation for the pressure was solved by a Fourier method for scheme (7.7) and by the multigrid program mgd1 of Wesseling [14] for scheme (7.8).

(8.1)



The boundary conditions were taken as follows:

(a) I and IV: fixed no slip walls: $u_1 = u_2 = 0$.

(b) II prescribed inlet: $u = 0$, $v = -\sin(\pi(x^3 - 3x^2 + 3x))e^{1-1/t}$

(c) III outlet. We experimented with prescribed boundary conditions and "natural" boundary conditions

$$\nu \frac{\partial u_1}{\partial x} - p = 0, \qquad \frac{\partial u_2}{\partial x} = 0.$$

We will deal with the physical significance of these boundary conditions later on. For the convergence it made little difference which boundary condition we took. (It may, however, make a difference for large Reynolds numbers.)

Initial value:

$$u_1 = u_2 = 0. \tag{8.2}$$

We integrated this equation with time steps 1/8, 1/16, 1/32 and 1/64 from $t = 0$ to $t = 1$. In scheme (7.8) the number of multigrid iterations per time step was 3.

Denoting the quantity

$$\frac{\|u(\Delta t) - u(\frac{1}{2}\Delta t)\|}{\|u(\frac{1}{2}\Delta t) - u(\frac{1}{4}\Delta t)\|}$$

by $K(u, \Delta t)$ we obtained the following results: ($\|x\| = (\sum x_i^2)^{1/2}$) (see Table 1).

TABLE 1

	Scheme (5.7)	Scheme (5.8)
$K(u_1, 1/8)$	4.1	4.0
$K(u_1, 1/16)$	4.0	3.9
$K(u_2, 1/8)$	3.9	4.0
$K(u_2, 1/16)$	3.8	3.7

This is in complete agreement with the theoretical results. For the pressure the results appeared to be less trustworthy. However, once the velocities are known with prescribed accuracy it is a simple matter to calculate the pressure with the same accuracy by solving

$$LL^T p^n = -Lf(u^n) + \dot{g}^n.$$

9. Numerical simulation of a flow problem. We simulated a flow problem (somewhat related to the flow in a glass furnace) with schemes (7.7) and (7.8) for various Reynolds numbers. It was found that the results of (7.7) and (7.8) are very similar so that it is hardly necessary to calculate both. It is to be noted that (7.8) uses half the computer time of (7.7).

The configuration is shown in Fig. 9.1. The boundary conditions at the free slip wall are

$$\frac{\partial u_1}{\partial y} = 0, \qquad u_2 = 0.$$

At the no slip walls both velocity components are zero. In the first experiment we calculated the stationary solution using (7.7) as a relaxation method. At the inlet we prescribed the velocities

$$u_1 = 0, \qquad u_2 = -432(x - 0.25)^2 x$$

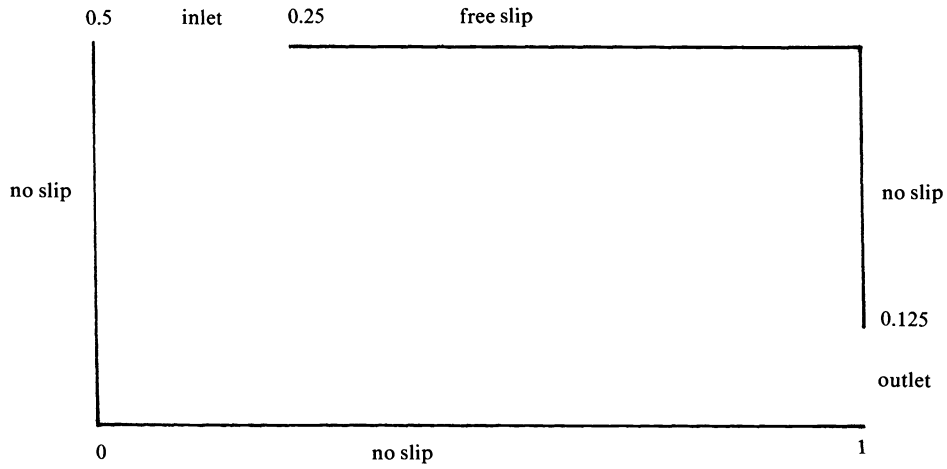


FIG. 9.1

and at the outlet we also prescribed the velocities

$$u_2 = 0, \qquad u_1 = 432(0.125 - y)y. \tag{9.1}$$

As may be easily checked these values satisfy the compatibility condition

$$\int_{\partial\Omega} \mathbf{u} \cdot \mathbf{n}\, d\,\partial\Omega = 0.$$

Calculations were performed on a 12×24 grid with time step $\Delta t = 0.1$ until convergence was obtained. Results are plotted for Re-numbers 100, 500 and 1500. See Figs. 9.2-9.4[1]. As might be expected the numerical boundary layer at the outlet caused wiggles in the stationary solution. These wiggles disappeared when the grid was locally refined in such a way that the mesh Péclet number (u_1 Re Δx) was so small that the numerical boundary layer at the outlet could be represented by the grid. The results of this can be seen in Figs. 9.5-9.6. The observed behaviour of numerical wiggles is completely analogous to and explained by what has been found for a simple convection-diffusion equation by Segal [10]. Note that wiggles are absent, although central

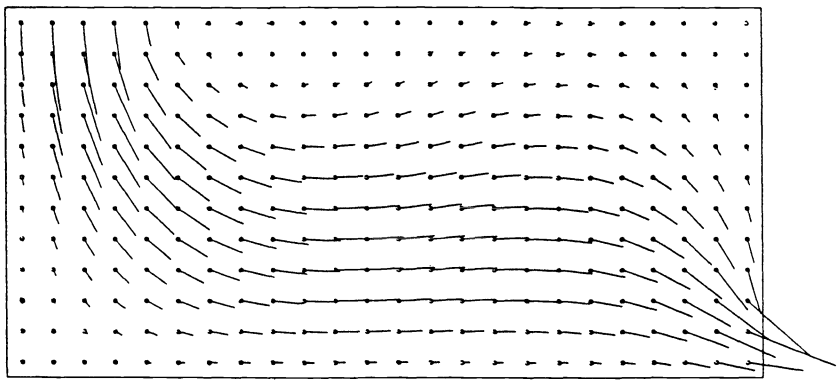


FIG. 9.2. Re = 100, $u_{max} = 1.4$, *equidistant grid, scheme* (7.7).

[1] In all vector plots the velocity is scaled to the maximum occurring velocity. The magnitude of this velocity is given as u_{max}.

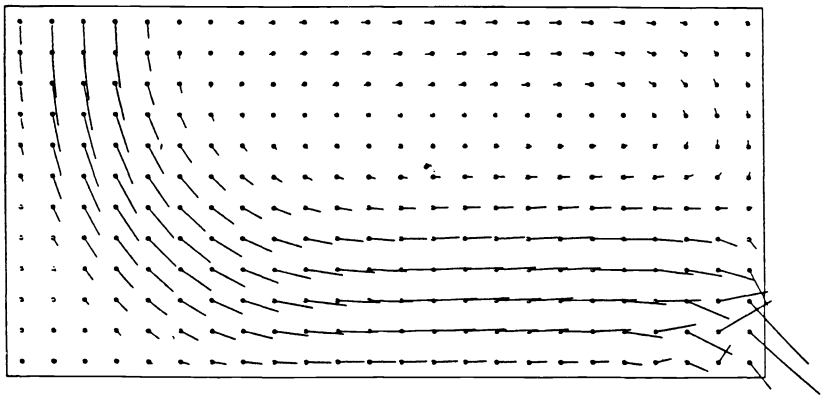


FIG. 9.3. Re = 500, $u_{max} = 1.8$, *equidistant grid, scheme* (7.7).

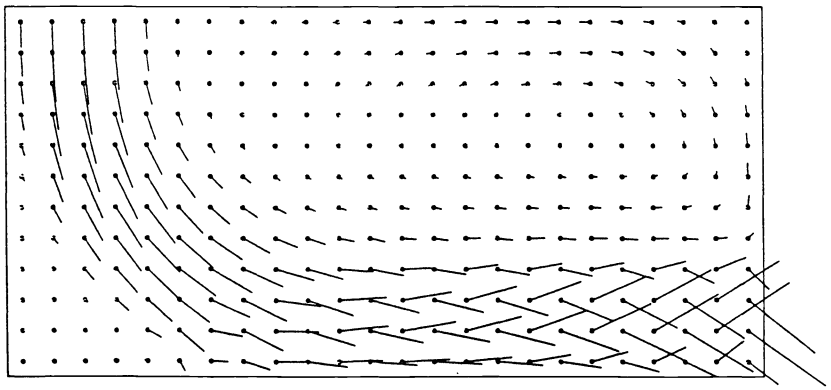


FIG. 9.4. Re = 1500, $u_{max} = 1.7$, *equidistant grid, scheme* (7.7).

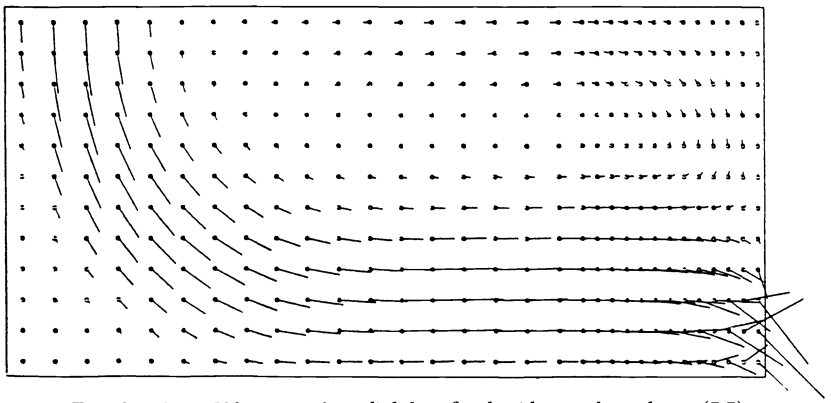


FIG. 9.5. Re = 500, $u_{max} = 2.1$, *slightly refined grid at outlet, scheme* (7.7).

differences are used and the mesh Péclet number is much larger than two in a large part of the flow. This is in accordance with the study of Gresho and Lee [6]. It was considered not very satisfactory to use a large number of gridpoints to represent a boundary layer that has no physical significance and we experimented with various other boundary conditions at the outlet. As Gresho and Lee [6] remark the physical natural boundary condition at the outlet is: no viscous normal and tangential stresses,

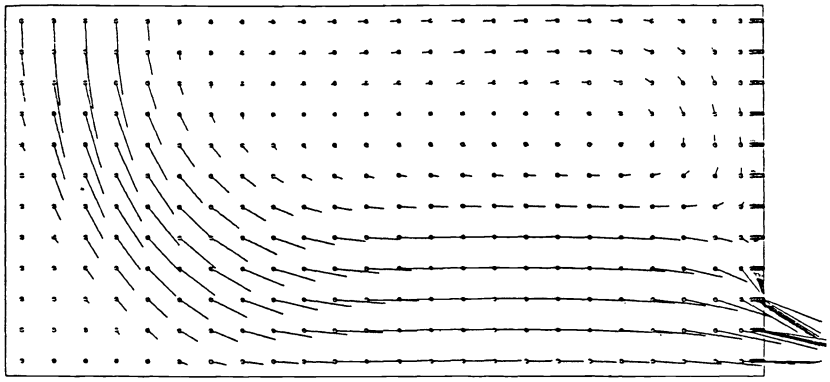


FIG. 9.6. Re = 500, $u_{max} = 1.6$, *very refined grid at outlet, (scheme* (7.7)).

in other words,

$$-p + 2\nu \frac{\partial u_1}{\partial x} = 0, \qquad \frac{\partial u_2}{\partial y} = 0. \tag{9.2}$$

These boundary conditions do not fit in easily in the finite difference scheme and we modified (9.2) a little to obtain

$$-p + \nu \frac{\partial u_1}{\partial x} = 0, \qquad \frac{\partial u_2}{\partial y} = 0, \tag{9.3}$$

which can be considered to be a sort of "natural" boundary condition for the second order part of the Navier-Stokes equation. This procedure did not totally eliminate the wiggles, but as can be seen from Figs. 9.7 and 9.8 it takes considerably less grid refinement to eliminate them. This is in accordance with the findings of Gresho and Lee [6] and Segal [10].

The alternative approach to eliminate the wiggles which is advocated in much of the literature (for example, [11, Chap. 5.2.2]), namely to replace the central differences in the convective terms by so-called "upwind" differences, is one which we did not want to adopt. There is an increasing amount of evidence that elimination of the wiggles by this method also corrupts the solution by introducing a very big cross-stream viscosity. See [7], [9], [10] and [13].

For Re = 1500 we could no longer obtain convergence with $\Delta t = 0.1$ both for scheme (7.7) and scheme (7.8) and this is probably due to nonlinear instability that

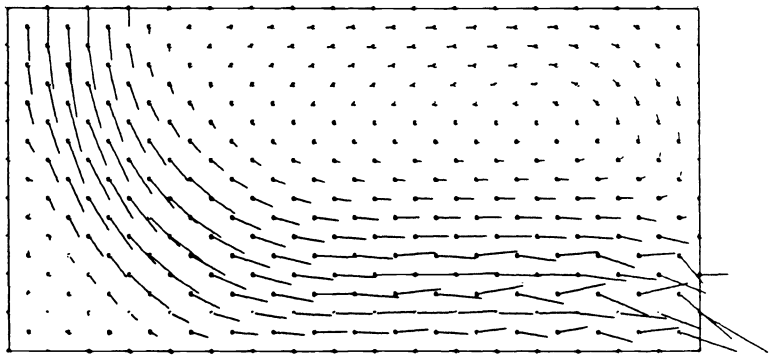


FIG. 9.7. Re = 500, $u_{max} = 1.3$, *equidistant grid, scheme* (7.8) *with boundary conditions* (9.8).

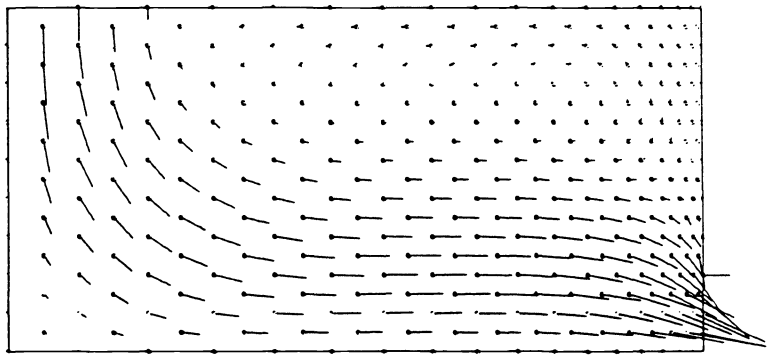


FIG. 9.8. $\mathrm{Re} = 500$, $u_{\max} = 2.1$, *slightly refined grid at outlet, scheme* (7.8) *with boundary conditions* (9.8).

plagues Crank-Nicolson methods in the high frequency domain. We also calculated with scheme (7.8) a real time-dependent problem. The prescribed velocity at the inlet was multiplied by a factor $(1-\cos 2\pi t)/2$, and the outflow condition was given by (9.3). All other data remained the same. The problem parameters were set to: $\Delta t = 1/16$, $\mathrm{Re} = 100$ and we integrated from $t = 0$ to $t = 4$.

For the time cycle starting at $t = 3$ and ending at $t = 4$ see Figs. 9.9-9.13. Remark that the velocity profiles at $t = 3$ and $t = 4$ are identical. (In fact they differ by 10^{-3}.)

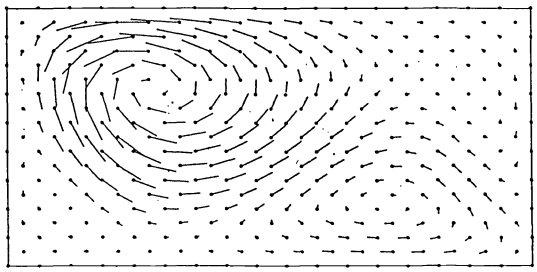


FIG. 9.9. $\mathrm{Re} = 100$, $u_{\max} = 0.19$, $t = 3.0$.

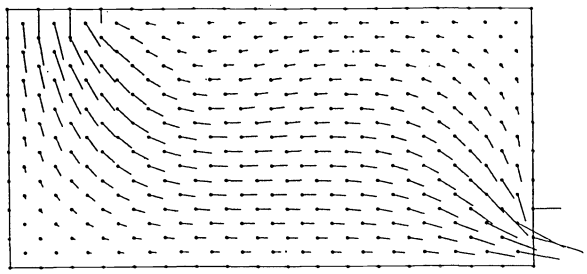


FIG. 9.10. $\mathrm{Re} = 100$, $u_{\max} = 0.62$, $t = 3.25$.

10. Concluding remarks. The practical computations of §§ 8 and 9 indicate that schemes (7.7) and (7.8) are very useful to calculate the transient solutions of the Navier-Stokes equations with a time accuracy of $O(\Delta t^2)$. As far as stability is concerned it is evident from the practical computations that the schemes are not unconditionally stable. We found that schemes (7.7) and (7.8) were unstable at $\mathrm{Re} = 1500$ and $\Delta t = 0.1$ with refined grid at the outlet, the theoretical result of § 6 notwithstanding. However,

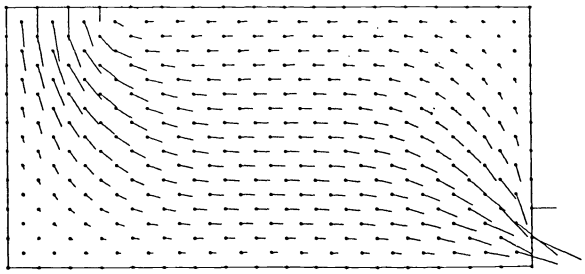


FIG. 9.11. Re = 100, $u_{max} = 1.2$, $t = 3.5$.

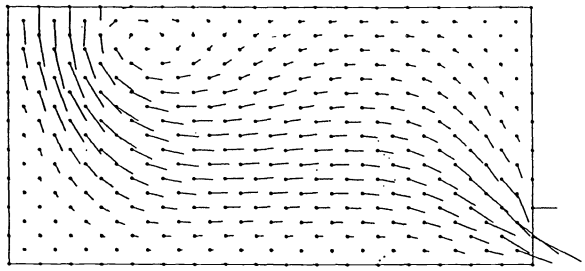


FIG. 9.12. Re = 100, $u_{max} = 0.65$, $t = 3.75$.

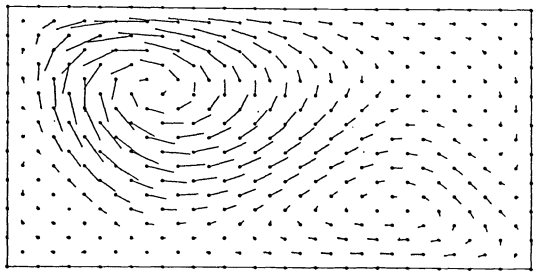


FIG. 9.13. Re = 100, $u_{max} = 0.19$, $t = 4.0$.

this result was obtained under simplifying assumptions of periodicity and only said something about the effect of pressure correction on a scheme that in itself was unconditionally stable (Crank-Nicolson) whereas it is not sure, that the ADI schemes of Chapter 7 are unconditionally stable. It is probably true (though by no means proved) that pressure correction does not deteriorate the stability of the underlying scheme. Unconditional stability of the ADI schemes, however, is usually proved under the assumption of commutativity of the split operators and this assumption is surely not satisfied in our practical examples. Another source of instability might be the nonlinearity of the convective terms and the fact that instability occurred at high Re numbers makes it probable that this is the main source. The equations will then be almost hyperbolic, giving an amplification factor for the Crank-Nicolson scheme of

$$((1-\tfrac{1}{2}\Delta t\mu)-\tfrac{1}{2}i\Delta t\nu)/((1+\tfrac{1}{2}\Delta t\mu)+\tfrac{1}{2}i\Delta t\nu). \tag{10.1}$$

For high Reynolds numbers μ will be very small and high frequency components (ν large) will almost have amplification 1. Nonlinear effects may let the amplification grow above 1 for high frequency components. It is debatable whether a construction

such as

$$\frac{u^{n+1}-u^n}{\Delta t}=\left(\frac{1}{2}+\varepsilon\right)f(u^{n+1})+\left(\frac{1}{2}-\varepsilon\right)f(u^n) \qquad (\varepsilon \text{ small, positive}) \tag{10.2}$$

which cures this problem should be advocated, because it probably introduces so much artificial viscosity, that we solve a different problem. What could be done however is make ε depend on Δt, something like $\varepsilon = c\Delta t$. This does not change the order of accuracy and probably has better stability properties. So far we have no experience with this scheme.

Acknowledgments. The author is indebted to his students C. R. J. Kieboom and A. van Ommen for performing most of the calculations reported here. The latter was also responsible for the improvement of scheme (7.7) which finally resulted in scheme (7.8).

REFERENCES

[1] M. BRAZA, *Simulation numérique du décollement instationnaire externe par une formulation vitesse-pression, Application à lécoulement autour d'un cylindre*, Thèse, Institut National Polytechnique de Toulouse, 1981.

[2] T. CEBECI, R. S. HIRSCH, H. B. KELLER AND P. G. WILLIAMS, *Studies of numerical methods for the plane Navier-Stokes equations*, Comp. Meth. Appl. Mech. Eng., 27 (1981), pp. 13-44.

[3] A. J. CHORIN, *Numerical solution of the Navier-Stokes equations*, Math. Comp., 22 (1968), pp. 745-762.

[4] G. FAIRWEATHER AND A. R. MITCHELL, *A new computational procedure for* ADI *methods*, SIAM J. Numer. Anal., 4 (1967), pp. 163-170.

[5] K. GODA, *A multistep technique with implicit difference schemes for calculating two or three-dimensional cavity flows*, J. Comp. Phys., 30 (1979), pp. 76-95.

[6] P. GRESHO, R. L. LEE AND R. L. SANI, *On the time dependent solution of the incompressible Navier-Stokes equations in two and three dimensions*, in: Recent Advances in Numerical Methods in Fluids, C. Taylor, R. Morgan eds., Vol. 1, Pineridge Press, Swansea, 1981.

[7] P. M. GRESHO AND R. L. LEE, *Don't suppress the wiggles—they're telling you something*, in Proc. Special Symposium on Finite Element Methods for Convection Dominated Flows, Winter Annual ASME meeting, December 2-7, 1979, New York, 1980.

[8] F. J. HARLOW AND J. E. WELCH, *Numerical calculation of time dependent viscous incompressible flow of fluids with free surface*, Phys. Fluids, 8 (1965).

[9] M. A. LESCHZINER, *Practical evaluations of three finite difference schemes for the computation of steady state recirculating flows*, Comp. Meth. Appl. Mech. Eng., 23 (1980), pp. 293-312.

[10] A. SEGAL, *Aspects of numerical methods for elliptic singular perturbation problems*, this Journal, 3 (1982), pp. 327-349.

[11] S. V. PATANKAR, *Numerical Heat Transfer and Fluid Flow*, McGraw-Hill, New York, 1980.

[12] R. TÉMAM, *Sur l'approximation de la solution de Navier-Stokes par la méthode des pas fractionnaires*, I, II, Arch. Rat. Mech. Anal., 32 (1969), pp. 135-153; pp. 377-385.

[13] TSAI TIMIN AND M. N. ESMAIL, *A comparative study of central and upwind difference schemes using the primitive variables*, Int. J. Num. Meth. Fluids, 3 (1983), pp. 295-305.

[14] P. WESSELING, *Theoretical and practical aspects of a multigrid method*, this Journal, 3 (1982), pp. 387-407.

SIAM J. SCI. STAT. COMPUT.
Vol. 7, No. 3, July 1986

013

AN APPLICATION OF SYSTOLIC ARRAYS TO LINEAR DISCRETE ILL-POSED PROBLEMS*

LARS ELDÉN† AND ROBERT SCHREIBER‡

Abstract. The application of systolic arrays to linear, discrete ill-posed problems is considered. The regularization method of Tikhonov–Phillips is used, and a new systolic array is proposed for the case when the matrix is banded. The array implements a QR-decomposition by plane rotations, and it consists of approximately $k^2/2$ processor elements, where k is the bandwidth of the matrix. The computation time for the QR-decomposition is $O(n)$ (n is the dimension of the problem). A generalization of the array is also mentioned. The reduction to banded form is briefly discussed.

Key words. systolic array, ill-posed, regularization, QR-decomposition, band structure

AMS(MOS) subject classifications. 65F20, 65F25, 68A05

1. Introduction. In this paper we discuss the application of systolic arrays to linear discrete ill-posed problems, i.e., linear systems of equations

$$Kf = g, \tag{1.1}$$

where K is a very ill-conditioned $m \times n$ matrix, $m \geqq n$. Such problems arise when Fredholm equations of the first kind are discretized. Because of the ill-conditioning the problem of solving (1.1) is unstable; small perturbations of the data will lead to very large errors in the solution, which usually show up as large oscillations.

One way of stabilizing the problem is to introduce a penalty on nonsmooth solutions. Then (1.1) is replaced by

$$\min_f \{\|Kf - g\|^2 + \lambda^2 \|Lf\|^2\}, \tag{1.2}$$

where $\|\cdot\|$ denotes the Euclidean vector norm, and the matrix L is equal to the identity matrix or a discretization of a differentiation operator. This is the *regularization method* of Tikhonov [18], Phillips [14]. The smoothness of the solution is controlled by choosing a suitable value of the *regularization parameter* λ. For a discussion of ill-posed problems, see [19], [4], [13]. Numerical algorithms are surveyed in [1], [20].

Systolic arrays [11] are highly parallel computing structures specific to particular computing tasks. The designs consist of one- or two-dimensional arrays of identical processor elements. Communication of data and control signals occur only between neighboring elements. There are a number of papers dealing with the application of systolic arrays to algorithms in numerical linear algebra. Of special interest here are the following: [2], [3], [7], [10], [15].

In §§ 2 and 3 we consider the case when L is equal to the identity matrix. If K is a square band matrix, then (1.2) can be solved efficiently using the algorithm given in [6]. This algorithm is based on a QR-decomposition computed by plane (Givens) rotations. A new systolic array, which implements this algorithm, is described in § 2. The array consists of approximately $k^2/2$ processing elements, and the time required is $O(n+k)$ (n is the number of unknowns and k is the bandwidth of K).

Numerical algorithms for solving (1.2), when K is a dense matrix, are usually based on a reduction of K to compact form, such as the singular value decomposition (SVD) [8], or a bidiagonalization [5]. In the context of systolic arrays it turns out to

* Received by the editors January 24, 1984, and in revised form April 26, 1985.
† Department of Mathematics, Linköping University, S-581 83 Linköping, Sweden.
‡ Department of Computer Science, Stanford University, Stanford, California 94305.

be advantageous to reduce K to upper triangular band form. The array, which has been described earlier [15], consists of kn processor elements, and the time required is $O((m+n)n/k)$. The reduction to band form is discussed in § 3.

The algorithm in [6] treats the case when both K and L are band matrices. The systolic array given in § 2 can be generalized to cover this case. This generalization is described in § 4.

If $L \neq I$ and K is dense, the method of Eldén [5] must be used to transform into a problem for which $L = I$. This transformation involves QR factorizations and can also be handled by a systolic array.

It is also possible to further develop the ideas of this paper to treat more general problems with band structure. This is done in a paper by one of the authors [16].

2. A systolic array for banded K, and $L = I$. We first give an outline of the algorithm for the solution of (1.2), when K is an $n \times n$ upper triangular band matrix, and L is equal to the identity matrix. Later in this section we consider the case when K is not upper triangular. (1.2) is solved by reducing the matrix of coefficients

$$Q^T \begin{bmatrix} \Lambda & \vdots & 0 \\ K & \vdots & g \end{bmatrix} = \begin{bmatrix} \bar{K} & \vdots & \bar{g}_1 \\ 0 & \vdots & \bar{g}_2 \end{bmatrix}, \tag{2.1}$$

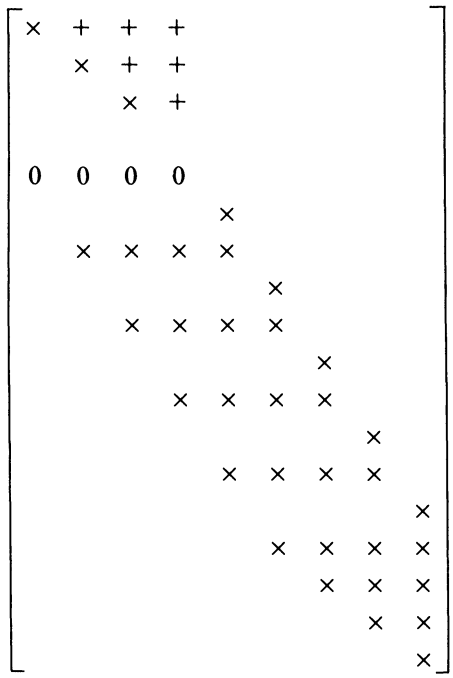
where $\Lambda = \lambda L = \operatorname{diag}(\lambda_1, \lambda_2, \cdots, \lambda_n)$, (it turns out to be more instructive not to assume that the diagonal elements are equal), $\bar{K}$ is an upper triangular band matrix with the same bandwidth as K, and Q is orthogonal. The solution of (1.2) is then obtained by solving $\bar{K}f = \bar{g}_1$ (a systolic array for this computation is described e.g. in [11, p. 272]). The orthogonal transformation in (2.1) is made up of a sequence of plane rotations. We explain the algorithm in terms of a small example, with $n = 9$ and the bandwidth k equal to 4. For more details, see [6]. We denote nonzero elements by $\times$. Elements zeroed in the present transformation are represented by 0, and new nonzero elements by $+$.

The rows of the matrix (2.1) are reordered according to the maximum column subscript of nonzero elements:

$$\begin{bmatrix}
\times & & & & & & & & \\
 & \times & & & & & & & \\
 & & \times & & & & & & \\
 & & & \times & & & & & \\
\times & \times & \times & \times & & & & & \\
 & & & & \times & & & & \\
 & \times & \times & \times & \times & & & & \\
 & & & & & \times & & & \\
 & & \times & \times & \times & \times & & & \\
 & & & & & & \times & & \\
 & & & \times & \times & \times & \times & & \\
 & & & & & & & \times & \\
 & & & & \times & \times & \times & \times & \\
 & & & & & \times & \times & \times & \times \\
 & & & & & & \times & \times & \times \\
 & & & & & & & \times & \times \\
 & & & & & & & & \times
\end{bmatrix}.$$

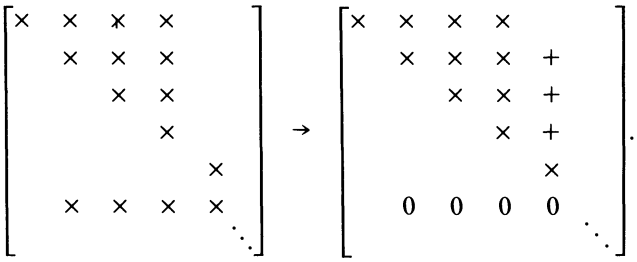
The elements in the 5th row are zeroed by a sequence of rotations in the (1, 5), (2, 5), (3, 5), and (4, 5) planes:

(2.2)



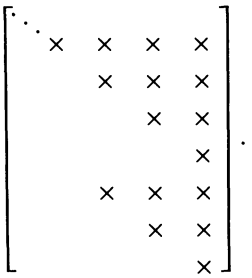
The row of zeros is moved to the bottom of the matrix, and the sixth row is zeroed by a sequence of rotations in the (2, 6), (3, 6), (4, 6), and (5, 6) planes:

(2.3)



All the subsequent steps but the last few (the last three in our example) are identical to that illustrated in (2.3). A $k \times k$ upper triangular submatrix is used to completely annihilate a row consisting of k nonzero elements. Fill-in is created only along the right edge of the submatrix. Note that no rotations are needed to zero the elements from the diagonal matrix Λ in (2.1).

Before the last three rows in (2.2) are processed, we have the following picture:



If we introduce zeros and augment the matrix

(2.4)

$$\begin{bmatrix}
\ddots & & & & & & & \\
\times & \times & \times & \times & & & & \\
 & \times & \times & \times & 0 & & & \\
 & & \times & \times & 0 & & & \\
 & & & \times & 0 & & & \\
 & & & & 0 & & & \\
 & \times & \times & \times & 0 & & & \\
 & & & & & 0 & & \\
 & & \times & \times & 0 & 0 & & \\
 & & & & & & 0 & \\
 & & & \times & 0 & 0 & 0 &
\end{bmatrix},$$

we see that we can use the same algorithm as before to zero these last rows. Note, that no fill-in is created.

The case when K is not upper triangular can be treated analogously. Assume e.g. that K is a lower triangular 9×9 matrix with band width 4. Make K upper triangular by introducing extra zeros at the top left corner, and augment Λ correspondingly. The reordered matrix then becomes

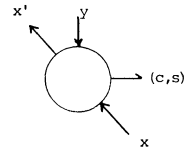
$$\begin{bmatrix}
0 & & & & & & & & \\
 & 0 & & & & & & & \\
 & & 0 & & & & & & \\
 & & & \times & & & & & \\
0 & 0 & 0 & \times & & & & & \\
 & & & & \times & & & & \\
 & 0 & 0 & \times & \times & & & & \\
 & & & & & \times & & & \\
 & & 0 & \times & \times & \times & & & \\
 & & & & & & \times & & \\
 & & & \times & \times & \times & \times & & \\
 & & & & & & & \times & \\
 & & & & \times & \times & \times & \times & \\
 & & & & & & & & \times \\
 & & & & & \times & \times & \times & \times \\
 & & & & & & & & \times \\
 & & & & & & \times & \times & \times \\
 & & & & & & & & \times \\
 & & & & & & & \times & \times \\
 & & & & & & & & \times \\
 & & & & & & & \times & \times
\end{bmatrix},$$

and it is immediately seen that the above algorithm is applicable.

2.1. The processor elements. To realize this algorithm, we need two types of processor elements: the *boundary cell*, which generates a plane rotation, and the *internal cell*, which applies and propagates a rotation (the reason why we call them boundary and internal cells will be apparent in § 2.2). These cells are defined in Fig. 2.1.

BOUNDARY CELL

if $y = 0$ **then**
begin
$c := 1;$
$s := 0;$
$x' := x$
end
else
begin
$x' := (x^2 + y^2)^{1/2};$
$c := x/x';$
$s := y/x'$
end;



INTERNAL CELL

$x' := c * x + s * y;$
$y' := -s * x + c * y;$

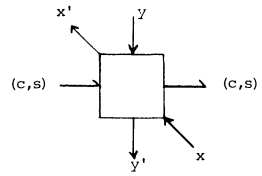


FIG. 2.1. *Cell definitions.*

It is assumed that the cells are connected in a synchronized network, where the time between the clock pulses is long enough to perform the required operation. In the absence of input, data lines will assume the value 0. The cells defined here are almost identical to those in [10], cf. also [7].

2.2. The array. The new array is illustrated in Fig. 2.2, for the case when the bandwidth k is equal to 4. There is a correspondence between the upper triangular

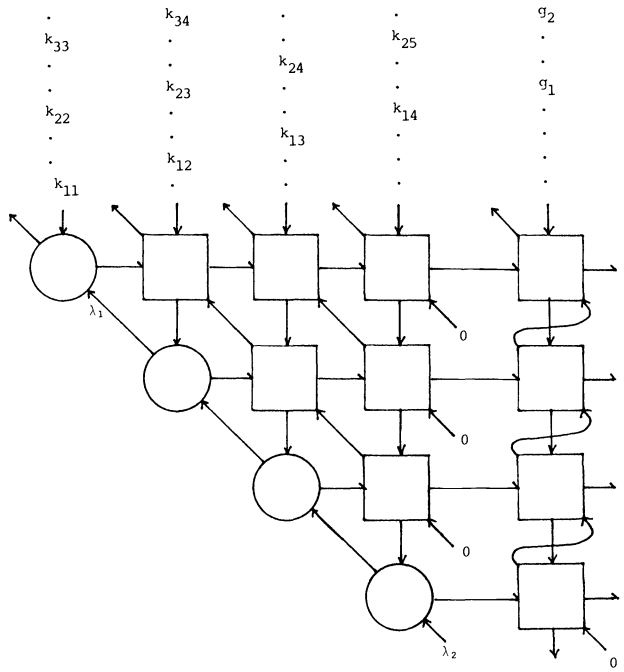


FIG. 2.2. *Systolic array for the band algorithm,* $k = 4$.

matrix in (2.3), and the triangular array of processor elements, in the sense that the rotations applied to a certain row in (2.3) are constructed and propagated along the corresponding row in the systolic array.

It is assumed that the elements of K and the right-hand side g are coming in the the top, and the elements of the diagonal matrix Λ from the bottom, so that k_{11} meets λ_1 at the top left boundary cell. This is the start of the transformations illustrated in (2.2).

The result, i.e. the elements of $\bar{K}$ and $\bar{g}_1$, come out at the top (by the data lines pointing northwest). The organization of the output is illustrated in Fig. 2.3.

To illustrate the data flow, we give a partial trace of three consecutive time steps in Fig. 2.4 (the numbers in the cells denote indices of the elements in K that are being operated on).

It is seen from Fig. 2.2 that before λ_1 meets k_{11} at the top boundary cell, and the actual computations start, the λ's have to be propagated from below. This takes k time steps. At every time step two-thirds of the cells are inactive. Therefore, to compute the decomposition (2.1) takes $k+3n+k$ time steps, where the last term is due to the fact that the rows of $\bar{K}$ are output in skewed order. Note, however, that three problems can be solved during the same time, if the matrix elements are interleaved. This is

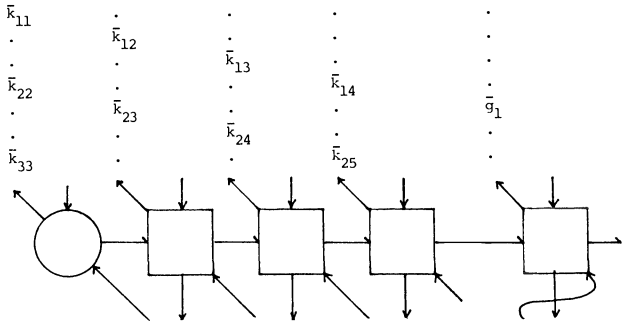


FIG. 2.3. *Output from the array.*

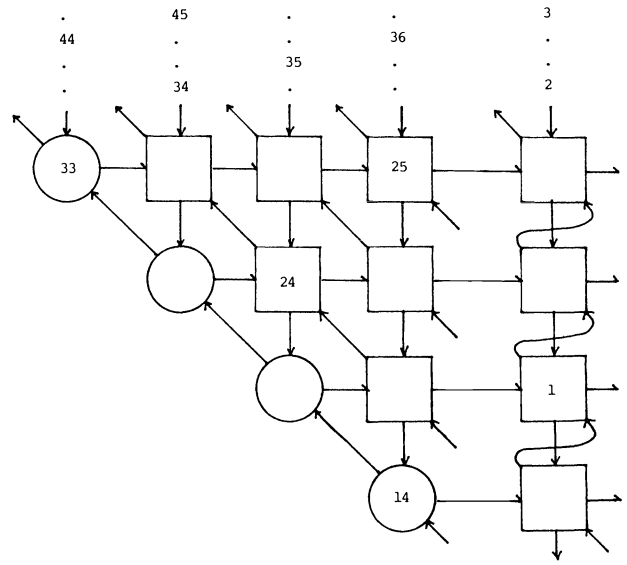


FIG. 2.4(a). *Trace of the dataflow during three consecutive time steps.*

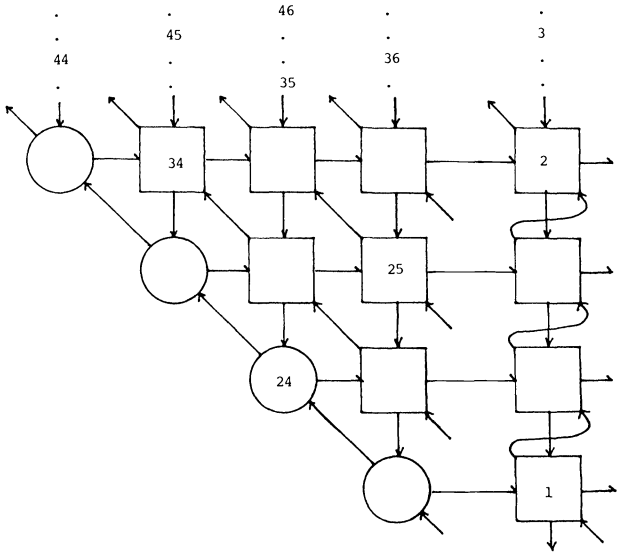


FIG. 2.4(b)

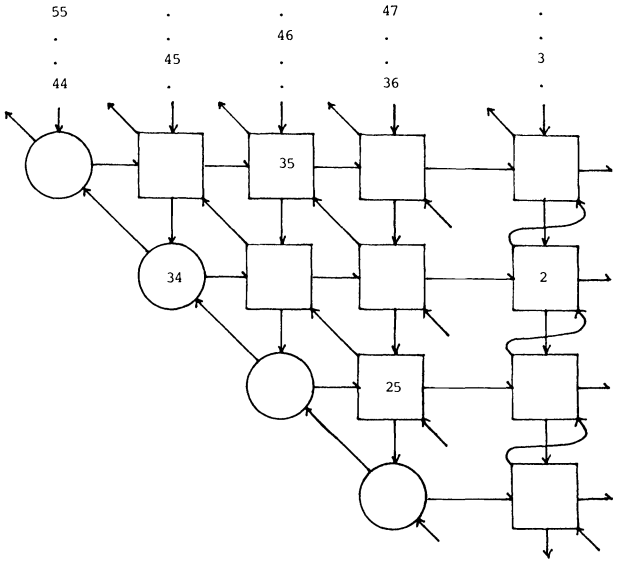


FIG. 2.4(c)

useful in the context of regularization, where it is necessary to solve (1.2) for several values of the regularization parameter λ. Thus interleaving becomes especially simple here: The elements of k are repeated three times, and only the λ's have to be exchanged.

3. Reduction to band form. In this section we discuss the reduction of the problem

$$\min_f \{\|Kf-g\|^2+\lambda^2\|f\|^2\}, \tag{3.1}$$

where K is assumed to be a dense $m \times n$ matrix, $m \geqq n$, to an equivalent problem with a square band matrix.

If we decompose

$$K = U\begin{bmatrix} S \\ 0 \end{bmatrix} V^T, \tag{3.2}$$

where S is an $n \times n$ matrix, and U and V are orthogonal $m \times m$ and $n \times n$ matrices respectively, we get the equivalent problem

$$\min_{\bar{f}} \{\|S\bar{f} - \bar{g}_1\|^2 + \lambda^2 \|\bar{f}\|^2\}, \tag{3.3a}$$

where

$$\bar{g} = \begin{bmatrix} \bar{g}_1 \\ \bar{g}_2 \end{bmatrix} = U^T g, \tag{3.3b}$$

$$\bar{f} = V^T f. \tag{3.3c}$$

The decomposition (3.2) is chosen so that it is faster to compute the solution of (3.1) from (3.3) than using the original form (3.1). Remember that it is usually necessary to solve (3.1) for several values of λ.

Note, that the closer to a diagonal matrix we choose S, the cheaper and simpler (in terms of computation time and hardware) it becomes to solve (3.3a). For example, if S is diagonal, then we can solve (3.3a) in one time step using n processor elements, or in n time steps using only one processor. On the other hand, the closer to a diagonal matrix we choose S, the more expensive and complicated it becomes to compute the decomposition (3.2). We want to solve (3.1) using systolic arrays in such a way that these costs are balanced, both in terms of computation time and complexity.

If S is chosen to be diagonal, then we have the SVD [9]. Using this, we can write down the solution of (3.1) explicitly [13], and it can be computed very efficiently. Several methods for computing the SVD using systolic arrays have been examined. Some are based on Hestenes method, and one almost linear time method has recently been proposed [3]. This method requires $O(n^2)$ processor elements.

Schreiber [15] has proposed a method for computing the SVD, which is a variant of the standard algorithm [9]. The matrix K is first reduced to upper triangular banded form. Then follows an iterative procedure, where the band matrix is reduced to diagonal form.

In the preceding section we saw that it is possible to efficiently solve problems with band structure. Therefore we can avoid the iterative part of the SVD algorithm, and still have an efficient method of solving (3.1). The same idea was used in [5], where, in connection with standard computer architecture, S was taken bidiagonal.

In the rest of this section we give a brief outline of the reduction to upper triangular band form using systolic arrays. A more detailed description is given in [15], [17]. For simplicity we here assume that $m = n$.

The systolic array is organized in trapezoidal form, with $k-1$ rows and n columns. If we send the matrix

$$K = \begin{bmatrix} K_{11} & K_{12} \\ K_{21} & K_{22} \end{bmatrix},$$

where K_{11} is $(k-1) \times (k-1)$, through the array, the output is

$$U_1^T K = \begin{bmatrix} S_{11} & C_{12} \\ 0 & C_{22} \end{bmatrix},$$

where U_1 is orthogonal, and S_{11} is an upper triangular $(k-1) \times (k-1)$ matrix. Thus the array applies an orthogonal transformation, realized by a sequence of plane

rotations, which zeros the first $k-1$ columns below the main diagonal. Then if we send $[C_{12}^T C_{22}^T]$ through the array we get

$$V_1^T[C_{12}^T \quad C_{22}^T] = \begin{bmatrix} T_{12}^T & \bar{K}_{22}^T \\ 0 & \bar{K}_{23}^T \end{bmatrix},$$

where T_{12} is $(k-1)\times(k-1)$ lower triangular. The result of these two transformations is

$$S^{(1)} = \begin{bmatrix} S_{11} & T_{12} & 0 \\ 0 & \bar{K}_{22} & \bar{K}_{23} \end{bmatrix}.$$

Now continue the process using

$$K^{(1)} = [\bar{K}_{22} \quad \bar{K}_{23}]$$

in place of K. After $J = \lceil n/k \rceil$ such steps, we have produced a k-diagonal, upper triangular matrix

$$S = \begin{bmatrix} S_{11} & T_{12} & & & \\ & S_{22} & T_{23} & & \\ & & \ddots & \ddots & \\ & & & & T_{J-1,J} \\ & & & & S_{JJ} \end{bmatrix},$$

such that $K = USV^T$. It is necessary to apply the left-hand rotations to the right-hand side (cf. (3.3b)), and accumulate the right-hand rotations to form the matrix V (cf. (3.3c)). This can be done by the array.

The total time for computing the band matrix S and the orthogonal matrix V is $O(n^2/k)$.

4. An array for the case of banded L. In some applications it is necessary to choose L equal to a discretization of a differentiation operator. Then L is usually a $p \times n$ band matrix, $p < n$. In this case (1.2) can be transformed to the standard form (3.1) in spite of the fact that L^{-1} does not exist [5]. The problem can then be solved using the methods described in the preceding sections. However, if K is also a band matrix, this transformation destroys the band structure and should therefore be avoided.

The algorithm given in [6] for the computation of a QR-decomposition

$$Q^T \begin{bmatrix} \lambda L \\ K \end{bmatrix} = \begin{bmatrix} R \\ 0 \end{bmatrix}, \tag{4.1}$$

where both K and L are band matrices, can be implemented in a systolic system. This application is described in a general setting in [16]. We here give a brief description of a simple special case, where it is assumed that K and L are upper triangular band matrices with bandwidths 4 and 3 respectively. The dimensions are assumed to be $n \times n$ and $(n-2) \times n$.

In a typical step of the algorithm we make the transformation (see [6])

$$\begin{bmatrix} \times & \times & \times & \times & \\ & \times & \times & \times & \\ & & \times & \times & \\ & & & \times & \\ & \times & \times & \times & \times \\ & & \times & \times & \times \\ & & & & \ddots \end{bmatrix} \rightarrow \begin{bmatrix} \times & \times & \times & \times & \\ & \times & \times & \times & + \\ & & \times & \times & + \\ & & & \times & + \\ & 0 & 0 & 0 & \times \\ & & 0 & 0 & 0 \\ & & & & \ddots \end{bmatrix}$$

Comparing this to (2.3), we see that here two extra matrix elements have to be zeroed by rotations. Therefore, in the systolic array two extra boundary cells are needed to generate these rotations.

The general idea in [16] is to interleave two arrays of the type of § 2. The appropriate array for this example is shown in Fig. 4.1. There is a 4×4 triangular *outer array* that

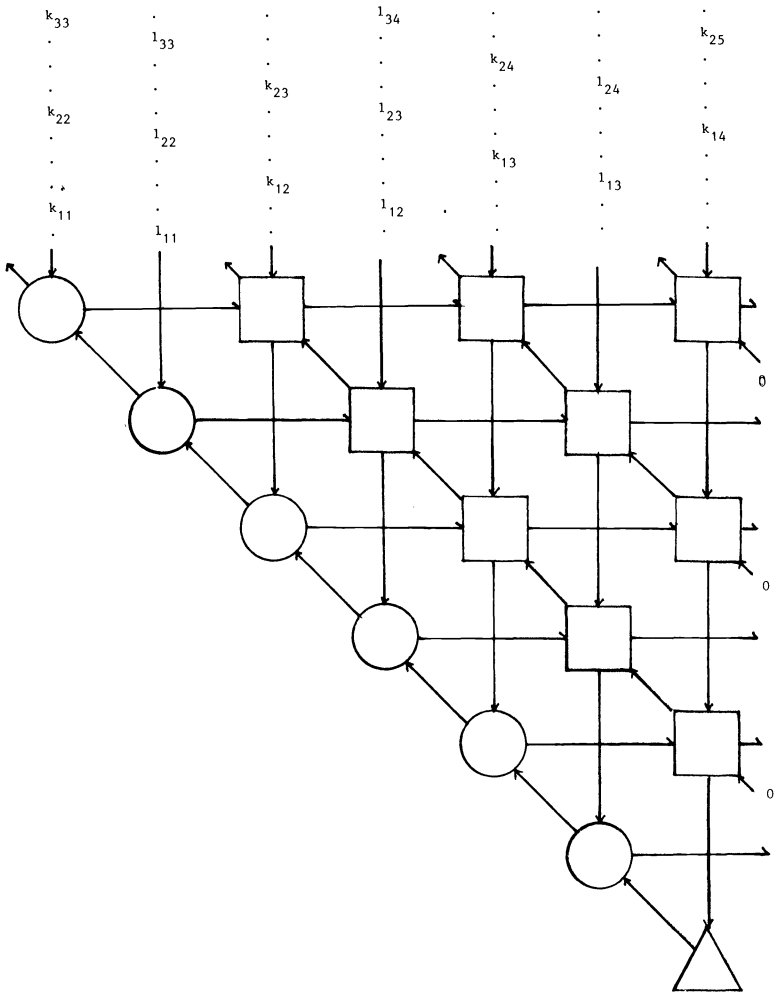


FIG. 4.1. *Systolic array for banded K and L.*

zeros elements of K, and a 3×3 triangular *inner array*, nested within the outer array, that zeros elements of L. Note that a new cell type, shown as a triangle, is used. This is a *delay cell* that simply holds its input for one clock cycle, then transmits it. Remember that in the absence of input, data lines will assume the value zero.

In general it is essential that, if the outer array is $q\times q$, then the inner array be $(q-1)\times(q-1)$ or $q\times q$. If the bandwidths of K and L differ by more than 1, the thinner must be expanded by including additional zero diagonals in its band. Thus if r is the bandwidth of K and s is the bandwidth of L, we require that $|r-s|\leqq 1$. If $s>r$ we send L into the outer array, while if $r\geqq s$ we send K into the outer array.

5. Conclusions. We have shown that discrete ill-posed problems of the form (1.2) can be efficiently solved, for several values of λ, when K is either dense or banded and $L = I$ or L is banded, using systolic arrays (see Fig. 5.1). When $L = I$, the array of § 2 can be used. If K is dense it must first be reduced to a band matrix as shown in § 3. If $L \neq I$ and both K and L are banded, the array of § 4 is used. If $L \neq I$ and K is dense, however, the method of Eldén [5] must be used to transform into a problem for which $L = I$. This transformation involves QR-factorizations and can also be handled by a systolic array.

	$L = I$	L banded
K dense	§ 3	[5]
K banded	§ 2	§ 4

FIG. 5.1. *Strategy for the solution of* (1.2).

REFERENCES

[1] Å. BJÖRCK AND L. ELDÉN, *Methods in linear algebra for ill-posed problems*, Linköping University, Dept. Mathematics, Report LiTH-Mat-R-33-1979.

[2] R. P. BRENT AND F. T. LUK, *A systolic architecture for the singular value decomposition*, Cornell University, Dept. Computer Science, Report TR-82-522, Ithaca, New York, 1982.

[3] R. P. BRENT, F. T. LUK AND C. VAN LOAN, *Computation of the singular value decomposition using mesh-connected processors*, Cornell University, Dept. Computer Science, Report TR-82-528, Ithaca, New York, 1983.

[4] F. R. DE HOOG, *Review of Fredholm equations of the first kind*, in The Application and Numerical Solution of Integral Equations, R. S. Anderssen, F. R. de Hoog and M. A. Lukas, eds., Sijthoff & Nordhoff, Alphen an den Rijn, Netherlands, 1980.

[5] L. ELDÉN, *Algorithms for the regularization of ill-conditioned least squares problems*, BIT, 17 (1977), pp. 134-145.

[6] ———, *An algorithm for the regularization of ill-conditioned, banded least squares problems*, this Journal, 5 (1984), pp. 237-254.

[7] W. M. GENTLEMAN AND H. T. KUNG, *Matrix triangularization by systolic arrays*, Proc. SPIE, vol. 298; Real-Time Signal Processing IV, Society of Photo-Optical Instrumentation Engineers, Bellingham, WA, 1981.

[8] G. H. GOLUB, *Some modified eigenvalue problems*, SIAM Rev., 15 (1973), pp. 318-334.

[9] G. H. GOLUB AND C. REINSCH, *Singular value decomposition and least squares solutions*, Numer. Math., 14 (1970), pp. 403-420.

[10] D. E. HELLER AND I. C. F. IPSEN, *Systolic networks for orthogonal decompositions, with applications*, this Journal, 4 (1983), pp. 261-269.

[11] H. T. KUNG AND C. L. LEISERSON, *Systolic arrays (for VLSI)*, in Sparse Matrix Proceedings, Society for Industrial and Applied Mathematics, Philadelphia, PA, 1978, pp. 256-282.

[12] C. MEAD AND L. CONWAY, *Introduction to VLSI Systems*, Addison-Wesley, Reading, MA, 1980.

[13] G. F. MILLER, *Fredholm equations of the first kind*, in Numerical Solution of Integral Equations, L. M. Delves and J. Walsh, eds., Clarendon Press, Oxford, 1974.

[14] D. L. PHILLIPS, *A technique for the numerical solution of certain integral equations of the first kind*, J. Assoc. Comput. Mach., 9 (1962), pp. 84-97.

[15] R. SCHREIBER, *A systolic architecture for singular value decomposition*, in Proc. 1er Colloque International sur les Methodes Vectorielles et Paralleles en Calcul Scientifiques, Electricité de France, Bulletin de la Direction des Etudes et Recherches, Série C, No. 1-1983.

[16] ———, *On systolic array methods for band matrix factorization*, Royal Inst. of Technology, Department of Numerical Analysis and Computer Science, Report TRITA-NA-8316, Stockholm, 1983.

[17] R. SCHREIBER AND P. J. KUEKES, *Systolic linear algebra machines in digital signal processing*, in VLSI and Modern Signal Processing, S. Y. Kung, H. Whitehouse and T. Kailath, eds., Prentice-Hall, Englewood Cliffs, NJ, 1984.

[18] A. N. TIKHONOV, *Solution of incorrectly posed problems and the regularization method*, Dokl. Akad. Nauk SSSR, 151 (1963), pp. 501-504; Soviet Math. Dokl., 4 (1963), pp. 1035-1038.

[19] A. N. TIKHONOV AND V. Y. ARSENIN, *Solutions of Ill-Posed Problems*, Winston & Sons, Washington DC, 1977.

[20] J. M. VARAH, *A practical examination of some numerical methods for linear discrete ill-posed problems*, SIAM Rev., 21 (1979), pp. 100-111.

SIAM J. SCI. STAT. COMPUT.
Vol. 7, No. 3, July 1986

014

DATA STRUCTURES FOR ADAPTIVE GRID GENERATION*

MARSHA J. BERGER†

Abstract. This paper describes data structures and algorithms for the automatic generation of adaptive subgrids, a technique used with adaptive mesh refinement for solving partial differential equations. Our algorithms generate a nested sequence of finer and finer grids on an underlying coarse grid. There are two aspects to the data structures. Trees are used to do the grid management for this type of grid structure. Secondly, the automatic grid generation algorithms use data structures with special nearest neighbor properties. Examples of grids from actual adaptive numerical computations are shown.

Key words. adaptive grid generation, data structures

AMS(MOS) subject classifications. 65M99, 68G10

1. Introduction. In this paper we discuss the use of data structures in tackling an adaptive grid generation problem. These grids are to be used in the solution of partial differential equations (pdes) by finite difference schemes using adaptive mesh refinement. In this approach, a nested sequence of locally uniform fine grids is superimposed on an underlying coarse grid until a given accuracy criterion is attained. These methods were originally developed for the solution of hyperbolic pdes (see [3], [7]), but we believe that our algorithms and data structures have a wider applicability. For example, we have extended our adaptive hyperbolic method to handle steady-state transonic flow [5]. Even in this steady-state case, the grid management methods need to be dynamic, however, since the regions needing refinement (for example the shock location) are not known in advance. Our grid generation package is also currently being used in a mesh refinement program for solving elliptic pdes [8]. It is natural to use these adaptive grids in conjunction with multigrid methods [16]. Indeed, we have begun such work for the steady Euler equations [6].

There are two aspects to the use of data structures in this context. The first is storing and manipulating grids. We use trees and linked lists to keep track of this irregular grid structure. The data structures themselves are not uncommon; it is their application to the numerical solution of pdes that is new. The second aspect is our method of automatic grid generation, which is based on data structures with special nearest neighbor properties. Our grid generation is at most a two pass algorithm which clusters grid points and fits locally regular rectangles for the fine grids.

Before describing our data structures, we justify our use of locally uniform, possibly rotated, rectangles as the building blocks of a general adaptive mesh refinement method. Rectangles have the simplest user interface. The same integrator for the coarse grid may be used to integrate all the fine grids too. By separating the integrator from the adaptivity strategy, an off-the-shelf integrator can be used without modification, as was done in the transonic flow calculations of [5]. This simplifies the programming for each new application and allows an easier front end. For rotated rectangular grids, it is easy to automatically transform the difference equations into the rotated coordinate system. If the area needing refinement is diagonal to the grid, a smaller total area is refined if we allow the rectangles to rotate too. Moreover, in some calculations it is numerically advantageous to use a coordinate system which is approximately locally normal and tangent to a front in the solution. Using multiple oriented subgrids allows

* Received by the editors March 22, 1983, and in final revised form May 20, 1985. This research was supported in part by the U.S. Department of Energy under contract DEAC0276ER03077-V.

† Courant Institute of Mathematical Sciences, New York University, New York, New York 10012.

this to happen. This would also permit refinement in only one coordinate direction. However, there is additional overhead associated with rotated grids, in both the interpolation procedures and the interface equations needed between the grids, particularly if the interface conditions need to be conservative. This overhead will be discussed later. Our algorithms can, therefore, be used to create rotated as well as nonrotated subgrids. For example, in Fig. 1a we show a coarse grid where the grid points which need refinement are marked with an *X*. Fig. 1b shows some typical fine grids our grid generation package produces in this situation. If we do not permit rotation, the subgrids our package produces are shown in Fig. 1c.

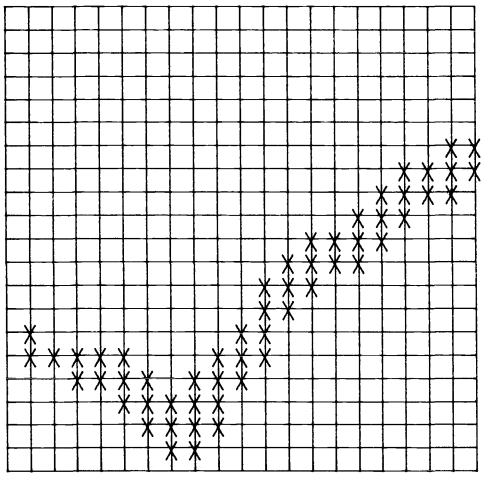


FIG. 1a. *Coarse grid with grid points needing refinement marked with an 'X'.*

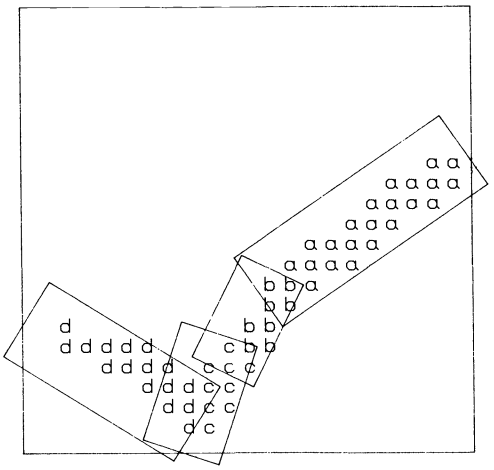


FIG. 1b. *Grid generation example using rotated rectangles for the fine grids that enclose the marked grid points.*

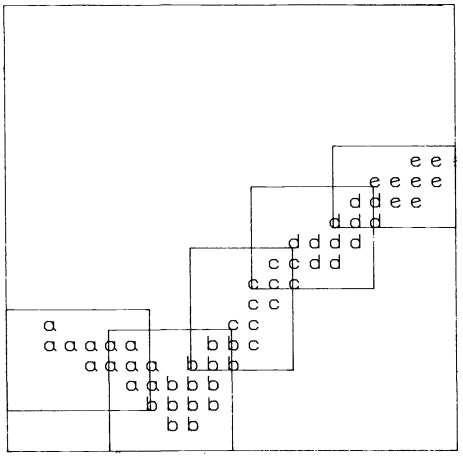


FIG. 1c. *Grid generation example using unrotated rectangles for the fine grids.*

In either case, this use of locally embedded fine grids should be contrasted with moving grid points, where regions of a grid are refined by "attracting" points into the region at the expense of resolution in the rest of the region (see [15] and references there). Such methods often have difficulty in controlling grid skewness, which can degrade accuracy. This will be even more of a problem for three-dimensional calcula-

tions. However, they do not have the problem of internal interfaces found in our method of refinement.

We wish to make two points about the grid structure. These fine grids are not merged into the underlying coarse grid, but are kept separately defined, each with its own solution storage vector. In this way, we take advantage of the local uniformity of each grid. The coarse grid points underneath a fine grid are in some sense wasted, unless they participate in the solution process itself, as in multigrid methods. However, even if this is not the case, we believe this is a smaller price to pay than alternatives which involve cell by cell or column oriented refinement and use much more storage overhead for pointers. Such methods typically have difficulties with multiply connected regions and inhibit the vectorization of integration algorithms.

The second point we make about our grid structure is the following. It may happen that many levels of refinement are needed to get sufficiently fine local resolution. This is no problem for the grid generation routines, only for the data structures that keep track of the grids. The grid generation algorithm is applied to the flagged points at each grid level to create the next finer level of grids. Fig. 2 illustrates an initial grid

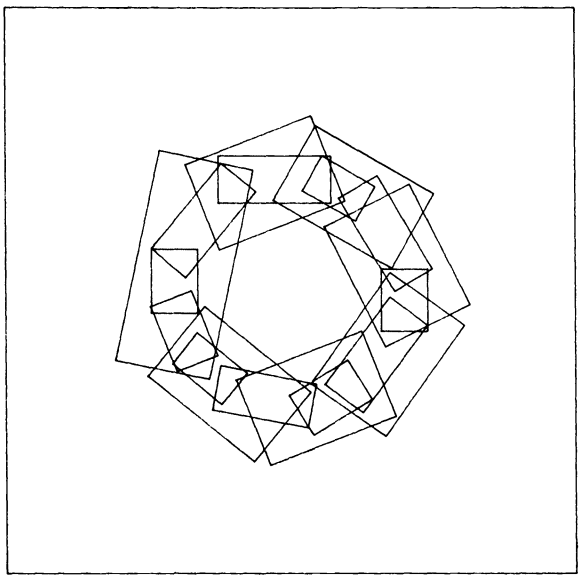


FIG. 2. *There are two levels of refinement around the circular expanding shock, inside the coarse grid.*

configuration with 3 levels of grids for a radially symmetric expanding shock problem. An error estimator applied to the coarse grid at level 0 yields 7 level 1 grids, each approximately 5 coarse mesh widths wide. When these grids have their error estimated, the grid generation algorithm yields 11 level 2 grids. Now, the level 2 grids are approximately 5 fine mesh widths wide. In this figure the reader can also see how one level 2 grid can be (partially) nested in one or more level 1 grids.

Given this nested hierarchy of finer and finer rectangular subgrids, we can now describe the two main roles that data structures play. In § 2, we describe the trees and lists used to store and access grid information in one and two space dimensions and to manage the one large array where the solution vector for each grid is stored. The more unusual way that data structures play a role is in the grid generation algorithms. Given a list of flagged grid points, any one of us can take a pen and draw in good

grids by eye. It is not so easy to get the computer to do it. We have been able to draw on some ideas from the pattern recognition literature in designing our grid generation package. However, most of these scene analysis programs make many passes over the data from one scene. Since for a time-dependent pde our grid generation method will be used every few timesteps, we need a faster algorithm. In § 3, we describe our two-part grid generation algorithm and the graph structures on which the algorithm is based. This is a more important and difficult problem, and we believe that more work will yield methods substantially better than the preliminary heuristics discussed here.

2. Data structures for grid management. In this section we describe the trees and linked lists used to store and access information for a grid structure. The one-dimensional structure is described first; a generalization of it is used in the two-dimensional case. We assume the reader is already familiar with linked lists and the usual ways to implement trees. A good reference for this is [1].

In one space dimension, the data structure we use to keep track of the grids is a tree structure. Each grid in the grid hierarchy corresponds to a node in the tree. When a fine grid is nested in a coarse grid, we say the corresponding node in the tree is an offspring of the parent node corresponding to the coarse grid. Two subgrids in the same coarse grid are said to be siblings. Fig. 3a shows a grid structure with one coarse grid, three fine grids at level 1, and three fine grids at level 2. The corresponding tree structure which details the relationships between the grids is shown in Fig. 3b. The only nonstandard links in this tree are indicated by the dashed lines. These additional pointers make the operation of finding all grids at a given level easy.

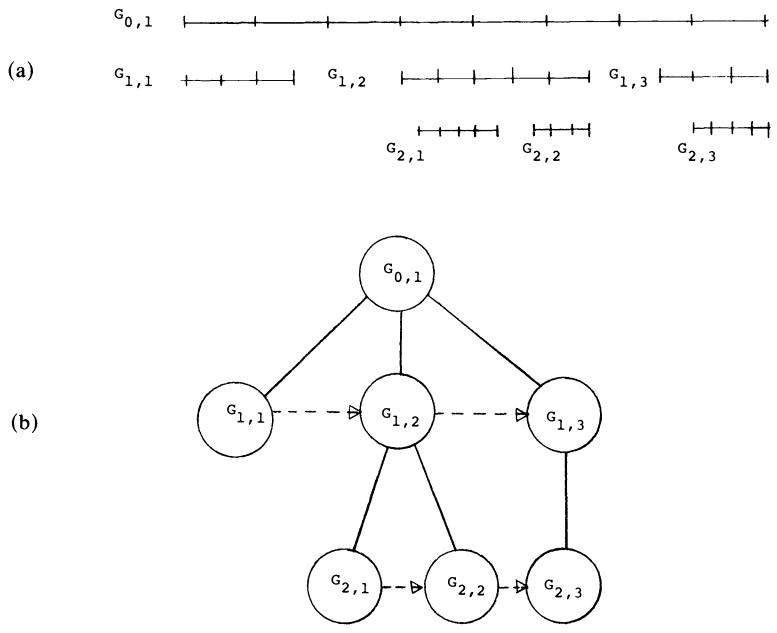


FIG. 3. *A one-dimensional grid structure and its associated tree data structure.*

To implement the tree, we would like each node to have a fixed number of items of information describing the grid. Since a grid can have an arbitrary number of subgrids, the tree is implemented by having each node point only to its first offspring, and the offspring points to its next sibling.

The information that is stored in a node for each one-dimensional grid is:

1) grid location,
2) Δx, Δt, t,
3) number of rows and columns,
4) parent grid pointer,
5) first subgrid pointer,
6) sibling pointer,
7) solution storage pointer.

Even in the unlikely event of a calculation having 50 fine grids, with 15 pieces of information per grid, this is only 750 words of storage. This is a very small amount of storage overhead compared to the amount of storage needed for the solution itself. By using locally uniform fine grids, we save the storage overhead found in irregular mesh refinement approaches, which is typically proportional to the number of grid points. (For a discussion of the CPU overhead in this approach to adaptive mesh refinement see [3].)

We emphasize two things about the tree data structure. Most integration algorithms (in particular, in [3]) have an information flow which follows path links in the tree, so there is no processing time overhead associated with them. For example, the fine grid typically gets boundary values from an underlying coarse grid. This information is directly available from the fine grid node, without having to traverse the entire tree. The second point about the data structure is that it needs to be dynamic in an unusual way for trees. Fine grids will be created and/or removed as needed during a calculation. This will occur more frequently for the finest level grids than the coarser grids. The data structure is changed by creating a new bottom half of the tree and joining it with the old top half of the tree. The nodes from the old bottom half are added to a list of free nodes.

There are several complications in the two-dimensional version of a grid structure that lead us to generalize the tree structure. First is the possibility of more than one coarse grid, and thus, more than one root node of the tree. Secondly, a grid may be nested in more than one coarse grid and thus have several parents in the tree. The parent slot can be replaced by a pointer to a short linked list of parent grids. Third, in two dimensions, grids at the same level of refinement may overlap, and so we add a pointer to a list of intersecting grids to the information which is stored for each node. In addition, for rotated grids we also store $\sin(\theta)$, $\cos(\theta)$, where θ is the angle of rotation of the grid. Fig. 4a shows a sample two dimensional grid structure, with two grids at each of the three levels; Fig. 4b shows the corresponding data structure. Conceptually, it can still be thought of as a tree.

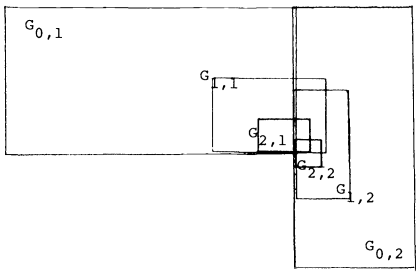
At this point we discuss some of the overhead associated with the use of rotated rectangular subgrids. In any operation between grids that are rotated with respect to each other, each indexing operation will involve an extra calculation. Suppose we need the coordinates of point (i, j), for example to interpolate an initial solution value for a point in a newly created fine grid. If the grid is not rotated, the typical calculation looks like

$$\begin{pmatrix} x_{ij} \\ y_{ij} \end{pmatrix} = \begin{pmatrix} c_x + (i-1)\,\Delta x \\ c_y + (j-1)\,\Delta y \end{pmatrix},$$

where (c_x, c_y) are the coordinates of the lower left hand corner of the fine rectangle. In the rotated case, the calculation looks like

$$\begin{pmatrix} x_{ij} \\ y_{ij} \end{pmatrix} = \begin{pmatrix} c_x + \cos(\theta)(i-1)\,\Delta x - \sin(\theta)(j-1)\,\Delta y \\ c_y + \sin(\theta)(i-1)\,\Delta x + \cos(\theta)(j-1)\,\Delta y \end{pmatrix}.$$

(a)



(b)

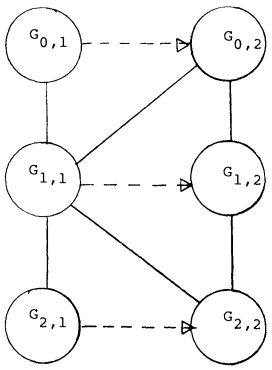


FIG. 4. *A two-dimensional grid structure and its associated graph.*

This amount of overhead is not bad. The mathematically difficult problem concerns the boundary conditions for a fine grid. In cases where the solution to a pde is discontinuous, the numerical method should be conservative. This rules out straightforward bilinear interpolation, and requires much more difficult interface equations (see [4] for a discussion of this problem). In such cases we use nonrotated grids, since the interface between grids then degenerates to a one-dimensional interface which is much more mathematically tractable. However, this is an active area of research today, and we do not believe that final conclusions can be drawn now.

The final data structure in our problem allocates storage for the solution on each grid from one long solution storage array. This array is managed by keeping a list of free chunks of storage, sorted by their location in the array. When a grid is created, its request for a certain number of words of storage is satisfied by taking contiguous storage from the first free block which is big enough. This is a first-fit algorithm. Such algorithms have been shown [13] to be preferable in most cases to best-fit algorithms, in which storage is allocated from the block that is closest in size to the requested amount.

3. Algorithms to generate the grid structure. In this section we present the algorithms used to generate the fine grids. We describe the algorithms in two dimensions, but they generalize immediately to higher dimensions. The algorithm starts with a rectangular coarse grid. Based on estimates of the error, grid points are flagged as needing to be in a grid with finer mesh width. The problem is this: given a list of flagged grid points, how should (rotated) rectangular subgrids be created to minimize the total area of the refined grids, so that each flagged point is interior to a fine grid

unless it is on the boundary of the physical domain. Since the work of integrating the solution on a fine grid is proportional to its area, it is clear that we would like to minimize the area of the coarse grid which is unnecessarily refined.

This is a difficult problem, and it is difficult to find a foolproof algorithm which works in all cases. Our approach is to use a simple grid generation first and then evaluate the resulting grids to try to detect when it fails. We then use a safer but more expensive grid generation algorithm. The complete procedure consists of:

1) a clustering algorithm to decide which flagged points go together in one fine grid;
2) a grid fitting algorithm, which fits a rotated rectangle to each cluster;
3) an evaluation step, which detects the failure of the simple clustering algorithm by measuring the area of the proposed fine grid which is unnecessarily refined. We would also like as few grids created as possible, since there is computational overhead associated with the integration of the boundary of each fine grid.

We describe the first pass through the clustering procedure and then the alternate algorithms used in the difficult cases. Using either clustering algorithm, the fitting of the rectangle is the same. In the nonrotated case, the new grid is defined by finding the dimensions of the rectangle enclosing the cluster. The procedure for finding the orientation for rotated subgrids is discussed at the end of this section. (For time dependent adaptive calculations, the grids are then enlarged to include a buffer zone of a few mesh widths, to lengthen the interval between regridding operations. How this buffer zone is added can affect the amount of overlap between the grids.)

The clustering algorithm serves two purposes. The first is to separate the flagged points which come from spatially separated phenomena (see Fig. 5a). This is simple to do using a nearest neighbor algorithm. Start with one flagged point in a cluster. Add flagged points if the distance between the point and the point nearest to it in the cluster is small enough, typically two mesh widths or less. Since the flagged points come from a regularly ordered grid, this algorithm runs in time approximately linear in the number of flagged points n, rather than the worst case $O(n^2)$.

The second purpose of the clustering algorithm is to break up one nearest neighbor cluster if it leads to an inefficient grid. In Fig. 5b, if the entire cluster were fit with one grid (the dotted rectangle) an unacceptably large area would be refined. Instead, the cluster should be divided in two. Step 3, the evaluation step, would detect this using the simple approximation of taking the ratio of the number of flagged grid points to the total number of coarse grid points in the new fine grid. If the ratio falls below a cutoff, typically between $\frac{1}{2}$ and $\frac{3}{4}$, the points must be reclustered.

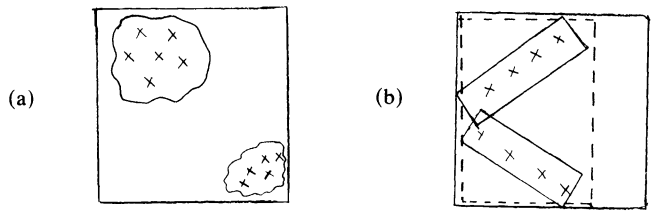


FIG. 5. *The clustering algorithms divide up the flagged grid points that should go together to make a new fine grid.*

The nearest neighbor clustering works well when entire regions of the domain need to be refined, and each such region is separated from the next. This occurs in transonic flow, for example, where the leading and trailing edges of the airfoil typically need refinement. If nearest neighbor clustering does not work, the assumption is that

the points needing refinement lie along either a long curved front (as in Fig. 2), or several intersecting fronts (as Fig. 1 indicates). Since there may be a lot of scatter in the flagged points, it is difficult to sense their direction or distribution. For this reason, we next use data structures to organize the flagged points to understand how they are related, so that a smart subdivision of the points can be made.

We first connect the flagged points into a minimal spanning tree (MST). A MST is the connected acyclic graph connecting all the points so that the sum of the lengths of the edges is a minimum [1]. In this graph, each flagged point is connected to its nearest neighbor. The hard part now is deciding how to break the graph into different subclusters. An iterative algorithm immediately comes to mind. Start with a cluster of one point at the end of the MST. Add neighboring flagged points into the cluster if the resulting grid is still acceptable. Since the grid fitting part of the algorithm is easy (discussed next), this algorithm is feasible even though it is iterative. Fig. 6 shows a sample set of points, their MST and the three rectangles this iterative algorithm produces, starting at the left.

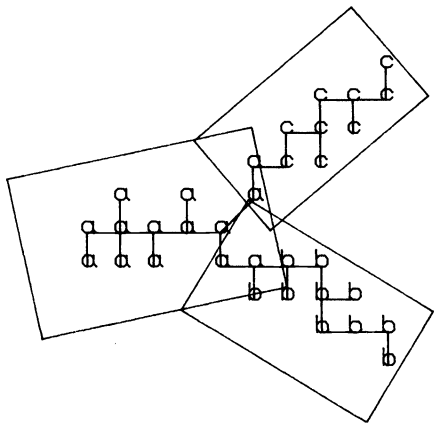


FIG. 6. *Three subgrids are generated from the* MST *connecting the flagged points.*

In practise, there are two changes made in the above algorithm. First it is less expensive to start with several points in a cluster rather than one. We take the original cluster and (inefficient) rectangle formed by the nearest neighbor algorithm, and repeatedly bisect it in the long direction until the component grids are acceptably efficient. The components are then put back together (if possible) in the iterative merging step based on the MST. Fig. 7a illustrates a "worst case" example which still works well. The flagged points make a cross pattern. Bisection makes the least obvious cuts, yielding 8 clusters. However, in Fig. 7b the merging step puts them back together nicely.

The second change we make in the algorithm as stated comes from the fact that the MST is not unique. Figs. 8a and 8b give two MST's for the same point set. The problem in Fig. 8 is the long path length between neighboring points. Since our flagged points come from a regular grid structure, this problem can occur frequently. We therefore generalize the MST by connecting a point to All its Nearest Neighbors, to make an ANN graph.

In constructing the MST or ANN connecting clusters of points, a little care must be taken in the definition of the location of a cluster of points. For example, in Fig. 9 the mean of the points would suggest that rectangles A and B be connected, but

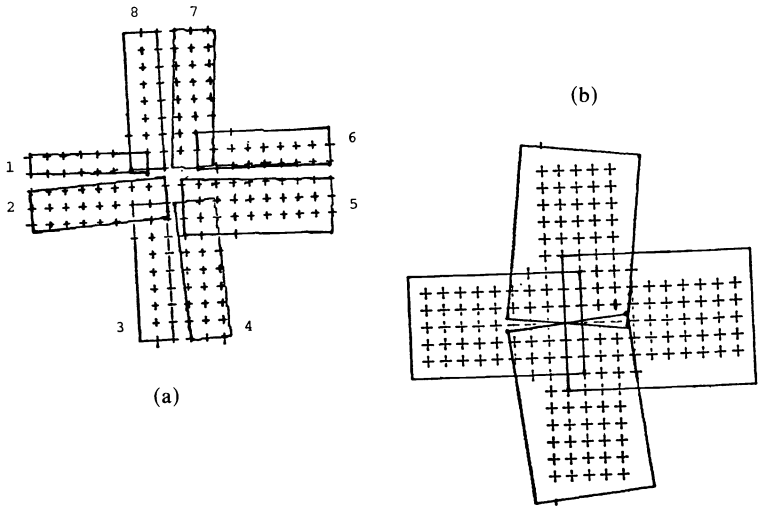


FIG. 7. *Bisection of the originally proposed rectangle yields* 8 *clusters. The merge step produces* 4 *fine grids.*

FIG. 8. *Two different* MST's *for the same point set.*

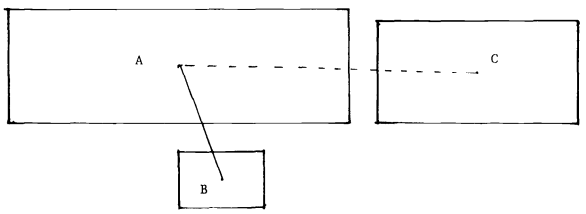


FIG. 9. *Rectangles A and C should merge, even though rectangles A and B are closer.*

clearly it is better to merge A and C. Other measures, such as corner separation, must be used. There are other graphs that provide useful structures with which to think about these problems. For example, clusters AB and AC in Fig. 9 would both be connected in a Relative Neighbor Graph [17], where two clusters are connected if no other cluster is closer to them both. We do not actually construct an RNG however, and admittedly, only approximate MST and ANN graphs are really needed in this work.

Returning to the grid generation algorithm, we now discuss how to generate a rotated rectangle to enclose all flagged points in one cluster. In addition, if flagged points have an orientation, the subgrid should have that orientation too. Let x_i, y_i, $1 \leqq i \leqq n$ be the coordinates of the flagged points, and $\bar{x}$, $\bar{y}$ their mean. Consider the symmetric matrix

$$M^t M = \begin{pmatrix} \sum_i x_i^2 - \bar{x}^2 & \sum_i x_i y_i - \overline{xy} \\ \sum_i x_i y_i - \overline{xy} & \sum_i y_i^2 - \bar{y}^2 \end{pmatrix},$$

where

$$M = \begin{pmatrix} | & | \\ x_i - \bar{x} & y_i - \bar{y} \\ | & | \end{pmatrix}.$$

This matrix determines an ellipse with the same first and second moments as the flagged points [9]. The axes of the ellipse are the eigenvectors of the 2 by 2 matrix M^tM, which we use to determine the orientation of the rectangle. The most expensive parts of this algorithm is the determination of the dimensions of the rectangle, given its orientation, so that all flagged points are included.

It turns out that this algorithm is related to a total least squares fit [11] of the data points in the following way. Suppose we look for a linear fit to the flagged points by a line through the mean,

$$(y - \bar{y}) = m(x - \bar{x}),$$

so that we have

$$\left\{ \begin{pmatrix} x_1 - \bar{x} \\ x_2 - \bar{x} \\ \vdots \\ x_n - \bar{x} \end{pmatrix} + \begin{pmatrix} e_1 \\ e_2 \\ \vdots \\ e_n \end{pmatrix} \right\} m = \begin{pmatrix} y_1 - \bar{y} \\ y_2 - \bar{y} \\ \vdots \\ y_n - \bar{y} \end{pmatrix} + \begin{pmatrix} r_1 \\ r_2 \\ \vdots \\ r_n \end{pmatrix}. \tag{3.1}$$

The slope of the line m is determined so that the Frobenius norm of the perturbation vectors $e = (e_1, e_2, \cdots, e_n)^t$ and $r = (r_1, r_2, \cdots, r_n)^t$ is a minimum. Rewrite (3.1) as

$$([x - \bar{x} | y - \bar{y}] + [e|r]) \begin{bmatrix} m \\ -1 \end{bmatrix} = 0.$$

For the smallest perturbation, take the perturbation matrix $C = [e|r]$ to be the smallest singular value of the matrix $[x - \bar{x}|y - \bar{y}]$. This matrix is just the matrix M of the flagged points. The solution vector $(m, -1)^t$ is the singular vector corresponding to the smallest singular value. This singular vector of M is one of the eigenvectors of the matrix M^tM above, and so both derivations give the same rectangle orientation. By using the total least squares derivation, we see that we are finding the slope of the line through the mean of the points which minimizes the sum of the squares of the distances from each point to the line.

One of the goals in grid generation is to create subgrids with total area as small as possible. In two dimensions it is computationally feasible to construct the minimum area rectangle enclosing a set of points. We will briefly describe how to do this and compare it with our procedure above. To construct a spanning rectangle, first find the convex hull of the set of points. In two dimensions, this may be done in $O(n \log n)$ operations, where n is the number of flagged points to be enclosed [12]. The next step makes use of a theorem by Freeman and Shapira [10] proving that the minimum area rectangle has a side collinear with the convex hull. It remains to check each side of the hull for the rectangle with the minimum area. For each line segment on the hull, the area of the spanning rectangle may be determined in time $O(h)$, where h is the number of vertices on the boundary of the convex hull, by carefully figuring which vertices on the boundary anchor the rectangle. Finally, choose the rectangle with the minimum area.

This algorithm does not generalize easily to higher dimensions, where the convex hull takes $O(n^2)$ operations to compute. Even in two dimensions, it is much more

expensive than using the ellipse method. Since the minimum spanning rectangle depends only on the convex hull of the set of points, it will be aligned with the points only if the clustering algorithm aligns the points first. In experiments comparing the minimum area spanning rectangle with moment generated rectangles, the latter typically has only 5 to 10% larger area than the former.

This performance of the ellipse algorithm has been heuristically explained in [2] in the following way. Suppose the sides of the rectangle are oriented in the directions of the orthonormal unit vectors r_1, r_2. The length of the side parallel to r_1 is $\max Ar_1 - \min Ar_1$, where A is the n by 2 matrix of the flagged points,

$$A = \begin{pmatrix} x_1 & y_1 \\ \vdots & \vdots \\ x_n & y_n \end{pmatrix}$$

and similarly for r_2. The max and min are taken componentwise over the vector. If the points are clustered so that there are no extreme outlying points, (which we certainly expect the clustering algorithm to accomplish), the length of a side is

$$\max Ar_1 - \min r_1 \approx 2\|Ar_1\|_\infty.$$

Thus we consider the maximum norm problem of minimizing the area of the enclosing rectangle,

$$\min_{\substack{r_1, r_2 \\ r_i r_j = \delta_{ij}}} \|Ar_1\|_\infty \|Ar_2\|_\infty. \tag{3.2}$$

If (3.2) is approximated using the 2-norm, this problem can be easily solved using the fact that

$$\|Ar_1\|_2^2 \|Ar_2\|_2^2 = r_1^t A^t A r_1 r_2^t A^t A r_2,$$

but for the orthonormal vectors $r_1 r_2$,

$$r_1^t A^t A r_1 + r_2^t A^t A r_2 = \text{constant}.$$

The problem again becomes that of minimizing $\|Ar\|_2$ over unit vectors r, giving the same orientation vectors that the moment generated ellipse gives. The area of the enclosing rectangle is related to the area of the minimum enclosing rectangle as the 2-norm is related to the maximum norm. In general there can be a large difference between those two norms. However, if we use a smart clustering algorithm, this will not be the case.

We end with examples of the grid generation package on several point sets which illustrate the effect of the efficiency parameter in the final step. In Fig. 10a, approximately 85% of the points in each of the six fine grids are flagged. In Fig. 10b only 45% were required to be flagged, and so larger grids were created. Figs. 11a and 11b illustrate a similar phenomenon on a different set of points.

There is still much more that could be done to come up with a fast, reliable grid generation package. For example, in Fig. 11a, the higher-efficiency-rated grids do not have a smaller total area refined, since the fine grids overlap more than in Fig. 11b. Other measures for evaluating grids might thus be beneficial. There are also alternative grid generation algorithms which seem reasonable but have not been tested. For example, based on the MST one could use the diameter of the graph to indicate the layout of the points. In problems with a discontinuous solution, if we assume that the flagged points follow the front, the diameter should approximate the shape of the front. Clusters can now be formed by segmenting the diameter. This is similar to an

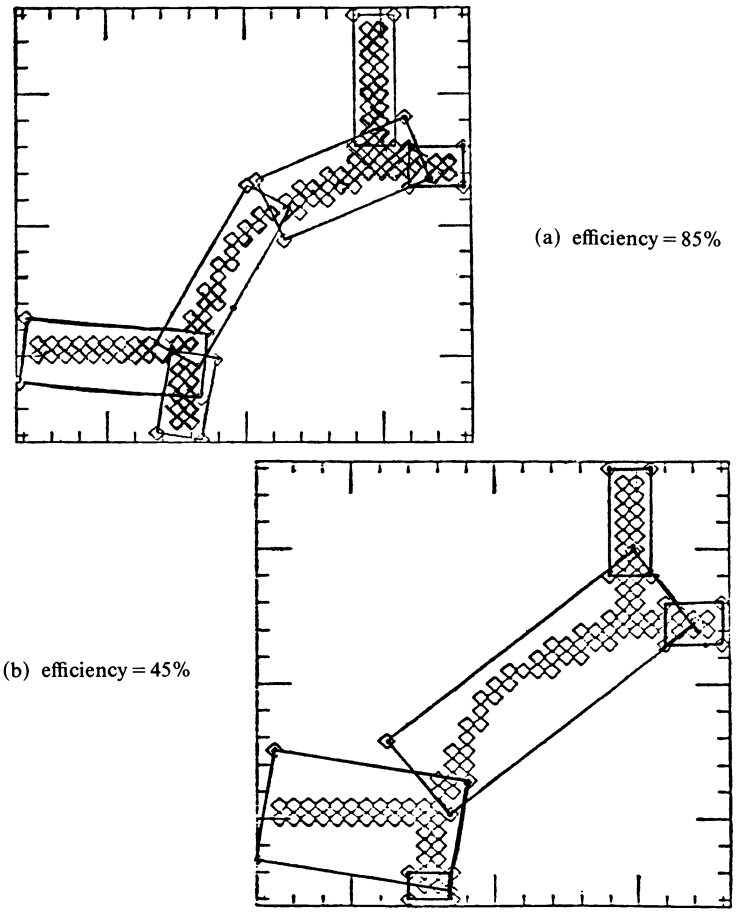


FIG. 10. (a) 85% *of the coarse grid points in the fine grid are flagged*; (b) *only* 45% *are flagged.*

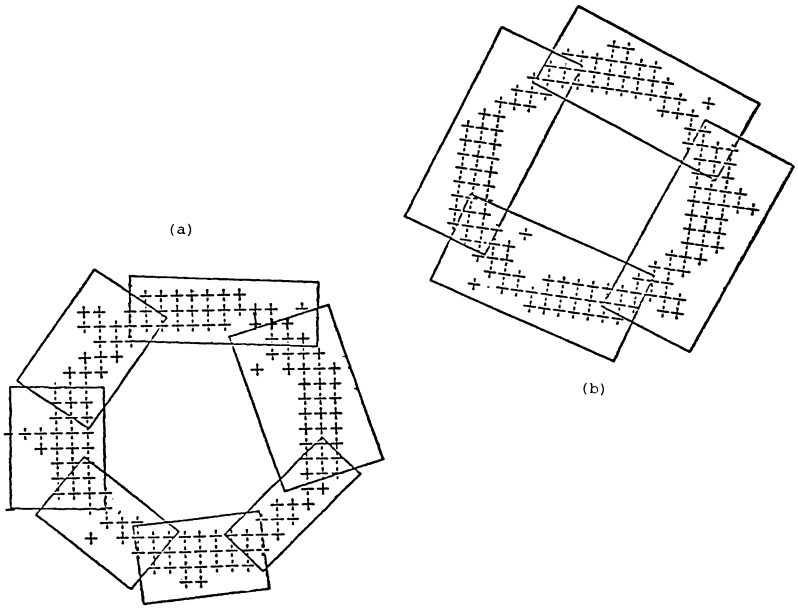


FIG. 11. (a) 75% *of the coarse grid points in the fine grids are flagged*; (b) *only* 50% *are flagged.*

algorithm in [14] which was used to recognize curved objects in the plane. Although the algorithms presented here are still experimental and under development, they have already been incorporated into adaptive mesh refinement programs for the solution of pdes. Although not optimal, they have been proven successful and efficient in the automatic generation of adaptive subgrids.

Acknowledgments. I would like to thank Doug Baxter for his participation in the development of the grid generation algorithms. I also thank William Gropp for his participation in the early stages of the development of the mesh refinement program.

REFERENCES

[1] A. AHO, J. HOPCROFT AND J. ULLMAN, *The Design and Analysis of Computer Algorithms*, Addison-Wesley, Reading, MA, 1974.

[2] D. BAXTER, M.S. thesis, Dept. Computer Science, Stanford Univ., Stanford, CA, 1981.

[3] M. J. BERGER AND J. OLIGER, *Adaptive mesh refinement for hyperbolic partial differential equations*, J. Comp. Phys., 53 (1984), pp. 484–512.

[4] M. J. BERGER, *On conservation at grid interfaces*, ICASE Report No. 84-43, NASA, Langley, VA, September, 1984.

[5] M. J. BERGER AND A. JAMESON, *Automatic adaptive grid refinement for the Euler equations*, AIAA J., 23 (1985), pp. 561–568.

[6] ——— *An adaptive multigrid method for the Euler equations*, in Lecture Notes in Physics 218, Soubbaramayer and J. P. Boujot, eds., Springer-Verlag, Berlin, 1985.

[7] J. BOLSTAD, Ph.D. thesis, Dept. Computer Science, Stanford Univ., Stanford, CA, 1982.

[8] S. CARUSO, Ph.D. thesis, Dept. Mechanical Engineering, Stanford Univ., Stanford, CA, in preparation.

[9] H. CRAMER, *Mathematical Methods of Statistics*, Princeton Univ. Press, Princeton, NJ, 1951.

[10] H. FREEMAN AND R. SHAPIRA, *Determining the minimum-area encasing rectangle for an arbitrary closed curve*, Comm. ACM, 18 (1975), pp. 409–413.

[11] G. H. GOLUB AND C. VAN LOAN, *An analysis of the total least squares problems*, SIAM J. Numer. Anal., 17 (1980), pp. 883–893.

[12] R. GRAHAM, *An efficient algorithm for determining the convex hull of a finite planar set*, Inform. Proc. Letters, 1 (1972), pp. 132–133.

[13] D. KNUTH, *The Art of Computer Programming*, Vol. 1, Addison-Wesley, Reading, MA, 1973.

[14] R. NEVATIA AND T. BINFORD, *Description and recognition of curved objects*, Artif. Intell., 8 (1977), pp. 77–98.

[15] J. SALTZMAN AND J. BRACKBILL, *Applications and generalizations of variational methods for generating adaptive meshes*, in Numerical Grid Generation, J. Thompson, ed., North-Holland, Amsterdam, 1982.

[16] K. STUBEN AND U. TROTTENBERG, *Multigrid method: fundamental algorithms, model problem analysis and applications*, in Multigrid Methods, W. Hackbusch and U. Trottenberg, eds., Springer-Verlag, Berlin, 1982.

[17] G. TOUSSAINT, *The relative neighborhood graph of a finite planar set*, Pattern Recognition, 12 (1980), pp. 261–268.

SIAM J. SCI. STAT. COMPUT.
Vol. 7, No. 3, July 1986

015

THE APPLICATION OF CELL DISCRETIZATION TO A "CIRCLE IN THE SQUARE" MODEL PROBLEM*

MARK COFFEY†, JOHN GREENSTADT‡ AND ALAN KARP†

Abstract. The Cell Discretization (CD) algorithm is applied to several elliptic p.d.e.'s with Dirichlet boundary conditions. To demonstrate the generality of CD, we use a "Circle in the Square" geometry—a square domain with an imbedded circle of arbitrary location and size. By this choice, we can illustrate the ability of CD to incorporate arbitrarily shaped interfaces without approximation, and coordinate systems and basis sets that may vary from cell to cell.

We discuss the solution of three problems: (1) Laplace's equation with the boundary values of a known harmonic function, (2) the calculation of the Green's function for the square and (3) Poisson's equation with discontinuous diffusion coefficient and source term. Each problem illustrates an important feature of CD. The results are compared either with analytic or fine-mesh finite difference solutions. The convergence properties of CD are demonstrated by these comparisons.

We conclude that Cell Discretization is a convenient, accurate method for solving elliptic partial differential equations even when unusual domain shapes or solution representations are involved.

Key words. cell discretization, elliptic partial differential equations, moment collocation

AMS(MOS) subject classifications. 65N30, 65N35

1. Introduction.

In past applications of the Cell Discretization method for solving elliptic partial differential equations [1], [2], several very large problems were attempted and solved; the method proved to be robust and efficient. Because of the necessity to simplify the logistics for programming purposes, only rectangular domains were treated. The potential of the CD method for handling quite general geometries has never been exploited.

We have now had the chance to test a small model problem with a geometrical structure which is quite different from these purely rectangular ones. It is a simple, two-domain problem, consisting of a circle in a square, as shown in Fig. 1. The plane is partitioned into three "cells"; the first is Ω_0, the exterior of the square, the second is Ω_1, the square itself minus a circular hole (making it a nonsimply connected domain), and the third is Ω_2, the circular disc with center at (x_0, y_0) and radius r_0.

As usual, the boundary of the square is regarded as the interface Γ_{10} between Ω_1 and the exterior cell Ω_0. Strictly speaking, Γ_{10} is a single "interface," but for purposes of better approximation, it is advantageous to consider it as consisting of four segments (the sides), which we label $\{\Gamma_{10\sigma}\}$, with $\sigma = 1, 2, 3, 4$. The interface consisting of the circle itself is denoted by Γ_{12} and is made up of only one segment. Although we shall, of course, be seeking only an approximate solution to each specific problem within this geometrical configuration, it is important to note that we are treating the geometrical entities *exactly*, i.e., we are not approximating the circle by a polygon, etc.

Within each cell, a simple elliptic, inhomogeneous p.d.e. is to be solved with a Dirichlet condition on the boundary of the square. Each such equation has the form:

$$-\nabla \cdot (a_k \nabla \Psi) + b_k \Psi = c_k \tag{1.1}$$

with $k = 1, 2$ (all coefficients and solutions are assumed to vanish identically in Ω_0). The quantities a, b and c are assumed to be functions of the coordinates with whatever

* Received by the editors January 16, 1984, and in revised form June 4, 1985.

† IBM Corporation, Palo Alto Scientific Center, Palo Alto, California 94304.

‡ Department of Applied Mathematics and Theoretical Physics, University of Cambridge, Cambridge, England CB3 9EW.

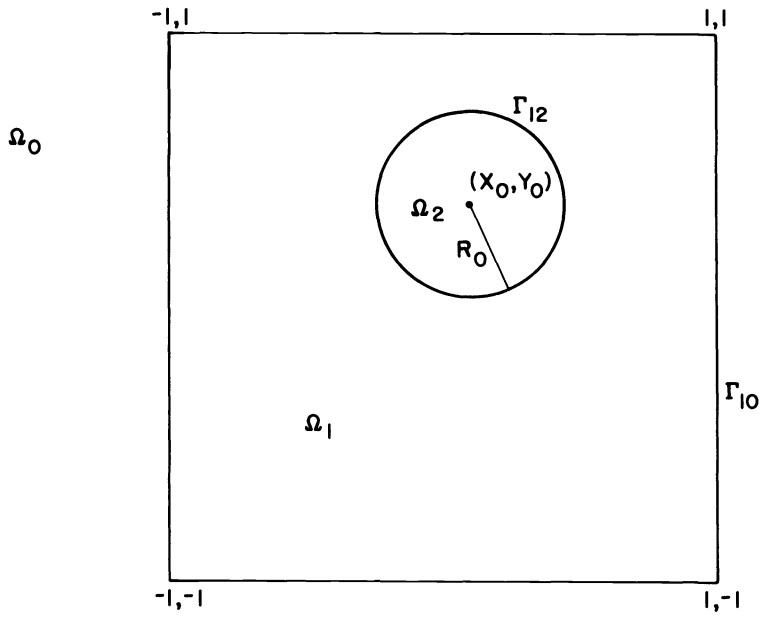


FIG. 1. *Geometry of the Circle in the Square model problem. The square domain Ω is partitioned by means of a circle of radius r_0 and center at (x_0, y_0). The boundary of Ω is denoted by Γ_{10}. The circular interface is denoted by Γ_{12} and the disc which it encloses by Ω_2. The cell $\Omega - \Omega_2$ is denoted by Ω_1.*

continuity properties are necessary, and Ψ is the unknown solution function which we wish to approximate with the CD procedure.

For conciseness of notation, when the point whose coordinates are (x, y) lies in the interior of one of the cells Ω_k, we shall label this point simply with the letter x; but when the point lies on one of the interfaces between contiguous cells, we shall label it with the letter s. Thus, the boundary conditions can be written concisely as:

$$\Psi(s) = f(s); \qquad s \in \Gamma_{10} \tag{1.2}$$

where $f(s)$ is a given function defined on the boundary.

There is also a corresponding continuity condition across Γ_{12} which, for the classical problem is understood implicitly, but which must be considered explicitly when the cell discretization is brought into play. This condition may be expressed conveniently if we define:

$$\begin{aligned} \Psi_1(x) &\equiv \Psi(x), \qquad x \in \Omega_1, \\ \Psi_2(x) &\equiv \Psi(x), \qquad x \in \Omega_2 \end{aligned} \tag{1.3}$$

in which case, the interface continuity condition across Γ_{12} can be written concisely as:

$$\Psi_1(s) = \Psi_2(s), \qquad s \in \Gamma_{12}. \tag{1.4}$$

The sections that follow describe the application of the Cell Discretization method to the problem just described. In § 2, we outline the basic features of the algorithm. In § 3, the particular choices of the intracell basis functions and the interface weight functions characterizing the discretization are described. In § 4, we describe how the *S*, *U*, *T* and *W* arrays, whose elements are integrals of various combinations of these functions and their derivatives, are evaluated. In § 5, detailed descriptions of our test problems are given, as well as the numerical results for them, mostly in the form of graphs. Finally, in § 6, these results and their implications are discussed.

2. Summary of the cell discretization algorithm. Although Fig. 1 shows the case we are considering here of two cells only, it is no more complicated to describe the CD method for the more general case of K cells, so we shall do so. In what follows, the labels 0, 1, 2, k, m, etc. shall mostly be written as if they were the arguments of a function, rather than being written as subscripts. We shall maintain this convention for quantities associated with each cell or interface. Although this makes for somewhat odd-looking expressions at times, it reduces the number of subscripts, and makes the equations easier to read.

The CD method is variationally-based, so we must start with the "bare" (i.e., without interface constraints) functional:

$$\Phi_0 \equiv \tfrac{1}{2}\int_\Omega (a(\nabla\Psi)^2 + b\Psi^2 - 2c\Psi)\, d\Omega. \tag{2.1}$$

We next partition Ω into K subdomains $\{\Omega_k\colon k = 1, \cdots, K\}$ and we call each subdomain a cell. Within each cell, we approximate Ψ by a function $\psi(x, \theta; k)$ of predetermined form, but with a set of parameters $\{\theta_\mu(k); \mu = 1, \cdots, M(k)\}$ with values as yet unknown. For conciseness, we shall consider this set of θ's as forming a vector $\theta(k)$. When we replace the unknown function $\Psi(x)$ by its system of approximations (or representations) $\{\psi(k)\}$, the functional Φ_0 has the value:

$$\Phi_0 \equiv \tfrac{1}{2}\sum_k \int_{\Omega(k)} (a(k)[\nabla\psi(k)]^2 + b(k)[\psi(k)]^2 - 2c(k)\psi(k))\, d\Omega_k. \tag{2.2}$$

In partitioning Ω, we have detached the various intracell approximations $\{\psi(x, \theta; k)\}$ of the single function Ψ from one another. It is therefore necessary to "reconnect" them by some means so that, collectively, they represent Ψ in some useful way. In [1], it was shown how this is done in the CD method by *interface moment collocation*, which requires the selection of a set of *weight functions* $\{w_\alpha(s; k, m); \alpha = 1, \cdots, L(k, m)\}$ for each interface. Note that $L(k, m)$, the total number of weight functions associated with the interface Γ_{km}, may differ from one interface to the next. The *discrepancy* $\Delta(s; k, m)$, in the continuity of the representation across Γ_{km} is defined by:

$$\Delta(s; k, m) \equiv \psi(s; k) - \psi(s; m). \tag{2.3a}$$

In order to cover the case of inhomogeneous Dirichlet boundary conditions while still retaining the symmetry of the interface constraints, we introduce extra interface "source terms" into the definition of the discrepancy as follows:

$$\Delta(s; k, m) \equiv (\psi(s; k) - R(s; k, m)) - (\psi(s; m) - R(s; m, k)). \tag{2.3b}$$

These will be used later to specify boundary values; for an interior interface, the R's are usually set to zero. Note again that the variable s stands for the variable x when it is evaluated on an interface. It is also important to take note at this time of certain symmetries in the indices k and m, which label contiguous cells. Obviously, $\Delta(k, m)$, by its definition (2.3), is antisymmetric in k and m and, also by definition, the labeling of the w's and the Γ's makes them symmetric in k and m. We therefore have:

$$\begin{aligned} \Delta(s; m, k) &\equiv -\Delta(s; k, m), \\ w(s; m, k) &\equiv w(s; k, m), \\ d\Gamma_{mk} &\equiv d\Gamma_{km} \end{aligned} \tag{2.4}$$

the last relation holding because we ignore the orientation of $d\Gamma$.

Clearly, it is not possible, in general, to make $\Delta(s; k, m)$ vanish identically over the whole of Γ_{km} as in (1.4), because Δ, being the difference between two approximate representations, is fixed in functional form and contains only a finite number of degrees of freedom. These are contained in the vectors $\theta(k)$ and $\theta(m)$, which together have $M(k)+M(m)$ elements. Therefore, it is necessary to *weaken* the continuity requirement by asking only that the $L(k, m)$ *moments* of $\Delta(k, m)$ (equal in number to the weights defined on Γ_{km}), vanish on Γ_{km}. We therefore require:

$$\int_{\Gamma(k,m)} \Delta(s; k, m) w_\alpha(s; k, m)\, d\Gamma_{km} = 0 \tag{2.5}$$

for $\alpha = 1, \cdots, L(k, m)$. (In the future, for the sake of conciseness, we shall omit the cell labels attached to integral signs, and rely on the labeling of the elements of integration to carry this information). Note that when one of the labels in (2.5) is zero, it represents a *boundary* moment collocation.

There is more than one way to incorporate the interface constraints (2.5) into the variational formulation. For large problems, the solution of the discrete equations is rendered much more efficient if the constraints are turned into identities by a *pretransformation* of the θ's, as was done in (1) and (2). For our purposes here, however, this rather elaborate technique is not necessary, and would perhaps obscure the basic simplicity of the calculation. We shall incorporate the interface constraints in the simplest possible way, viz., by the use of a set of Lagrange multipliers $\{\lambda_\alpha(k, m)\}$. We alter Φ_0 by adding the appropriate terms to form the composite Lagrangian function Φ for the problem, which is:

$$\Phi \equiv \Phi_0 + \sum_{km} \sum_{\alpha} \lambda_\alpha(k, m) \int_\Gamma \Delta(s; k, m) w_\alpha(s; k, m)\, d\Gamma_{km}. \tag{2.6}$$

Because of the symmetry relations (2.4), the integral over Γ is antisymmetric in k and m; therefore only a set of λ's which are antisymmetric in those indices will contribute to the summation. When, for simplicity, we assume $\{\lambda(k, m)\}$ to be antisymmetric, each nonvanishing term is repeated, with k and m interchanged. Because the sum in (2.6) is over contiguous neighbors only, k and m are always distinct; this should be borne in mind for all equations to come.

For linear equations and boundary conditions such as (1.1) and (1.2), the most sensible choice for the form of $\psi(x, \theta; k)$ is a linear combination of predetermined basis functions $\{\phi_\mu(x; k)\}$ with the θ's as coefficients:

$$\psi(x, \theta; k) \equiv \sum_{\mu} \theta_\mu(k) \phi_\mu(s; k) \tag{2.7}$$

with $\mu = 1, \cdots, M(k)$. When this expression is substituted into (2.6), we obtain,

$$\begin{aligned} \Phi = \sum_k \left(\tfrac{1}{2} \sum\sum_{\mu\nu} \theta_\mu(k) S_{\mu\nu}(k) \theta_\nu(k) - \sum_\mu \theta_\mu(k) T_\mu(k) \right) + \sum_{km} \sum_\alpha \lambda_\alpha(k, m) \\ \times \left(\sum_\mu U_{\mu\alpha}(k, m) \theta_\mu(k) - \sum_\nu U_{\nu\alpha}(m, k) \theta_\nu(m) - W_\alpha(k, m) \right) \end{aligned} \tag{2.8}$$

where S, T, U and W are defined below. The various index ranges in these summations can be "adjusted" to include the exterior cell with label 0, if we bear in mind that all quantities in Ω_0 vanish. It is really not necessary to extend the index range in the cellwise summation, because of this fact, but since we wish to deal with the boundary segments as if they were interfaces, we can do this most conveniently by assuming that

$k = 0, \cdots, K$ and $m = 0, \cdots, K$. (There are still K *interior* cells in the problem.) However, k and m are still restricted by antisymmetry in the double sum as described above. Also, $\mu = 1, \cdots, M(k)$, $\nu = 1, \cdots, M(m)$ and $\alpha = 1, \cdots, L(k, m)$. We can write Φ, for conciseness, in terms of the vectors θ, λ, T, and W, and the matrices S and U. These are defined in terms of their components as follows (with the appropriate index ranges):

$$
\begin{aligned}
&\theta(k) \equiv \{\theta_\mu(k)\}, \qquad \mu = 1, \cdots, M(k),\\
&\lambda(k, m) \equiv \{\lambda_\alpha(k, m)\}, \qquad \alpha = 1, \cdots, L(k, m),\\
&S(k) \equiv \{S_{\mu\nu}(k)\}, \qquad \mu, \nu = 1, \cdots, M(k),\\
&T(k) \equiv \{T_\mu(k)\}, \qquad \mu = 1, \cdots, M(k),\\
&U(k, m) \equiv \{U_{\mu\alpha}(k, m)\}, \qquad \mu = 1, \cdots, M(k); \qquad \alpha = 1, \cdots, L(k, m)\\
&W(k, m) \equiv \{W_\alpha(k, m)\}, \qquad \alpha = 1, \cdots, L(k, m).
\end{aligned} \tag{2.9}
$$

The θ's and the λ's are the discrete unknowns of the problem, and the elements of the various coefficient matrices are given by:

$$
S_{\mu\nu}(k) \equiv \int_\Omega \{a(k)[\nabla\phi_\mu(k) \cdot \nabla\phi_\nu(k)] + b(k)\phi_\mu(k)\phi_\nu(k)\}\, d\Omega_k, \tag{2.10a}
$$

$$
T_\mu(k) \equiv \int_\Omega c(k)\phi_\mu(k)\, d\Omega_k, \tag{2.10b}
$$

$$
U_{\mu\alpha}(k, m) \equiv \int_\Gamma \phi_\mu(s; k) w_\alpha(s; k, m)\, d\Gamma_{km} \tag{2.11a}
$$

and

$$
W_\alpha(k, m) \equiv \int_\Gamma [R(s; k, m) - R(s; m, k)] \times w_\alpha(s: k, m)\, d\Gamma_{km}. \tag{2.11b}
$$

In line with our earlier remark (and (2.3b)), the elements of W have been written in this rather elaborate form (which departs somewhat from the definition used in [1]) to preserve its antisymmetry, but the values of R in our Circle in the Square problem are very simple, being given by:

$$
\begin{aligned}
R(s; 1, 2) = R(s; 2, 1) = R(s; 0, 1) = 0,\\
R(s; 1, 0) = f(s).
\end{aligned} \tag{2.12}
$$

In fact, the only W-vector appearing in our calculation is $W(1, 0)$, and it consists of just the moments of $f(s)$, the boundary function.

With these abbreviations, Φ can be written:

$$
\begin{aligned}
\Phi = &\sum_k \left(\tfrac{1}{2}\theta^T(k)S(k)\theta(k) - \theta^T(k)T(k)\right)\\
&+ \sum_{km} \lambda^T(k, m)(U^T(k, m)\theta(k) - U^T(m, k)\theta(m) - W(k, m))
\end{aligned} \tag{2.13}
$$

from which we will obtain the discrete equations of the problem by differentiation with respect to $\theta(k)$ and $\lambda(k, m)$. It may easily be verified by checking in the component form that the results are:

$$
S(k)\theta(k) + \sum_{m[k]} U(k, m)\lambda(k, m) = T(k) \tag{2.14}
$$

and

$$U^T(k, m)\theta(k) - U^T(m, k)\theta(m) = W(k, m). \tag{2.15}$$

The notation $m[k]$ in (2.14) indicates that the summation is over the cell labels m of *contiguous neighbors* of Ω_k. This notation follows that of [1] for the general problem.

For the very simple geometry of the problem we are now considering, it is clear that $m[1] = \{0, 2\}$ and $m[2] = \{1\}$. In fact, the discrete equations may be written out explicitly without much trouble. Specializing (2.14) gives:

$$\begin{aligned} S(1)\theta(1) + U(1,0)\lambda(1,0) + U(1,2)\lambda(1,2) &= T(1), \\ S(2)\theta(2) + U(2,1)\lambda(2,1) = S(2)\theta(2) - U(2,1)\lambda(1,2) &= T(2). \end{aligned} \tag{2.16}$$

The change of sign in the second equation is a consequence of the antisymmetry of $\lambda(k, m)$. Equation (2.15) reduces to:

$$\begin{aligned} U^T(1,0)\theta(1) &= W(1,0), \\ U^T(1,2)\theta(1) - U^T(2,1)\theta(2) &= 0. \end{aligned} \tag{2.17}$$

The missing terms result, of course, from the assumption that everything in Ω_0 vanishes.

These equations may be arranged as one matrix equation, which exhibits in a perspicuous way the structure of the equation system. It has the form:

$$\begin{bmatrix} S(1) & 0 & U(1,0) & U(1,2) \\ 0 & S(2) & 0 & -U(2,1) \\ U^T(1,0) & 0 & 0 & 0 \\ U^T(1,2) & -U^T(2,1) & 0 & 0 \end{bmatrix} \begin{bmatrix} \theta(1) \\ \theta(2) \\ \lambda(1,0) \\ \lambda(1,2) \end{bmatrix} = \begin{bmatrix} T(1) \\ T(2) \\ W(1,0) \\ 0 \end{bmatrix}. \tag{2.18}$$

In this form, the symmetry of the coefficient matrix is manifest.

The reader may note that in an $S(k)$ matrix, the gradient terms multiplied by $a(k)$ correspond to the so-called stiffness matrix in applications of the Finite Element Method (FEM). On the other hand, the integration of the term $\phi_\mu \phi_\nu$ which multiplies $b(k)$ corresponds to the mass matrix in the FEM. For the special case $b = 0$, the S matrix in the CD method would be identical with the stiffness matrix in the FEM if, instead of a piecewise polynomial representation which might be discontinuous across interfaces, a globally continuous one were used. The upper left-hand corner of the coefficient matrix in (2.18), i.e., the matrix

$$\begin{pmatrix} S(1) & 0 \\ 0 & S(2) \end{pmatrix} \tag{2.19}$$

is the analogue of the sum of the global stiffness and mass matrices in the FEM for the case of two subdomains. Furthermore, $T(k)$ is the CD analogue of the generalized force vector in the FEM, as can be seen from the definition (2.10b).

For the simple problems considered here, the system of equations (2.18) will be solved directly by elimination. The condition number of the matrix varies with the number of interface constraints on both interfaces, with the numbers of degrees of freedom $M(1)$ and $M(2)$, and with the particulars of the geometry (such as the size and location of the circle within the square).

This treatment of the discrete equations is satisfactory for a very small model problem, such as the one we are considering to illustrate the geometric flexibility of the CD method. However, for larger problems, involving many more variables than we would generate here, it is neither stable enough nor efficient enough. For large

problems, it is preferable to use the technique of pretransformation described in [1], which has the effect of removing all the λ's, all of the explicit interface constraints, and some of the degrees of freedom (the θ's) from the problem. Moreover, the resulting system of equations has a positive-definite coefficient matrix, so that various iterative methods can be used, thereby removing the necessity to use direct methods for its solution.

3. Choice of basis functions and weight functions. For Ω_1, the square with a disc punched out, the most natural coordinate system is the Cartesian one with x and y as the coordinates. In this system, the most convenient basis set consists of the monomials consisting of products of nonnegative powers, viz., $\{x^i y^j\}$. These functions are not the most desirable from the point of view of numerical stability and accuracy, but all of the integrations involving them are relatively easy to do analytically. The individual monomials are ordered in the usual way, viz., $\{1, x, y, x^2, xy, y^2, x^3, x^2y, xy^2, y^3, \cdots\}$, so that, for example, $\phi_5(x, y, 1) = xy$. This ordering may be regarded as a mapping of the powers i and j into the sequence number for the basis functions, viz.:

$$\phi_{\mu(i,j)}(x, y; 1) \equiv x^i y^j \tag{3.1a}$$

or, conversely, as a mapping of μ into i and j, viz.:

$$\phi_\mu(x, y; 1) \equiv x^{i(\mu)} y^{j(\mu)}. \tag{3.1b}$$

(We remind the reader again that the argument "1" refers to the fact that we are defining the basis functions in Ω_1.)

The weight functions on Γ_{10}, the outer boundary, are chosen for the maximum convenience consistent with reasonable approximative properties. By this, we are referring to the fact that, although Γ_{10} is a single interface, it is preferable to partition it into segments $\{\Gamma_{10\sigma};\ \sigma = 1, 2, 3, 4\}$, as described in § 1. The segments are numbered counterclockwise, starting with the (bottom) horizontal face ($x \in [-1, 1]$, $y = -1$). The supports of the weight functions are distributed accordingly, so that $w_1(s; 1, 0) = 1$ on segment 1 and vanishes on the other three segments, $w_2(s; 1, 0) = 1$ on segment 1 and vanishes on the other three, $w_5(s; 1, 0) = x$ on segment 1 and vanishes on the other three, $w_6(s; 1, 0) = y$ on segment 2 and vanishes on the others, etc. With these weights all the interface integrals can also be readily evaluated analytically.

Within the disc, on the other hand, the most appropriate coordinate system is the polar one, involving the variables r and θ, where r is measured from the center of the disc, located at (x_0, y_0). The basis functions consist of products of nonnegative powers of r with the trigonometric functions $\{1, \sin\theta, \cos\theta, \sin 2\theta, \cos 2\theta, \cdots\}$. If we denote the jth function in this sequence by $\tau_j(\theta)$, we have the relation:

$$\tau_j(\theta) = (1 - t_j) \sin(p_j\theta) + t_j \cos(p_j\theta) \tag{3.2a}$$

where

$$p_j = [\tfrac{1}{2}j], \qquad t_j = j \pmod 2 \tag{3.2b}$$

and the bracket indicates that the integer part of $\frac{1}{2}j$ is being used. Clearly, we are assuming that $j = 1, 2, 3, \cdots$, while if we denote the power of r by i, we also have $i = 0, 1, 2, \cdots$.

There is one restriction on the combinations of i and j. When $i = 0$, we do not allow j to exceed unity because the basis functions with $i = 0$ would be discontinuous at the origin, and we do not wish to permit this in the problems we shall solve. More precisely, the basis functions with $i = 0$, viz., $\sin\theta$, $\cos\theta$, etc. are not elements of the Sobolev space $H^1(\Omega_2)$, which they are required to be in the selfadjoint formulation

of the cell method described here. Use of these basis functions would cause the integrals (2.10a) needed for $S_{\mu\nu}(2)$ to diverge. This problem does not arise for $i>0$. Thus, the basis functions in Ω_2 may be displayed as follows:

$$\begin{aligned} &1, \\ &r,\ r\sin\theta,\ r\cos\theta,\ r\sin 2\theta,\ r\cos 2\theta,\ \cdots, \\ &r^2,\ r^2\sin\theta,\ r^2\cos\theta,\ r^2\sin 2\theta,\ r^2\cos 2\theta,\ \cdots \end{aligned} \tag{3.3}$$

The labeling by the index μ in Ω_2 follows the obvious sequence shown in (3.3), i.e., term-by-term on each row, and one row after the other. Since we are dealing with finite approximations, we must obviously terminate each row as well as the sequence of rows. Let the maximum value of i be denoted by I, the maximum value of j by J, and the maximum multiplier of θ by $p_{\max}$ (hence, as is clear from (3.3), $J=2p_{\max}+1$). A function $\mu(i,j)$, which gives the value of the index μ in terms of those of i and j may be defined as follows:

$$\begin{aligned} &\mu(0,1)=1, \\ &\mu(0,j>i)\ \text{undefined}, \\ &\mu(i>0,j)=J(i-1)+j+1. \end{aligned} \tag{3.4}$$

Using this function, expressions for the basis functions in Ω_2 may be written in concise form, interpreting, as before, the mapping between indices to go in either direction:

$$\begin{aligned} &\phi_\mu(i,j)^{(r,\theta;2)}=r^i\cdot\tau_j(\theta), \\ &\phi_\mu(r,\theta;2)=r^{i(\mu)}\cdot\tau_{j(\mu)}(\theta). \end{aligned} \tag{3.5}$$

The most reasonable choice for the system of weight functions on Γ_{12} is just the set $\{\tau_j(\theta)\}$. With this choice, the moment collocation integrals over Γ_{12}, involving the basis sets of both cells Ω_1 and Ω_2, may be done analytically.

4. Evaluation of matrix quantities. In this section, the computation of the various arrays necessary to set up (2.18) is presented. We will discuss the nature of the integrations needed for evaluating the S matrices, the U matrices, and finally make note of the integrals needed in finding the T's and W. The description of finding $T(k)$, $k=1,2$ and W will be given for a particular choice of $c(k)$ in (1.1) and specific boundary conditions. It will be seen that, because of the particular choice of basis and weight function which was made, all of the necessary integrations can be obtained as finite sums, and that elementary functions appear throughout. This means that no error is introduced by truncation of series, the use of asymptotics, or numerical integration. Hence, all computations are exact to machine precision, so that any discretization errors are those of the CD method itself.

4.1. The S matrices. According to (2.10a) and the choice of monomial basis functions in the subdomain Ω_1, the matrix $S(1)$ is given by

$$S_{\mu\nu}(1)=\iint_{\Omega_1}\left\{a(1)\left(\frac{\partial\phi_\mu}{\partial x}\frac{\partial\phi_\nu}{\partial x}+\frac{\partial\phi_\mu}{\partial y}\frac{\partial\phi_\nu}{\partial y}\right)+b(1)\phi_\mu(x,y;1)\phi_\nu(x,y;1)\right\}dx\,dy \tag{4.1}$$

where $\mu,\nu=1,\cdots,M(1)$ and the derivatives are understood to have the same arguments as the functions below them. We remind the reader that the region of integration is the set $\Omega_1=\Omega-\Omega_2$, or, more explicitly: the set

$$\Omega_1=\{-1\leqq x,y\leqq 1\}-\{(x-x_0)^2+(y-y_0)^2\leqq r_0^2\}. \tag{4.2}$$

For the problems to be discussed here, we will only be interested in the case where $a(1)$ and $b(1)$ are constants. Under these circumstances, when the basis set in Ω_1 consists of the monomials defined in (3.1), each term of the integrand in (4.1) will be a monomial. Hence, the evaluation of $S(1)$ may be accomplished as an integration of monomials over the square region Ω minus the integration of the same monomials over the circular disc Ω_2.

The integration of monomials over the square involves only the trivial generic integral

$$G_s \equiv \int_{-1}^{1}\int_{-1}^{1} x^i y^j \, dx\, dy = \begin{cases} 0, & i \text{ or } j \text{ odd}, \\ \dfrac{4}{(i+1)(j+1)}, & i \text{ and } j \text{ even}. \end{cases} \tag{4.3}$$

As mentioned above, the contribution from the circular disc Ω_2 is subtracted from G_s to form the $S(1)$ matrix. In order to integrate monomials over Ω_2, we need to evaluate the generic integral

$$G_d(1) \equiv \iint_{\Omega_2} x^i y^j \, dx\, dy. \tag{4.4}$$

To do this, we substitute for x and y:

$$x = x_0 + r\cos\theta, \qquad y = y_0 + r\sin\theta \qquad (0 \leqq r \leqq r_0)(0 \leqq \theta \leqq 2\pi) \tag{4.5}$$

and the integral can be expanded by the binomial theorem into a finite sum of integrals of the following simple form (see [3, p. 21]):

$$\int_0^{2\pi} \cos^m\theta \sin^n\theta \, d\theta = \begin{cases} \dfrac{\pi}{2^{m+n-1}} \dfrac{m!n!}{(m/2)!(n/2)!((m+n(/2)!}, & m \text{ and } n \text{ even}, \\ 0, & m \text{ or } n \text{ odd}. \end{cases} \tag{4.6}$$

Thus, by evaluating expressions of the type (4.3), (4.7) and (4.6), the integrations necessary for $S(1)$ can be performed.

In contrast to formula (4.1), the use of polar coordinates in Ω_2 results in the expression for $S(2)$:

$$S_{\mu\nu}(2) = \int_0^{2\pi}\int_0^{r_0} \left\{ a(2)\left(\frac{\partial\phi_\mu}{\partial r}\frac{\partial\phi_\nu}{\partial r} + \frac{1}{r^2}\frac{\partial\phi_\mu}{\partial\theta}\frac{\partial\phi_\nu}{\partial\theta} \right) + b(2)\phi_\mu(r,\theta;2)\phi_\nu(r,\theta;2) \right\} r\, dr\, d\theta \tag{4.7}$$

where $\mu, \nu = 1, \cdots, M(2)$. (Again, the derivatives are understood to have the same arguments as the functions below them). In computing $S(2)$, we only allow for constant $a(2)$ and $b(2)$. In this case, using the basis set $\{r^i \cdot \tau_j(\theta)\}$, it is clear that integrals of the following type are required to find $S(2)$:

$$G_d(2) = \int_0^{2\pi}\int_0^{r_0} r^i \begin{Bmatrix} \cos p\theta \\ \sin p\theta \\ 1 \end{Bmatrix} \begin{Bmatrix} \cos q\theta \\ \sin q\theta \\ 1 \end{Bmatrix} dr\, d\theta. \tag{4.8}$$

The notation

$$\begin{Bmatrix} f(\theta) \\ g(\theta) \\ 1 \end{Bmatrix}$$

means that one of the functions is a possible factor in the integrand. These integrals are easily evaluated by using the orthogonality of sines and cosines on the interval $[0, 2\pi]$.

4.2. The U matrices. Before proceeding with our discussion of the U matrices in the Circle in the Square problem, we recall that the calculation of any $U(k, m)$ matrix follows the procedure of restricting the basis functions $\phi(x; k)$ to the interface Γ_{km}, multiplying by weights $w(s; k, m)$, and performing a surface integration with respect to the variable s. For the Circle in the Square geometry, we have

$$d\Gamma_{10}(s) = \begin{cases} dx, & -1 \leqq x \leqq 1, \quad y = \pm 1, \\ dy, & -1 \leqq y \leqq 1, \quad x = \pm 1, \end{cases} \tag{4.9}$$

$$d\Gamma_{12}(s) = r_0\, d\theta, \qquad 0 \leqq \theta \leqq 2\pi.$$

Of the three U matrices arising in the Circle in the Square problem, $U(1, 0)$ and $U(2, 1)$ require very simple integrations for evaluation of their entries. However, computing $U(1, 2)$ requires a substantially new class of integrals beyond those encountered in finding the S matrices, so that we will give a rather detailed description of this calculation.

The calculation of $U(1, 0)$ reduces to trivial integrations because the basis functions in Ω_1 are monomials and the weight functions on Γ_{10} are either powers of x or of y. Hence, the integrals required for $U(1, 0)$ are of the type (see § 3 and (4.9)):

$$\int_{-1}^{1} x^l x^i y^j\, dx \Big|_{y=\pm 1} \quad \text{or} \quad \int_{-1}^{1} y^m x^i y^j\, dy \Big|_{x=\pm 1} \tag{4.10}$$

where l or m is the degree of the weight function, and i, j are the exponents appropriate to a generic monomial basis function.

For the calculation of $U(1, 2)$, it is possible to make use of the angular integrations in (4.8). This results from the fact that both the basis functions in Ω_2 and the weight functions on Γ_{12} include sines and cosines.

In order to calculate $U(1, 2)$, we first restrict the basis functions in Ω_1 to the circle Γ_{12} by setting $r = r_0$ in (4.5). Next, we multiply by the set of weight functions $\{\tau_j(\theta)\}$ and integrate with respect to θ, using (4.9). Therefore, the calculation of $U(1, 2)$ involves integrals of the form:

$$G_{12} \equiv r_0 \int_0^{2\pi} (x_0 + r_0 \cos\theta)^i (y_0 + r_0 \sin\theta)^j \begin{Bmatrix} \sin p_j\theta \\ \cos p_j\theta \end{Bmatrix} d\theta. \tag{4.11a}$$

Using the binomial theorem gives

$$G_{12} = r_0 \sum_{m=0}^{i} \sum_{n=0}^{j} \binom{i}{m} \binom{j}{n} x_0^{i-m} y_0^{j-m} r_0^{m+n} \int_0^{2\pi} \begin{Bmatrix} \sin p_j\theta \\ \cos p_j\theta \end{Bmatrix} \cos^m\theta \sin^m\theta\, d\theta. \tag{4.11b}$$

The double summation is necessary when x_0 and y_0 do not vanish.

It is known [4, pp. 375, 477] that the angular integrations in (4.11) can be written in terms of the Beta-Function. However, for integral values of m and p, these reduce to finite sums of binomial coefficients. Hence, G_{12} can itself be represented as a finite sum of elementary algebraic terms. We shall not pursue the details further.

4.3. The T and W matrices. We will describe the generation of the T and W arrays for a specific right-hand side and boundary condition for equation (1.1). We make no attempt to describe the most general $c(k)$ and boundary condition which can be treated by our computer program. Instead, we will only mention the cases relevant to the model problems.

In § 5, we will be concerned with finding a numerical approximation to a Green's function for the square domain Ω. We will be interested in the case where the right-hand side of (1.1) is a delta-function with singularity at the center of the circle Γ_{12}:

$$c = \begin{cases} 0, & (x, y) \in \Omega_1, \\ \delta(x - x_0, y - y_0), & (x, y) \in \Omega_2. \end{cases} \tag{4.12}$$

The Green's function will be subject to a homogeneous Dirichlet boundary condition, i.e.,

$$\psi(\pm 1, y) = \psi(x, \pm 1) = 0, \qquad -1 \leqq x, \quad y \leqq 1. \tag{4.13}$$

We will show that when a delta-function, as in (4.12), is the only inhomogeneity in (1.1), the only contribution to the right-hand side of (2.18), in the case of zero boundary condition, is that of the first component of $T(2)$, which is equal to 1.

Using definitions (2.12) and (2.11b) for $W(1, 0)$, it is obvious that the boundary condition (4.13) results in $W_\alpha(1, 0) = 0$ for $\alpha = 1, \cdots, L(1, 0)$. Similarly, applying definition (2.10b) for the case $c(1) = 0$ immediately gives $T_\nu(1) = 0$ for $\nu = 1, \cdots, M(1)$.

In order to illustrate the calculation of $T_\mu(2)$ for the case (4.12), let us first consider the situation when

$$c(2) = \delta(x - \bar{x}, y - \bar{y}) \tag{4.14}$$

where (x, y) and $(\bar{x}, \bar{y})$ are in Ω_2. According to the definition (2.10b), we have

$$T_\mu(2) = \int_{\Omega_2} c(2) \phi_\mu(2) \, d\Omega_2 \tag{4.15a}$$

$$= \int_0^{2\pi} \int_0^{r_0} c(r, \theta; 2) r^{i(\mu)} \tau_{j(\mu)}(\theta) r \, dr \, d\theta, \qquad \mu = 1, \cdots, M(2) \tag{4.15b}$$

where we have used the notation of § 3 for the basis functions in Ω_2. In order to perform the integrations in (4.15b), we can make use of the change of variable formula (from rectangular to polar coordinates) for the delta-function [5, p. 551],

$$\delta(x - \bar{x}, y - \bar{y}) = \frac{1}{r} \delta(r - \bar{r}) \delta(\theta - \bar{\theta}) \tag{4.16}$$

where $r = (x^2 + y^2)^{1/2}$, $\theta = \tan^{-1}(y/x)$ and $\bar{r}$, $\bar{\theta}$ are similarly related to $\bar{x}$, $\bar{y}$. Then, we have

$$T_\mu(2) = \int_0^{2\pi} \int_0^{r_0} \frac{1}{r} \delta(r - \bar{r}) \delta(\theta - \bar{\theta}) r^{i(\mu)} \tau_{j(\mu)}(\theta) r \, dr \, d\theta \tag{4.17a}$$

$$= \bar{r}^{i(\mu)} \tau_{j(\mu)}(\bar{\theta}). \qquad \mu = 1, \cdots, M(2). \tag{4.17b}$$

This concludes the calculation of $T_\mu(2)$ for the case that $\bar{r} \neq 0$.

We recall that the placement of the delta-function singularity as in (4.12) means that $\bar{r} = 0$ for the (polar) coordinate system of Ω_2. Although there is no obvious interpretation of formula (4.16) when $\bar{r} = 0$, we can nonetheless perform the calculation (4.15) in the following way: We take the limit in formula (4.17b) as $\bar{x} \to x_0$, $\bar{y} \to y_0$, i.e., in terms of the polar coordinates in Ω_2, $\bar{r} \to 0$. Therefore, for the case that

$c(2) = \delta(x - x_0, y - y_0)$ we have

$$T_\mu(2) = \begin{cases} 1, & \mu = 0, \\ 0, & \mu \neq 0, \end{cases} \tag{4.18}$$

i.e., we get zeros for all the elements of $T(2)$, except for the index $\mu = 1$, because that element alone does not contain a positive power of r as a factor. Hence, with homogeneous Dirichlet data and $c(2) = \delta(x - x_0, y - y_0)$, only $T_1(2)$ "drives" the linear system (2.18).

For the first problem studied, $c(k) \equiv 0$ in equation (1.1). Then, by formula (2.10b), the only contribution to the right-hand side of (2.18) is from $W(1, 0)$. We further specialize to a boundary condition which we use in § 5, namely,

$$\psi(\pm 1, y) = 0, \quad \psi(x, -1) = 0, \quad \psi(x, 1) = \cos\frac{\pi x}{2} \qquad (\text{for } -1 \leqq x, y \leqq 1). \tag{4.19}$$

This boundary function has support on segment 3 of the boundary Γ_{10}. Hence only the sequence of weights $w_3(s; 1, 0) = 1$, $w_7(s; 1, 0) = x$, $w_{11}(s; 1, 0) = x^2, \cdots$, when integrated against $\psi(x, 1)$ can possibly give a nonzero result for $W(1, 0)$. Since, however, $\psi(x, 1)$ is also an even function of x, only "half" of these boundary moments will be nonzero. The nonzero moments of $\psi(x, 1)$ will lead to integrals:

$$\int_{-1}^{1} x^i \cos\frac{\pi x}{2}\, dx = \left(\frac{2}{\pi}\right)^{1+1} \int_{-\pi/2}^{\pi/2} u^i(\cos u)\, du \tag{4.20}$$

for i even.

The integrals (4.20) can be computed by recursion. If we define

$$F_m \equiv \int_a^b u^m \cos u\, du \tag{4.21}$$

and integrate by parts, we can obtain:

$$F_m = 2\left(\frac{\pi}{2}\right)^m - m(m-1)F_{m-2} \tag{4.22}$$

with $F_0 = 2$. This recursion formula was used for even m to compute $W(1, 0)$, for the boundary condition (4.19).

In § 5, we will also make use of the following choice of the source term c in equation (1.1):

$$c = \begin{cases} 0, & (x, y) \in \Omega_1, \\ 1, & (x, y) \in \Omega_2 \end{cases} \tag{4.23}$$

along with the homogeneous Dirichlet boundary condition (4.13). For the case defined by (4.23) and (4.12), only $T(2)$ is nonzero on the right-hand side of (2.18). As for the Green's function problem, $W(1, 0)$ is again identically zero, due to the homogeneous boundary condition. The array $T(1)$ is of course zero, since $c(1) = 0$ according to definition (2.10b).

For the calculation of $T(2)$, using (2.10b) and (4.23), we have

$$\begin{aligned} T_\mu(2) &= \int_{\Omega_2} c(2)\phi_\mu(2)\, d\Omega_2 \\ &= \begin{cases} 0, & j(\mu) \neq 1, \\ \dfrac{2\pi r_0^{i(\mu)+2}}{i(\mu)+2}, & j(\mu) = 1 \end{cases} \end{aligned} \tag{4.24}$$

where

$$\phi_\mu(2) = r^{i(\mu)}\tau_{j(\mu)}(\theta). \tag{4.25}$$

5. Description of test problems and numerical results. Three model problems which we have treated in detail will be described. These problems illustrate the handling of a Dirichlet problem, the calculation of an approximation to a singular function (Green's function), and the solution of a problem with discontinuous coefficient and source term. For the last problem, the exact solution was not available, so a fine-mesh finite difference solution was used as a reference standard. All computations were performed in IBM/370 double-precision arithmetic (about 16 decimal digits).

5.1. Laplace's equation. The first model problem requires the solution of Laplace's equation in the whole square $\Omega(\equiv\Omega_1\cup\Omega_2)$, with continuous Dirichlet data on the boundary:

$$\left.\begin{aligned}&\psi(x,-1)=\psi(-1,y)=\psi(1,y)=0\\&\psi(x,1)=\cos\frac{\pi x}{2}\end{aligned}\right\}\quad \text{for } -1\leqq x, y\leqq 1. \tag{5.1}$$

With this boundary condition Laplace's equation separates, and the exact solution ψ_e is easily seen to be:

$$\psi_e(x,y)=\cos\frac{\pi x}{2}\left(A\cosh\frac{\pi y}{2}+B\sinh\frac{\pi y}{2}\right) \tag{5.2}$$

where

$$A=1/2\cosh(\pi/2) \quad \text{and} \quad B=1/2\sinh(\pi/2).$$

The circle was centered at the origin, with radius $r_0=0.5$. The approximate computed solution ψ_c was compared with ψ_e using the L_∞ and L_2 norms. More particularly, we used the maximum norm

$$\|\text{error}\|_\infty(\Omega)\equiv\max_{-1\leqq x,\,y\leqq 1}|\psi_e(x,y)-\psi_c(x,y)| \tag{5.3a}$$

evaluated over a uniform mesh covering the full square Ω. The values of ψ at 400 points were examined to estimate $\|\text{error}\|_\infty(\Omega)$. (A close investigation indicated that this mesh was sufficiently fine to yield an error estimate good to at least two figures). We also used the root-mean-square norm, defined by:

$$\|\text{error}\|_2(\Omega)\equiv\left\{\int_{-1}^{1}\int_{-1}^{1}(\psi_e(x,y)-\psi_c(x,y))^2\,dx\,dy\right\}^{1/2} \tag{5.3b}$$

where the integral is approximated by the trapezoidal rule in two dimensions, using the same mesh used to compute the maximum norm.

We have determined the dependence of the error on some of the parameters of the problem by finding an "optimal" solution and varying each parameter from its "best" value, one at a time. The optimal solution is defined in terms of roundoff error. With r_0, x_0 and y_0 fixed, the numbers $M(1)$, $M(2)$, $L(1,0)$ and $L(1,2)$ were increased until the solution deteriorated due to roundoff error. In other words, the optimal solution was that solution at which $\|\text{error}\|_2(\Omega)$ could not be reduced by increasing any parameter value. An examination of the reciprocal condition number of the coefficient matrix in (2.18) showed no significant decrease for parameter values less than those for the optimal solution. In this sense, the optimal solution is a stable one.

A plot of the CD solution with the optimal cell parameters appears in Fig. 2. Notice how smoothly the solutions in Ω_1 and Ω_2 match on the circle.

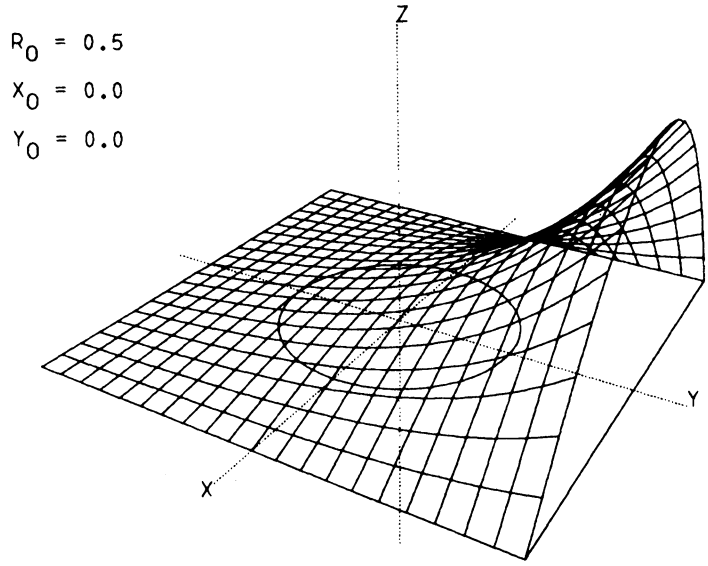


FIG. 2. *A perspective view of the* CD *solution of* (5.1) *with cell parameters* (5.4). *Although* CD *solutions are in general discontinuous, with the size of the mesh employed here (for display), no jump in solution values across the interface is perceptible.*

In Figs. 3a–3c, we have plotted the common logarithm of $\|\text{error}\|_2(\Omega)$ versus the number of weight functions per boundary segment ($=L(1,0)/4$), versus the number of weight functions on the circle ($=L(1,2)$), and versus the maximum power of $r(=1)$ used in the basis set for Ω_2. The optimal solution was found at

$$M(1)=78, \quad L(1,0)=32, \quad L(1,2)=19, \tag{5.4}$$
$$J=19, \qquad I=5,$$

with $\|\text{error}\|_2(\Omega)=2.1\times 10^{-7}$. The approximate reciprocal condition number of the coefficient matrix, which we shall denote by RCN, was 2.5×10^{-9}. The onset of dominant roundoff error can be clearly seen in each figure.

The reader will note that the total number of weight functions on the boundary is a multiple of four. Although it would be possible to provide each side of the square with a different number of weight functions, we chose to keep the number of weight functions on each boundary segment the same. This approach has the advantage of reducing the number of parameters in the Dirichlet problem, and allows us to focus on the difference in the dependence of the error on the boundary weights and on the weights on the circle. As for the number of weights on the circle, we always take $L(1,2)$ to be an odd integer. In this way, the functions $\cos p_{\max}\theta$ and $\sin p_{\max}\theta$ are included simultaneously in the set of weights.

As can be seen in Figs. 3a and 3b, the rate of decrease of $\|\text{error}\|_2(\Omega)$ as a function of $L(1,2)$ is almost constant (until roundoff dominates), while the dependence on $L(1,0)$ "alternates" according to whether the number of weight functions is an odd or an even multiple of four. This effect is very likely related to the fact that as $\frac{1}{4}L(1,0)$ goes from an odd to an even integer, the additional moments of the boundary function are all zero.

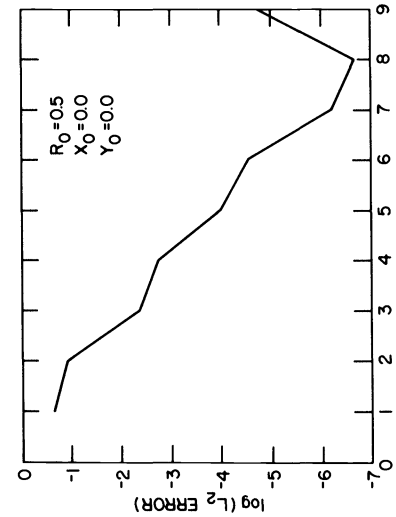


FIG. 3a. *The common logarithm of* $\|error\|_2$ *for the solution of* (5.1) *is plotted as a function of the number of weights per boundary segment. The circle* Γ_{12} *is centered at the origin with radius* $r_0 = .5$.

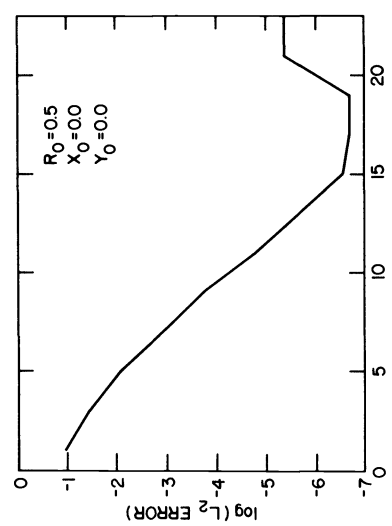


FIG. 3b. *The common logarithm of* $\|error\|_2$ *for the solution of* (5.1) *is plotted as a function of the number of weights on the circular interface* Γ_{12}. *The circle* Γ_{12} *is centered at the origin with radius* $r_0 = .5$.

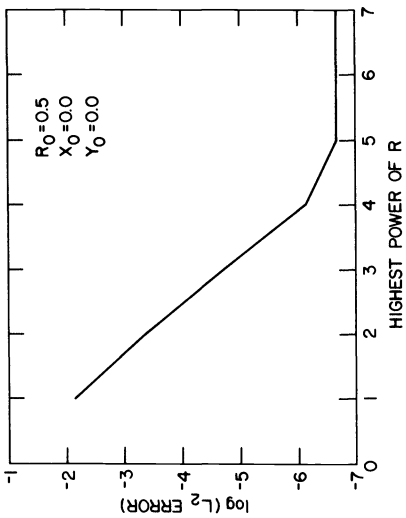


FIG. 3c. *The common logarithm of* $\|error\|_2$ *for the solution of* (5.1) *is plotted as a function of the maximum power of* r *used in the set of basis functions for* Ω_1. *The circle* Γ_{12} *is centered at the origin with radius* $r_0 = .5$.

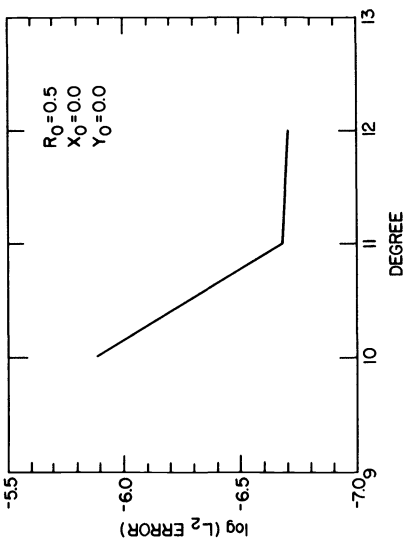


FIG. 3d. *The common logarithm of* $\|error\|_2$ *for the solution of* (5.1) *is plotted as a function of the highest degree of the monomials used in the set of basis functions for* Ω_1. *The circle* Γ_{12} *is centered at the origin with radius* $r_0 = .5$.

Observe that with the number of weight functions fixed, the number of basis functions cannot be varied arbitrarily, due to the collocation conditions (e.g., the former cannot be less than the latter). Moreover, it seems desirable to use only those values of $M(1)$ which correspond to a full set of monomials up to a given degree. In view of these limitations, we give in Fig. 3d a plot of $\log \|\text{error}\|_2(\Omega)$ versus the degree of $\psi(1)$, for three values of $M(1)$ only. These cases correspond to $M(1)=66$, 78 and 91. The coefficient matrix for the case $M(1)=55$ was found to be singular (this is related to the so-called *degeneracy* described in [1]).

A geometrically asymmetric version of the above model problem chosen for study had $(x_0, y_0)=(0.25, 0.6)$ and $r_0=3$. Although the boundary condition is symmetric with respect to the y-axis, with this location, the circle is not symmetrically placed with respect to any line of symmetry of the square. The optimal solution was found to occur for

$$M(1)=78, \quad L(1,0)=32, \quad L(1,2)=15, \quad J=15, \quad I=5, \tag{5.5}$$

with $\|\text{error}\|_2(\Omega)=2.2\times 10^{-7}$. The approximate reciprocal condition number RCN was 2.8×10^{-11}. The increase of the condition number over that of the previous case is probably due to the powers of r_0, x_0 and y_0 that enter into the various integrals by way of equations (4.7) and (4.11). In Figs. 4a–c, a semilog plot is again given of $\|\text{error}\|_2(\Omega)$ versus the number of weight functions per boundary segment, versus the number of weight functions on the circle and versus the maximum power of r used in the basis set for Ω_2.

5.2. The Green's function. The next problem we considered in depth was that of obtaining an approximate Green's function for Laplace's equation. That is, we solved

$$-\nabla^2 G(x, y; x', y') = \delta(x-x', y-y'), \qquad -1 \leqq x, y, x', y' \leqq 1 \tag{5.6a}$$

subject to

$$G(\pm 1, y; x', y') = G(x, \pm 1; x', y') = 0 \tag{5.6b}$$

with the same set of basis and weight functions as described previously. Here the geometry is such that the circle is centered at $(x_0, y_0)=(x', y')$. This placement of the circle would be useful in those applications where maintaining radial symmetry about the point source is important. The circle provides an interface for "matching" the singular behavior at (x', y') to the rest of the square.

For (5.6a), it is well known that the solution of the $2D$ problem near (x', y') behaves like the negative logarithm of the radial distance from the point source. A closed form solution by Fourier series expansion (cf. [5, p. 523]) is

$$G(x, y; x', y') = \frac{2}{\pi} \sum_{m=1}^{\infty} \frac{\sinh(m\pi(1+y)/2)\sinh(m\pi(1-y')/2)}{m \sinh m\pi} \times \begin{cases} \sin(m\pi x/2)\sin(m\pi x'/2) & \text{for } m \text{ even}, \\ \cos(m\pi x/2)\cos(m\pi x'/2) & \text{for } m \text{ odd} \end{cases} \tag{5.7}$$

for $y<y'$. The same expression, with y and y' interchanged, is used when $y>y'$. We have used this form of the exact solution to compute the error of the solution obtained by using the cell method. The solution is infinite at (x', y')—the series diverges there—and we must discard this point in computing the maximum error. More will be said of the error computation further on.

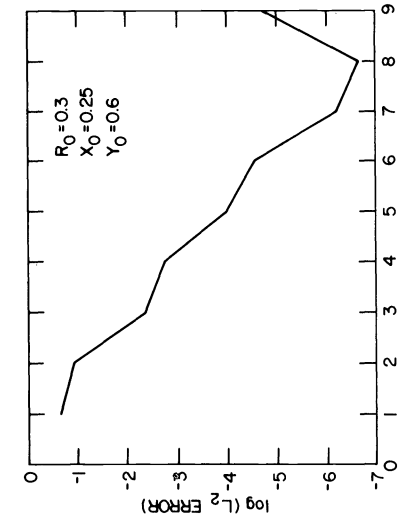


FIG. 4a *The common logarithm of* $\|error\|_2$ *for the solution of* (5.1) *is plotted as a function of the number of weights per boundary segment. The circle* Γ_{12} *is centered at* $(x_0, y_0) = (.25, .6)$ *with radius* $r_0 = .3$.

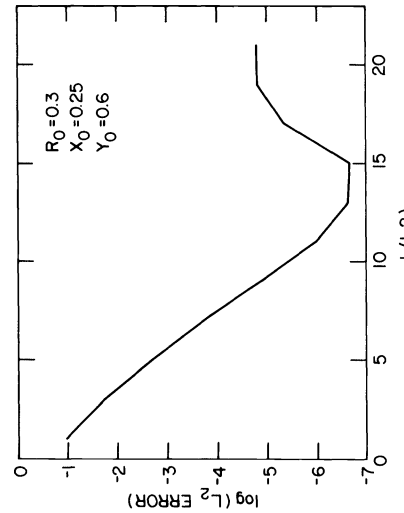


FIG. 4b. *The common logarithm of* $\|error\|_2$ *for the solution of* (5.1) *is plotted as a function of the number of weights on the circular interface* Γ_{12}. *The circle* Γ_{12} *is centered at* $(x_0, y_0) = (.25, .6)$ *with radius* $r_0 = .3$.

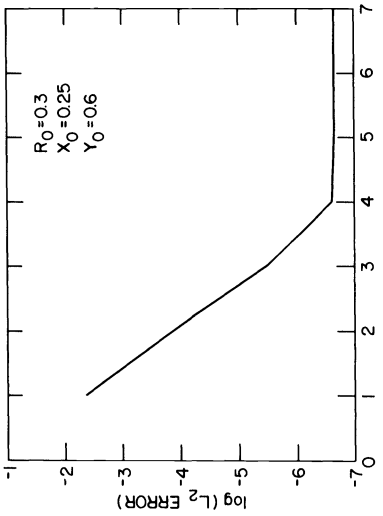


FIG. 4c. *The common logarithm of* $\|error\|_2$ *for the solution of* (5.1) *is plotted as a function of the maximum power of r used in the set of basis functions for* Ω_1. *The circle* Γ_{12} *is centered at* $(x_0, y_0) = (.25, .6)$ *with radius* $r_0 = .3$.

We bring the reader's attention to the fact that an unmodified logarithmic function $\log(r)$ is not in the Sobolev space $H^1(\Omega)$, although it does belong to $H^0(\Omega) = L_2(\Omega)$. Rather, $\log(r)$ belongs to any Sobolev space $H^{1-\varepsilon(\Omega)}$ for any $\varepsilon > 0$. The elliptic boundary value problems which we may hope to solve exactly with the present CD method (in the limit as $L(k, m) \to \infty$, $M(k, m) \to \infty$ and $h \to 0$—where h is the maximum cell diameter) are well-posed problems in $H^1(\Omega)$. Hence, $\log(r)$ is not an admissible basis function for the variational formulation of the CD method.

As we cannot include an unmodified log function in the basis set for Ω_2 in the present scheme, the CD method must give, at best, only an approximation to the Green's function. The basis set we have used in Ω_2 does not contain a function which is singular at any finite point, so that the numerical solution cannot become unbounded on any subregion of Ω. In particular, the numerical Green's function must have a finite value at (x', y').

In light of the above discussion, a measure of pointwise convergence at or near (x', y') is not meaningful. For this reason, we have not tried to evaluate $\|\text{error}\|_\infty(\Omega)$ as previously defined. The numerical approximation of $\|\text{error}\|_2(\Omega)$ also involves careful handling of the region near the singularity. We have not attempted the numerical approximation of the singular integrals needed for finding $\|\text{error}\|_2(\Omega)$ even though, mathematically, the exact Green's function is in L_2. We have chosen to compute the maximum error of the numerical solution in the region Ω with a small disc of radius r_1 deleted. We will denote the maximum error of the numerical solution over the region

$$\Omega' \equiv \Omega - \{(x-x')^2 + (y-y')^2 \leqq r_1^2\} \tag{5.8}$$

by $\|\text{error}\|_\infty(\Omega')$. That is, we evaluate the maximum difference between the exact and computed solution over the square Ω, deleting the disc of radius r_1 centered at $(x'y')$.

In using (5.7) as the exact solution, two computational considerations come into play. The first is that when m is sufficiently large, the arguments of the hyperbolic sines in (5.7) will be large and lead to overflow of the computer arithmetic. We avoided this problem by rewriting the series in terms of exponentials with negative arguments. The second computational consideration is when to truncate the series in (5.7). A numerical comparison of the summation of (5.7) with 500 terms versus 1000 terms indicated convergence to at least four decimal places for points (x, y) at a distance r_1 of more than 0.012 from (x', y'). Four digits of precision, as a result of summing (5.7) to 500 terms, turned out to be more than enough for our purposes.

By the symmetry of the Green's function problem, when $x' = y' = 0$, it is sufficient to calculate the maximum error in a quadrant in order to find the maximum error in all of Ω. The particular quadrant used was the one described by: $y \in [0, 1]$, $x \in [-y, y]$. The maximum error was also computed on the line: $y = 1$, $-1 \leqq x \leqq 1$, which again by symmetry is sufficient to evaluate $\|\text{error}\|_\infty$ on all of the boundary of Ω (which we have denoted by Γ_{10}). We use the notation $\|\text{error}\|_\infty(\Gamma_{10})$ for the maximum error on the boundary of Ω, computed in this way.

We give numerical results for the Green's function problem when Γ_{12} is centered at $(x', y') = (0, 0)$, using a circle with radius $r_0 = 0.5$. The numerical as well as theoretical aspects of this model problem differ markedly from the first. In distinction to the first model problem, the maximum error outside of a disc of radius .012 could not be reduced past a value of roughly .044. That is, $\|\text{error}\|_\infty(\Omega')$ could not be diminished indefinitely (until roundoff error dominated) by increasing the number of cell parameters. The first solution at which this minimal error within Ω' occurred was characterized by:

$$M(1)=21, \quad L(1,0)=12, \quad L(1.2)=7, \tag{5.9}$$
$$I=4, \qquad J=7$$

with RCN $=5.0\times10^{-7}$ and $\|\text{error}\|_\infty(\Omega')=.044$. The maximum error on the boundary was found to be: $\|\text{error}\|_\infty(\Gamma_{10})=.013$, comparable with the maximum error outside $r_1=.012$.

A block-centered finite difference calculation for the Green's function problem (5.6) was also performed. An examination of the maximum error outside the circle of radius $r_1=.012$ showed a behavior in $\|\text{error}\|_\infty(\Omega')$ similar to that for the CD solution. For the finite difference solution, the maximum error remained on the order of 10^{-3}, with a significant increase in grid dimensions having little effect.

In order to obtain high accuracy near the singularity in (5.6), both the finite difference and finite element methods have needed to introduce modifications in their respective algorithms. A brief description of alternatives for these methods is given in [6]. One approach in FEM work has been to augment the basis set with a logarithmic function. Although the details of these methods vary, the CD method has a similar treatment of delta-function singularities. A discussion of various approaches for CD, their implementation for the Circle in the Square geometry, and their relation to other numerical methods will be taken up in a forthcoming paper.

Although the maximum error of the CD solution in Ω' could not be reduced below a certain value, the maximum error on the boundary of Ω could be decreased by increasing the numbers of weight and basis functions. (This improvement is possible because $\|\text{error}\|_\infty(\Omega')$ tends to be found at points closer to the deleted disc of radius r_1 for increasing parameter values). For instance, for the solution with the parameter values:

$$M(1)=78, \quad L(1,0)=32, \quad L(1,2)=19, \tag{5.10}$$
$$1=8, \qquad J=19.$$

$\|\text{error}\|_\infty(\Gamma_{10})$ has a significant decrease from the above case, while $\|\text{error}\|_\infty(\Omega')$ has a slight increase. These errors are: $\|\text{error}\|_\infty(\Gamma_{10})=8.8\times10^{-4}$, $\|\text{error}\|_\infty(\Omega')=.048$ and RCN $=7.1\times10^{-15}$.

We believe this behavior in $\|\text{error}\|_\infty(\Omega')$ is the result of attempting to fit the log function with the set of functions $\{r^i\cdot\tau_j(\theta)\}$. The nature of this approximation seems to be quite similar to a least-squares fit. Having performed a least-squares fit of polynomials in r to a log function on the interval $[0, r_0]$, we can note the following points which tend to substantiate this claim. First, the resulting least-squares coefficients alternate in sign and second, they grow in magnitude in a manner similar to the coefficients of the CD solution of the Green's function problem. In particular, we obtain numerically, for the case when $(x', y')=(0,0)$ (ignoring a small angular dependence), an approximation to the Green's function of the form

$$\psi_c=\alpha-ar+br^2-cr^3+dr^4-\cdots \tag{5.11}$$

where $\alpha, a, b, \cdots$ are positive constants. In practice, we typically find values of α, a, and b on the order of 1, 10, and 100, respectively. Third, the maximum norm of the error between the least-squares approximant and the fitted function on the interval $[.01, r_0]$ is comparable to the errors $\|\text{error}\|_\infty(\Omega')$ reported above.

A plot of a typical Green's function obtained with the cell method is given in Fig. 5. This case had cell parameters $L(1,0)=32$, $L(1.2)=13$, $J=13$, $I=6$, $M(1)=78$, $r_0=.3$ and $(x_0, y_0)=(.25, .6)$. The finite "peak" and the decreased angular dependence

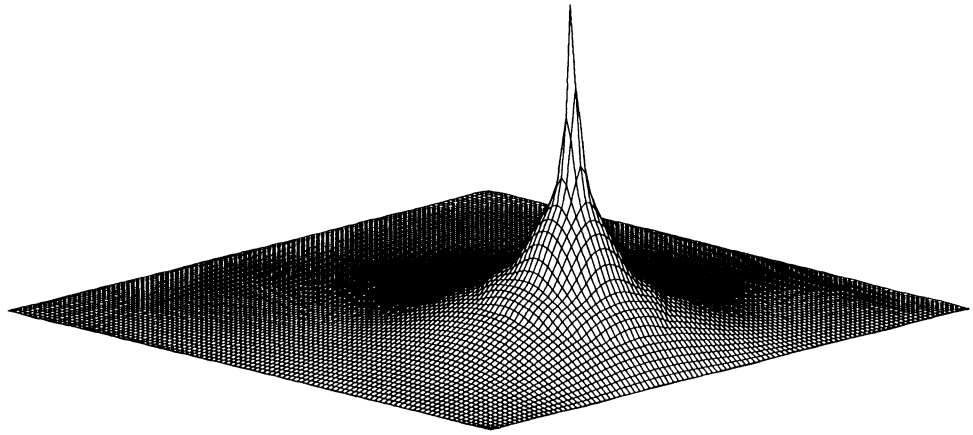


FIG. 5. *A perspective view of a typical Green's function obtained with the cell method. The circle* Γ_{12} *is centered at* $(x_0, y_0) = (.25, .6)$, *with radius* $r_0 = .3$. *The* CD *solution has a "peak" value of* $\psi_c(x_0, y_0) = .91$. *Some irregularity in the solution near and within the circle can be seen, where the solution attempts to approximate a logarithmic singularity and to "match" it to a monomial representation in* Ω_1.

as compared with the first model problem are obvious features. The "peak" value of the CD solution is $\psi_c(x_0, y_0) = .91$. Some irregularity of the solution near and within the circle can be seen where the solution attempts to approximate a logarithmic singularity and to "match" it, at Γ_{12}, to a monomial representation in Ω_1.

5.3. Poisson's equation. The third test problem which we have solved by the cell method is given as follows:

$$-\nabla \cdot (a \nabla \psi(x, y)) = c, \qquad -1 \leqq x, y \leqq 1 \tag{5.12}$$

where a and c are constants within a given cell. The specific choice of a and c is:

$$a = \begin{cases} 1, & (x, y) \in \Omega_1, \\ 2, & (x, y) \in \Omega_2. \end{cases} \qquad c = \begin{cases} 0, & (x, y) \in \Omega_1, \\ 1, & (x, y) \in \Omega_2. \end{cases} \tag{5.13}$$

The boundary condition is homogeneous Dirichlet:

$$\psi(\pm 1, y) = \psi(x, \pm 1) = 0, \qquad -1 \leqq x, y \leqq 1. \tag{5.14}$$

As far as we know, an analytic solution of the above problem has not been given, even though a and c are piecewise constant.

Since an analytical solution of (5.12) was not available, we compared the results of our CD calculations to a fine-mesh finite difference solution. It was found that the difference between the CD solution and the finite difference solution was on the same order as the CD solution's departure from the value of zero on the boundary. The cell geometry used for this calculation was determined by $r_0 = .5$, $x_0 = y_0 = 0$. Using the cell parameters (5.4) of the optimal solution of the first model problem gave a CD solution differing from the finite difference solution in the fourth decimal place.

The coefficient a and source term c are bounded but discontinuous functions of x and y. As such, we should expect the convergence of the cell method solution of this model problem to be no faster than that for the first model problem, and no slower than that for the Green's function problem. Although we have not performed an exhaustive study of the convergence properties of the CD solution of (5.12), this expectation is confirmed by the cases we have examined. As an example, using the

cell parameters (5.4) results in a solution with an error on the boundary of $\|\text{error}\|_\infty(\Gamma_{10}) = 6.9 \times 10^{-4}$. This value is greater than the maximum error of the optimal solution of the first model problem. In addition, this error value is comparable to the smallest maximum error obtainable on the boundary for the Green's function problem, using a larger number of parameters. These results tend to confirm that the CD solution of the third model problem is intermediate in its rate of convergence when compared to the other two problems.

In Fig. 6, a plot of a CD solution of (5.12) is presented. The approximate value of the computed solution at the origin is 0.127.

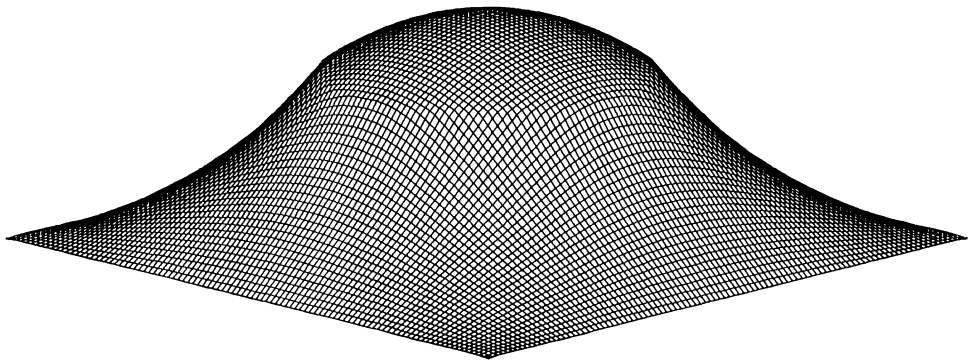


FIG. 6. *A perspective view of a* CD *solution of* (5.12). *The convex "dome" results from the nonzero constant source term within the circle. The solution within* Ω_2 *is smoothly joined across* Γ_{12} *to that in* Ω_1, *where the boundary condition is homogeneous Dirichlet. The approximate value of the computed solution at the origin is* 0.127.

6. Conclusion. We have presented here the first treatment of a p.d.e. by the CD method where a nonrectangular domain is involved, in this case, the circular disc in the square. We have shown in detail how easily the use of two subdomains, each with a different coordinate system, fits into the CD scheme. We recall that the interior and interface integrations have been performed exactly, i.e., the circular interface has not been approximated in any way, and all the integrations have been performed analytically. Furthermore, the particular choice of weight and basis functions was made on the basis of the convenience with which the various integrals could be evaluated, and of programming convenience, but not as a result of constraints placed by the CD method.

Although this work cannot pretend to be a general-purpose implementation of CD, we have nonetheless investigated some special cases with a certain claim to general interest. We will briefly review some of the numerical results of our treatment of the model problems.

As described in § 5, for the first model problem (5.1), it was found that accurate solutions could be obtained. The numerical L_2 and L_∞ errors of the solution (the two errors are comparable for this problem) appear to behave continuously as functions of various cell parameters. This result is consistent with the well-posed nature of the variational formulation of the problem.

On the other hand, the Green's function problem (5.6) is not well-posed in the Sobolev space $H^1(\Omega)$. Hence, for the version of the CD method used here, we can expect the solution to have difficulty converging to the logarithmic singularity. It was observed that the maximum error for the Green's function problem was never less than the maximum error for the first model problem, when using a similar number of

cell parameters. This situation prevailed whether $\|\text{error}\|_\infty(\Gamma_{10})$ or $\|\text{error}\|_\infty(\Omega')$ was used as a measure of the error in the Green's function problem. Thus, the numerical behavior of the maximum error seems to reflect the difficulty of convergence to the logarithmic singularity.

Despite the difference in the convergence properties of the solutions of the first two model problems, the CD formulation may be a viable alternative for treating the Green's function problem. Inasmuch as special handling is called for in any numerical treatment of a singularity such as appears in case (5.6), and keeping in mind that the basis set in Ω_2 does not contain a logarithm, the results suggest that the cell method performs satisfactorily away from the singularity and in particular, in Ω_1. By modifying the cell geometry or the basis set used, it may be possible to obtain accurate results near the singular point. A similar procedure has, of necessity, been used in the finite difference method, and analogous techniques are known in the FEM [6].

The third model problem is a simple example of a true interface problem, in that the coefficient a is not smooth (or even continuous). The classical formulation of the problem would be to require that (5.12) hold separately in Ω_1 and Ω_2, and that ψ and $a\,\partial\psi/\partial n$ be continuous across Γ_{12}, n being the normal to the interface Γ_{12}. The weak form of (5.12), on which the CD formalism is based, remains posed in $H^1(\Omega)$. Numerically, we found the problem (5.12) to be intermediate in ease of solution by the CD method, when compared to the first two model problems.

In order to give the reader an idea of the amount of computational effort required using the cell method, we can compare the size and nature of the discrete equation systems for the CD solution—with and without pretransformation—to that for a finite difference solution. Since we have already discussed the finite difference solution to the third model problem, we shall make the computational comparison for that problem.

For the Circle in the Square geometry (which has only two subdomains), the total number of basis functions is $M(1)+M(2)$ and the total number of weight functions is $L(1,0)+L(1.2)$. Hence, the total number of unknowns (including the Lagrange multipliers) is $M_T^+ \equiv M(1)+M(2)+L(1,0)+L(1,2)$. In general, the matrices in (2.18), aside from the zero blocks, are full and we have a linear system of order $M_T^+ \times M_T^+$. The value of M_T^+ is usually smaller than that for the optimal solution (5.4), where $M_T^+ = 225$. A direct method of solution can be used in this case where there are a small number of unknowns.

With the pretransformation technique [1] to variables $\{\sigma_{km}, \rho_k;\ k=1,2;\ m=m[k]\}$, the total number of unknowns for the Circle in the Square problem can be reduced to well below M_T^+. To be precise, the total number of unknowns will be $M_T^- \equiv M(1)+M(2)-L(1,0)-L(1,2)$, of which there are $L(1,2)\sigma(1,2)$'s, $N(1)\rho(1)$'s and $N(2)\rho(2)$'s, where $N(1) \equiv M(1)-L(1,0)-L(1,2)$ and $N(2) \equiv M(2)-L(1,2)$. Also as a result of the pretransformation, the equations for the σ's and ρ's are uncoupled so that, as shown in [1], each set of these variables can be solved for independently of the others. These features facilitate the solution of very large problems [2].

For the third model problem, CD solutions with $\|\text{error}\|_\infty(\Gamma_{10})$ on the order of 10^{-4} could be obtained. For such CD solutions without using pretransformation, a linear system with an M_T^+ of about 225 had to be solved. If pretransformation were used, the solution of three uncoupled linear systems of orders $L(1,2)\times L(1,2)$, $N(1)\times N(1)$ and $N(2)\times N(2)$ would be required. As an example, for the cell parameters (5.4), we have $L(1,2)=19$, $N(1)=27$ and $N(2)=77$.

The finite difference calculations for the third model problem required a grid of 50×50 unknowns for $\|\text{error}\|_\infty(\Omega)=1.6\times 10^{-3}$, and a grid of 100×100 unknowns for $\|\text{error}\|_\infty(\Omega)=2.9\times 10^{-5}$. (The quantity $\|\text{error}\|_\infty(\Omega)$ is here the maximum difference

between the finite difference solution to which it applies and a "standard" finite difference solution using a 120×120 grid.) To solve the system resulting from the use of a 100×100 grid, a total of 170 conjugate gradient iterations were needed. A finite difference solution based on a grid somewhere between the 50×50 and 100×100 cases would have an estimated $\|\text{error}\|_\infty(\Omega)$ comparable with the $\|\text{error}\|_\infty(\Gamma_{10})$, for a CD solution, on the order of 10^{-4}.

Thus, a finite difference system of equations for roughly 5,000 unknowns has to be solved in order to obtain the same accuracy as that of a CD solution which requires solving three systems, consisting of 19, 27 and 77 equations respectively.

In conclusion, we find Cell Discretization to be a convenient and accurate method for solving elliptic partial differential equations. It can handle domains of general shape, general basis and weight sets, and discontinuous coefficients and source terms in p.d.e.'s Work is in progress to extend this method to nonself-adjoint, time-dependent and nonlinear problems.

REFERENCES

[1] J. GREENSTADT, *The cell discretization algorithm for elliptic partial differential equations*, this Journal, 3 (1982), pp. 261–288.

[2] ———, *The application of cell discretization to nuclear reactor problems*, Nuc. Sci. and Eng., 82 (1982), pp. 78–95.

[3] E. JAHNKE AND F. EMDE, *Tables of Functions*, Dover, New York, 1945.

[4] I. GRADSHTEYN AND I. RYZHIK, *Tables of Integrals, Series and Products*, Academic Press, New York, 1980.

[5] E. BUTKOV, *Mathematical Physics*, Addison-Wesley, Reading, MA, 1968.

[6] L. HAYES et al., *The treatment of sources and sinks in steady-state reservoir engineering simulations*, in Advances in Computer Methods for Partial Differential Equations, R. Vichnevetsky, ed., IMACS, Bethlehem, PA, 1977, pp. 301–306.

SIAM J. SCI. STAT. COMPUT.
Vol. 7, No. 3, July 1986

016

HIGH ORDER THREE POINT SCHEMES FOR BOUNDARY VALUE PROBLEMS I. LINEAR PROBLEMS*

AVI LIN†

Abstract. A general strategy for formulating numerical schemes for solving boundary value problems (BVP) and initial BVP (IBVP) is presented. Using this strategy it is possible to formulate three point numerical schemes for a BVP which are accurate to any order of the spatial interval. It can be shown that when this idea is applied to a general IBVP, schemes which are one step in time and three points in space can be obtained to any order of accuracy. Although this paper deals mainly with the linear case, this method can be extended to any nonlinear IBVP. One of the important features of this approach is that usually it has some free parameters and therefore the numerical scheme can be adjusted or tuned according to some additional requirements or restrictions. Some new schemes are derived and applied to linear equations, with different types of boundary conditions.

Key words. high order schemes, numerical solution of boundary value problems

AMS(MOS) subject classifications. 03C95, 03C20, 03E55, 03C55, 03C30

1. Introduction. This is the first paper in a series of two, describing a new strategy for deriving numerical schemes for solving two point boundary value problems (BVP). Many numerical methods have been proposed to solve BVPs. Today it can be said that most of these methods have shown good performance over a wide range of problems. Of course there are still many BVPs which are difficult to solve numerically (especially nonlinear cases) a difficulty which is usually associated with the (proofs of) uniqueness and existence of the solution. It should be noted that many of the methods were formulated primarily for iteratively solving (by lines, say) elliptic partial differential equations.

Roughly speaking, apart from initial value (shooting) methods [18], there are two classes of schemes for solving BVPs that appear frequently in the literature: the first class contains methods which are based on finite different (FD) schemes, and the second class contains methods which are based on the finite element (FE) approach. Usually the FE methods depend on the Ritz or Galerkin criterion with some specific shapes for the local basis functions [5]. These standard FE methods were recently improved by some other techniques, like the H^{-1} technique, the hybrid and the mixed techniques. However, these methods are very slow in convergence although now they are closer to the FD methods [15]. Many of the comparisons between the FE and the FD methods have shown superiority of the FE methods over the FD methods [5]:

- the standard FD method is analogous to the FE method with linear basis functions for the Dirichlet problem.
- unlike the FD method the FE method exactly satisfies boundary conditions of the Neumann type.

In the spirit of these arguments, which have been found to be incorrect, a general strategy for deriving a three point FD approximation for a BVP to any accuracy level is presented in this paper. The FD approach for numerically solving BVP's is well established and widely used. In a recent survey [1], for example, it is emphasized that the most efficient numerical schemes for solving a general BVP are based on either finite differences or (multiple) shooting schemes. Another commonly used approach

* Received by the editors May 17, 1983, and in revised form April 15, 1985. This research was supported by the Technion Research Foundation under grant 121-606 (1982).

† Computer Science Department, Technion - Israel Institute of Technology, Haifa, Israel, 32000.

is the collocation method [2]: it was shown, for example, that the technique based on the Lobatto quadrature points is very accurate and highly efficient. However, it was argued [1] that using the Keller box scheme with Richardson extrapolation is superior to the collocation method. A numerical method based on cubic splines was also suggested [3], giving a high numerical accuracy with reasonable storage requirements. Interval methods have been used with standard versions of finite difference techniques to solve ill-conditioned BVP's [4].

The present paper suggests a new high order FD numerical scheme for a BVP. It is well known that many efforts have been made already to get high order solutions for a general BVP and some of them will be mentioned in the present paper. The COLSYS Fortran code [15] uses piecewise polynomials (B-splines) and collocation (at Gaussian points) for discretizing the BVP. Although this method is of the FE type, it is worth mentioning here since by its special implementation it is very competitive with the FD methods in terms of speed and stability. The NONREF code [16] is in principle also a FE scheme based on the Galerkin approach. One of the schemes most closely related to the present work is the PSVAV3 code [17]. Here the BVP is given by a system of first order ODEs, and is solved by employing trapezoidal or midpoint rules for the derivatives while utilizing a deferred correction based on the truncation errors to get high order accurate solutions. The theory behind this code goes back to 1967–68 where deferred corrections were first used and proved to be more effective than Richardson extrapolation for getting high order accurate solutions. Later this scheme was improved by incorporating a dynamic nonuniform mesh adaptation procedure, and automatic variable order and variable step size selection which are monitored by the equidistribution of the norm of the local truncation error [13], [17]. Another related scheme is the GAP method [19] where analytic formulas have been used for the second order derivatives with improved accuracy by the use of Hermite polynomials. An interesting approach is the improved accuracy scheme of [10], that also deals with first order ODEs, where the improvement is based on an interpolation scheme for the function's values and its derivatives at points inside the interval under consideration. The scheme in [10] is based on a well-documented approach which is different than the present one. With the approach in [10] the accuracy of the schemes previously mentioned can be improved; the possibility of improving the present scheme will be discussed later. However the present approach is more general than [10] since solutions obtained with the present method can be accurate to any order. The general basis of the FD analogue for a BVP will be presented in the following section. Then highly accurate boundary condition approximations are derived and illustrated. Although only linear cases will be discussed here, the conclusions can be extended to the nonlinear general case, as will be shown in a subsequent paper [12]. Finally, this strategy is used to get numerical solutions for one-dimensional parabolic equation (initial BVP) in the framework of one step high order methods. Schemes which are second order in time and space are presented.

2. Numerical presentation of a BVP. The formulation of the present FD scheme for the BVP is motivated by the following principles: (i) the ability to solve on present computers directly and in a rapid manner certain banded systems of algebraic equations; (ii) in order to increase the stability of the scheme, one should keep as small as possible the number of grid points over which the FD approximation is spread. The numerical methods for solving BVP's that will be studied here are restricted to three point schemes. It is assumed that the BVP is to be solved over the one-dimensional domain Ω: ($x \in [L, R]$), with the boundary $\partial\Omega$: ($x = L, x = R$). Let $N+1$ be the number of discrete

points spread evenly over the domain Ω with $N>0$ and a spacing $h=(R-L)/N$, where $x_k=L+kh$, $k=0(1)N$. It should be noted that the present numerical method can be applied also to the case where h is not a constant [14]. According to the above motivation the FD equation at any inner point i can be written as:

$$A_i\Phi_{i-1}+B_i\Phi_i+C_i\Phi_{i+1}=D_i, \qquad i=1(1)N \tag{1}$$

where

$$\Phi_i=\Phi(x_i).$$

Now, the main problem is to find the most general family of BVP's which FD approximation at the node i is expressed by (1). This approach is quite different from the methods that have been mentioned previously. However this type of approach is not new. A similar approach can be found in Hamming [6, p. 395] where the general theory of predictor-corrector methods for solving initial value (IVP) ODEs is discussed. The main advantage of this approach for IVP is that the scheme contains enough free parameters that most of the accuracy requirements can be fulfilled minimizing the instabilities and roundoff errors. Let us consider the following general BVP

$$a(x)\Phi_{xx}=g(x,\Phi,\Phi_x)$$

with

$$\Phi_x=\frac{\partial\Phi}{\partial x}, \quad \Phi_{xx}=\frac{\partial^2\Phi}{\partial x^2}, \quad \Phi_{xxx}=\frac{\partial^3\Phi}{\partial x^3}, \quad \Phi_{xxxx}=\frac{\partial^4\Phi}{\partial x^4}, \quad \Phi_{x^n}=\frac{\partial^n\Phi}{\partial x^n}$$

subject, for simplicity, to the following Dirichlet boundary conditions

$$\Phi(L)=\Phi_L, \qquad \Phi(R)=\Phi_R$$

and where g is a nonlinear functional. Let us define also

$$b=-\frac{\partial g}{\partial\Phi_x}, \qquad e=-\frac{\partial g}{\partial\Phi}.$$

It is assumed in addition that g, b, e, a are continuous over Ω, and that this BVP has a unique solution (e.g. $e/a<0$ and $|b/a|$ is bounded [7]). Since nonlinear problems will be of interest only later [12] the nonlinearity of (1) is treated for simplicity by Newton's method: By using an artificial time coordinate, the following iterative technique is obtained:

$$a(x)\Phi^n_{xx}+b(x,\Phi^{n-1},\Phi^{n-1}_x)\Phi^n_x+e(x,\Phi^{n-1},\Phi^{n-1}_x)\Phi^n=d(x,\Phi^{n-1},\Phi^{n-1}_x).$$

The iteration procedure begins by assuming some function, $\Phi^0(x)$, and with the last equation a sequence $\Phi^1(x)$, $\Phi^2(x)$, etc. is produced until some convergence criterion is achieved. A more precise study will be presented in the subsequent paper [12]. Hence, generally, at every iteration stage, the following linear FD model of (1) has to be solved:

$$a(x)\Phi_{xx}+b(x)\Phi_x+e(x)\Phi=d(x). \tag{2}$$

Let us now establish the relations between the nodal coefficients set $M=[A_i, B_i, C_i, D_i]$ of (1) and the BVP coefficients set $m=[a(x), b(x), e(x), d(x)]$ of (2). The optimal situation occurs when the solutions of (1) and (2) are the same at the grid points. The motivation is to find under what conditions or circumstances this goal can be achieved.

Trivially, it can be shown that the difference between the solutions of (1) and the solution of (2) are functions of h. Therefore by defining

$$\alpha_i = \frac{(A_i + C_i)h^2}{2}, \quad \beta_i = (C_i - A_i)h, \quad \gamma_i = A_i + B_i + C_i \tag{3a,b,c}$$

and using Taylor expansion for $\Phi_{i\pm1}$ around the point x_i, (1) can be written at any point i as follows:

$$\gamma_i\Phi + \beta_i\Phi_x + \alpha_i\Phi_{xx} + \frac{h^2}{6}\beta_i\Phi_{xxx} + \frac{h^2}{12}\alpha_i\Phi_{xxxx} + \cdots = \delta_i. \tag{4}$$

Comparing (4) with (1) it can be seen that the relation between the set M and the set m is through a perturbation series in the parameter h. Let us assume the following asymptotic expansion at any discrete point i:

$$\alpha = a_0 + a_1h + a_2h^2 + \cdots, \qquad \beta = b_0 + b_1h + b_2h^2 + \cdots, \tag{5a,b}$$

$$\gamma = e_0 + e_1h + e_2h^2 + \cdots, \qquad \delta = d_0 + d_1h + d_2h^2 + \cdots. \tag{5c,d}$$

Substitution of (5) into (4) gives for every inner point $1 \leqq i \leqq N-1$ the following equation:

$$\begin{aligned}
&(a_0\Phi_{xx} + b_0\Phi_x + e_0\Phi) + h(a_1\Phi_{xx} + b_1\Phi_x + e_1\Phi) \\
&\quad + h^2\left(\frac{a_0}{12}\Phi_{xxxx} + \frac{b_0}{6}\Phi_{xxx} + a_2\Phi_{xx} + b_2\Phi_x + e_2\Phi\right) \\
&\quad + h^3\left(\frac{a_1}{12}\Phi_{xxxx} + \frac{b_1}{6}\Phi_{xxx} + a_3\Phi_{xx} + b_3\Phi_x + e_3\Phi\right) \\
&\quad + h^4\left(\frac{a_2}{12}\Phi_{xxxx} + \frac{b_2}{6}\Phi_{xxx} + a_4\Phi_{xx} + b_4\Phi_x + e_4\Phi\right) + \cdots \\
&\qquad = d_0 + hd_1 + h^2d_2 + \cdots.
\end{aligned} \tag{6}$$

This equation is the same as (1) when an infinite number of terms are considered in the infinite series on both sides of (6). In principle, numerical schemes are formulated by truncating these series. Hereafter several schemes for the set M will be derived, using (6) and the BVP to be solved (equation (2)).

2.1. Second order models. Second order models are the simplest numerical approximations to (2). These approximations are obtained from (16) by truncating all the terms of the order higher than $O(h^2)$ [13]. By comparing (2) with the reduced (6) the following relations are obtained:

$$\frac{a_p}{a} = \frac{b_p}{b} = \frac{e_p}{e} = \frac{d_p}{d} = \lambda_p \quad \text{with } p = 0, 1 \text{ and } \lambda_0 \neq 0. \tag{7}$$

Thus, using (3) the following equation holds for every inner point i:

$$C_i = \left(\frac{a}{h^2} + \frac{b}{2h}\right) + \frac{a_2 + b_2h/2}{G} + O(h^2), \tag{8a}$$

$$A_i = \left(\frac{a}{h^2} - \frac{b}{2h}\right) + \frac{a_2 - b_2h/2}{G} + O(h^2), \tag{8b}$$

$$B_i = \left(e - 2\frac{a}{h^2}\right) + \frac{e_2 h^2 - 2a_2}{G} + O(h^2), \tag{8c}$$

$$D_i = d + \frac{d_2 h^2}{G}, \tag{8d}$$

where

$$G = \lambda_0 + h\lambda_1, \qquad |\lambda_0| > O(h). \tag{8e}$$

Equations (7)-(8) describe a general family of schemes which are second order accurate. It is interesting to note that if the first derivative is modeled by the classical central differences

$$(b\Phi_x)_i = \frac{b(x_i)}{2h}(\Phi_{i+1} - \Phi_{i-1}) \tag{9}$$

and, similarly, the second derivative is modeled by

$$(a\Phi_{xx})_i = \frac{a(x_i)}{h^2}(\Phi_{i+1} - 2\Phi_i + \Phi_{i-1}), \tag{10}$$

then equations (8)-(10) give the following FD model for Φ at the grid point i:

$$(e\Phi)_i \approx \frac{a_2 + b_2 h/2}{G}\Phi_{i+1} + \left(e - \frac{2a_2 - e_2 h^2}{G}\right)\Phi_i + \frac{a_2 - b_2 h/2}{G}\Phi_{i-1}. \tag{11}$$

This four parameter scheme is usually used with

$$a_2 = b_2 = e_2 = 0 \tag{12}$$

to give the standard FD approximation for Φ_i:

$$(e\Phi)_i \approx e(x_i)\Phi_i.$$

Another possibility for choosing the free parameters is similar to (12), but with

$$a_2 \neq 0. \tag{13}$$

Thus, assuming again the FD approximations (9) and (10) the following FD model is obtained:

$$(e\Phi)_i \approx e(x_i)[\eta\Phi_{i+1} + (1 - 2\eta)\Phi_i + \eta\Phi_{i-1}] \tag{14}$$

where

$$\eta = \frac{a_2}{eG}. \tag{15}$$

In this special one parameter second order scheme, $|\eta|$ can be assigned any value which is less than $O(1/h)$. Thus, it is possible to impose some additional requirements and properties which the standard scheme, (12), does not have. For example it is possible to find a parameter η such that the absolute value of the truncation error of the scheme described by (13) will not be larger than that of the scheme described by (10). Or the free parameter can be used to reduce the truncation error still further. In this case, the truncation error of the scheme defined by (13) is

$$T = -h^2\left[\frac{b}{a}\left(a_2 - \frac{b^2}{12a}\right)\Phi_x + \frac{e}{a}\left(a_2 - \frac{b^2}{12a} - \frac{e}{12}\right)\Phi + \left(\frac{d_{xx}}{12} + \frac{bd_x}{12} + d\left(a_2 - \frac{b^2}{12a} - \frac{e}{12}\right)\right)\right], \tag{16a}$$

and that of (12) is

$$T=\frac{h^2}{12a}\left[\frac{b^3}{a}\Phi_x+e\left(\frac{b^2}{a}+e\right)\Phi\right]. \tag{16b}$$

In order for the two truncation errors to be as close to each other as possible, the following equation has to be fulfilled:

$$\eta=\max\left(\frac{b^2}{6ae}+\frac{1}{6},\frac{b^2}{6ae}\right).$$

Thus, for example, for the case where $b=0$ we have

$$\eta=\tfrac{1}{6}. \tag{17}$$

The same result can also be obtained with the FE method by assuming linear basis functions (cf. [5, p. 259]). It can be shown that the truncation errors of the schemes described by (16) and (17) have the same absolute values with opposite signs (this can be verified also numerically by [5, Table 5.5]). This result also supports another fact that was previously mentioned in [5], namely that the (converged) errors of the FE method are lower bounded, and those of the FD method are upper bounded when compared to the exact solution. The minimum truncation error ($T=0$) is obtained at any point i for the following value of η:

$$\eta=\frac{1}{12}\left(1+\frac{b^2}{ae}\right)+\frac{1}{12}\left[\frac{b\Phi_x}{a\Phi_{xx}}-(d_{xx}+bd_x)\right]. \tag{18}$$

Thus, by definition, this is a fourth order scheme and it presents one subfamily of the schemes of this order that will be discussed shortly. It should be noted that for the $b=d=0$ special case, the FE method with the quadratic shape functions also suggests that $\eta=\frac{1}{12}$.

2.2. Fourth order models. Most of the recent numerical methods for BVP propose fourth order schemes. The PASVA3 method [17] (which is based on the IDC method [13]), one of the best known of these procedures, is most similar to the present one. With this method the second order accurate solution is used to approximate the truncation error, which in turn is used as a deferred correction (source term) to get again a higher (4th) order solution. However unlike the present method, some instabilities and wiggles were detected for several BVPs when the deferred correction method was used to get high order solutions. A possible explanation for this may be the fact that as the accuracy demands increase the correction terms are spread over a larger number of grid points; nevertheless, this observation may be a subject for a more careful investigation. On the other hand the present numerical scheme is limited only to three grid points, and therefore similar instabilities may be eliminated. The present fourth order method is obtained again from (6) by asking the $O(h^3)$ and $O(h^4)$ terms' coefficients to have the following relations:

$$\frac{a_p}{12}\Phi_{xxxx}+\frac{b_p}{6}\Phi_{xxx}+a_{p+2}\Phi_{xx}+b_{p+2}\Phi_x+e_{p+2}\Phi=d_{p+2}\quad\text{with } p=0,1. \tag{19}$$

It can be shown that if the variation of the m group's coefficients, a, b, e and d with x is slower than that of Φ, then (19) can be fulfilled by the following relations at any

point i:

$$(20a,b,c) \qquad a_2=\lambda_2 a_0+\frac{1}{12}\left(e_0+\frac{b_0^2}{a_0}\right), \qquad b_2=\lambda_2 b_0+\frac{b_0 e_0}{12a_0}, \quad e_2=\lambda_2 e_0,$$

$$(20d,e,f) \qquad a_3=\lambda_3 a_1+\frac{1}{12}\left(e_1+\frac{b_1^2}{a_1}\right), \qquad b_3=\lambda_3 b_1+\frac{b_1 e_1}{12a_1}, \quad e_3=\lambda_3 e_1,$$

with $|\lambda_2|>O(h^2)$.

Equations (20) can be substituted back into (3) to formulate the tridiagonal matrix coefficients:

$$(21a) \qquad C_i=K\left(\frac{a}{h^2}+\frac{b}{2h}\right)+\frac{1}{12}\left(e+\frac{b^2}{a}+h\frac{be}{2a}\right),$$

$$(21b) \qquad A_i=K\left(\frac{a}{h^2}-\frac{b}{2h}\right)+\frac{1}{12}\left(e+\frac{b^2}{a}-h\frac{be}{2a}\right),$$

$$(21c) \qquad B_i=K\left(e-\frac{2a}{h^2}\right)-\frac{1}{6}\left(e+\frac{b^2}{a}\right),$$

where

$$K=1+K_2h^2, \qquad K_2=\frac{\lambda_0\lambda_2+\lambda_1\lambda_3 h}{G}.$$

If (9) and (10) are considered as a reasonable model for the first and second derivatives, then according to (21), Φ is approximated at the grid point i as follows:

$$(22) \qquad \begin{aligned}(e\Phi)_i \approx{}& \Phi_{i+1}\left[\frac{1}{12}\left(e+\frac{b^2}{a}\right)+h\frac{be}{24a}+\alpha K_2h^2\right]+\Phi_i\left[Ke-2aK_2h^2-\frac{1}{6}\left(e+\frac{b^2}{a}\right)\right]\\ &+\Phi_{i-1}\left[\frac{1}{12}\left(e+\frac{b^2}{a}\right)-h\frac{be}{24a}+aK_2h^2\right].\end{aligned}$$

This is a one parameter model for the fourth order numerical approximation for the BVP (eq. (2)) and can also be tuned. The truncation error of this scheme can be found by assuming as before that the variation with x of the m group's coefficients is slow compared to that of the variable Φ:

$$(23) \qquad \begin{aligned}T=-\frac{h^4}{144}\Bigg\{&\frac{be}{a}\left(12K_2+\frac{e}{a}-\frac{b^2}{a^2}\right)\Phi_x\\ &+\left[\frac{b^2}{a^2}\left(e-\frac{b^2}{a}\right)+\left(\frac{e}{a}+12K_2\right)\left(e+\frac{b^2}{a}\right)\right]\Phi_{xx}\Bigg\}.\end{aligned}$$

The simple situation is when $K_2=0$: then for the case where $b=0$ as in (17), the term $(e\Phi)_i$ is modeled by an equation similar to (22) but with

$$(24) \qquad \eta=\tfrac{1}{12}$$

(this result is embedded implicitly in (18)). It is not trivial to get this fourth order model (22) with the FE method. The present fourth order approximation is tested in the following example.

Example 1. Let us consider the following linear BVP:

$$(25) \qquad \Phi_{xx}+\Phi_x-2\Phi=2$$

with $L=0$ and $R=1$. The following boundary conditions are imposed:

$$\Phi(x=0)=P+Q-1, \qquad \Phi(x=1)=Pe^{1}+Qe^{-2}-1 \tag{26}$$

where e is the natural exponent and P, Q are any two constants.

The FD approximation at every point i is (see (21))

$$C=\frac{3}{2}N^2K-\frac{1}{12}\left(1+\frac{2}{N}\right), \tag{27a}$$

$$A=\frac{1}{2}N^2K-\frac{1}{12}\left(1-\frac{2}{N}\right), \tag{27b}$$

$$B=2(1-N^2)K+\frac{1}{6}, \tag{27c}$$

where N is the number of equal intervals spread over Ω. The results for the second and fourth order schemes can be verified in Table 1. It can be seen that (27) represent actually a fourth order scheme.

TABLE 1

Second, fourth and sixth order schemes' errors for example 1, *at* $x=0.5$ *solved with equal spread mesh.*

Scheme	$N=26$	$N=51$	$N=101$	$N=201$
2nd order	0.69580 (−4)	0.17401 (−4)	0.43585 (−5)	0.10903 (−5)
PASVA3 [17] 2nd order	0.19190 (−3)	0.48513 (−4)	0.12204 (−4)	0.30572 (−5)
4th order	−0.82340 (−8)	−0.51991 (−9)	−0.31547 (−10)	0.19706 (−11)
PASVA3 [17] 4th order	0.55432 (−6)	0.42657 (−7)	0.35546 (−8)	0.29667 (−9)
6th order	−0.15838 (−8)	−0.24759 (−10)	−0.41197 (−12)	0.64931 (−14)
4th order boundary condition	0.75417 (−8)	0.47148 (−9)	0.29815 (−10)	0.18933 (−11)

It is important to note that the errors vary linearly with K_2, approximately like $5\ 10^{-6}\ K_2$. This scheme can be compared to the PASVA3 scheme [17] for which results are also presented in this table. The second order solutions of this scheme were obtained by taking the forward differences for Φ_x in (25), with the correction term $(N/2)\Phi_{xx}$. The derivative in the correction term was approximated by the standard central difference scheme (10). It can be seen that both schemes produce second order solutions. However, the solutions obtained with the present scheme have smaller errors and are obtained while solving the system only once. In any event, for this case, both systems use three points schemes and the behavior of the errors is similar. Comparing the fourth order solutions, it can be seen that the present scheme produces errors which are much smaller than those of the PASVA3 scheme. It can be observed also that the latter is not exactly fourth order $[\sim O(h^{3.7})]$; this may be due to the influence of spreading the FD approximation of the corrections over many grid points.

2.3. Sixth order models. The sixth order model can be formulated in a similar way to that of the second and the fourth order models. It is necessary to fulfill (7) with the first and second terms of the series presented in (5), and to fulfill (20) with the third and fourth terms of this series. After this is done, the following $O(h^4)$ and

$O(h^5)$ equations are obtained:

$$\frac{a_p}{360}\Phi_{x^6}+\frac{b_p}{120}\Phi_{x^5}+\frac{a_{p+2}}{12}\Phi_{xxxx}+\frac{b_{p+2}}{6}\Phi_{xxx} + a_{p+4}\Phi_{xx}+b_{p+4}\Phi_x+e_{p+4}\Phi=d_{p+4}, \qquad p=1,2. \tag{28}$$

After reducing (28), one can get the following relations:

$$a_4=\lambda_4 a+\frac{\lambda_0}{720a}\left(3e^2-\frac{b^4}{a^2}+4\frac{b^2e}{a}\right)+\frac{\lambda_2}{12}\left(e+\frac{b^2}{a}\right), \tag{29a}$$

$$b_4=\lambda_4 b+\frac{\lambda_0 eb}{720a^2}\left(3e^2-\frac{b^2}{a}\right)+\lambda_2\frac{eb}{12a}, \tag{29b}$$

$$e_4=\lambda_4 e, \tag{29c}$$

and the tridiagonal matrix coefficients at any inner point i are

$$C_i=L\left(\frac{a}{h^2}+\frac{b}{2h}\right)+\frac{Kh^2}{12}\left(e+\frac{b^2}{a}+h\frac{be}{2a}\right) +\frac{h^2}{60}\left(\frac{e^2}{4a}-\frac{b^4}{12a^3}+\frac{b^2e}{3a^2}\right)+\frac{h^3eb}{480a^2}\left(e-\frac{b^2}{3a}\right), \tag{30a}$$

$$A_i=L\left(\frac{a}{h^2}-\frac{b}{2h}\right)+\frac{Kh^2}{12}\left(e+\frac{b^2}{a}-h\frac{be}{2a}\right) +\frac{h^2}{60}\left(\frac{e^2}{4a}-\frac{b^4}{12a^3}+\frac{b^2e}{3a^2}\right)-\frac{h^3eb}{480a^2}\left(e-\frac{b^2}{3a}\right), \tag{30b}$$

$$B_i=L\left(e-\frac{2a}{h^2}\right)-K\left(e+\frac{b^2}{a}\right)-\frac{h^2}{360a}\left(3e^2-\frac{b^4}{a^2}+4eb^2\right), \tag{30c}$$

where

$$L=K+K_4h^4 \quad \text{and} \quad K_4=\frac{\lambda_0\lambda_4+h\lambda_1\lambda_5}{G}.$$

This sixth order scheme has been applied to the BVP in Example 1, and some of the results are given in Table 1. It can be seen that the scheme described in (30) is of the order $O(h^6)$. Another simple example is that which is given in [5, pp. 259]. With $K_2=K_4=0$, the FD coefficients are as follows:

$$C_i=\frac{a}{h^2}+\frac{b}{2h}+\frac{e}{12}+\frac{(eh)^2}{240},$$

$$A_i=\frac{a}{h^2}-\frac{b}{2h}+\frac{e}{12}+\frac{(eh)^2}{240},$$

$$B_i=-\frac{2a}{h^2}+\frac{5e}{6}-\frac{(eh)^2}{120}.$$

With the approximations given by (9) and (10) it can be shown that the sixth order model for the term $(e\Phi)$ is similar to that given by (14) with

$$\eta=\frac{1}{12}\left(1+\frac{eh^2}{20}\right). \tag{31}$$

Although it is quite easy to generate this scheme from finite difference considerations as it was shown here, it is extremely difficult to find the appropriate basis functions for a FE method to give the results presented in (30) and (31).

3. Boundary conditions. The accuracy of the BVP's solutions depends also very much on the numerical approximations to the boundary conditions.

Let us consider here only linear boundary conditions. If "0" is a grid point on $\partial\Omega$ and "1" is the closest grid point to "0", then the entries corresponding to the boundary condition in the tridiagonal algebraic system have to take the following form:

$$B_0\Phi_0 + C_0\Phi_1 = D_0. \tag{32}$$

This algebraic equation simulates the general boundary condition of the form:

$$b_B\Phi_x + e_B\Phi = d_B. \tag{33a}$$

As for the general case (4), (32) can be manipulated at the point $i=0$ in a similar way to (3) to give

$$\alpha\Phi_{xx} + \beta\Phi_x + \gamma\Phi + \frac{h^2}{6}\left(\beta\Phi_{xxx} + \frac{\alpha}{2}\Phi_{xxxx}\right) + \cdots = D \tag{33b}$$

where

$$\alpha = \frac{C_0h^2}{2}, \quad \beta = C_0h, \quad \gamma = C_0 + B_0. \tag{33c,d,e}$$

In order to establish the FD approximation, (32) and (33b) have to be compared. Thus the next step is to assume a perturbation series for α, β and γ as in (5), where the accuracy of the FD models depends on the truncated terms in these series. For consistency, let us assume the following relation:

$$b(x_i) = \frac{2a(x_i)}{h}$$

and then (6) does also apply.

3.1. Second order models. The second order approximation for the boundary condition is found in a similar way to that of a regular inner point as is derived in § 2.1. Since it is desired that (6) will approximate (32) to the second order then

$$a_p\Phi_{xx} + b_p\Phi_x + e_p\Phi = d_p, \qquad p = 0, 1;$$

or

$$b_0 = \mu_0 H\frac{2a}{h}, \quad e_0 = \mu_0(e_B + He), \quad d_0 = \mu_0(d_B + Hd), \tag{34a,b,c}$$

$$b_1 = \mu_1 H\frac{2a}{h}, \quad e_1 = \mu_1(e_B + He), \quad d_1 = \mu_1(d_B + He), \tag{34d,e,f}$$

where μ_0 and μ_1 are two parameters, and

$$H = \frac{hb_B}{2a - hb}. \tag{34g}$$

Thus

$$C_0 = H\frac{2a}{h^2}, \quad B_0 = e_B + H\left(e - \frac{2a}{h^2}\right), \quad D_0 = d_B + Hd. \tag{35a,b,c}$$

The term $\mu_0 + h\mu_1$ was factored out from (35). The main difference between the boundary equations (35) and the equations governing the variation of Φ over Ω, (8),

is that the scheme at the boundary does not have any free parameter to match additional demands from the governing system of equations. Thus, it can be expected that the boundary conditions will reduce the capability to produce additional schemes of the same order. This limitation is not so dramatic, since the free parameter may be varied with x (or with the index i), to get its imposed (zero) values at the boundary.

3.2. Fourth order models. In order to recover the fourth order approximation, similar equations to those presented by (19) have to be formulated. Choosing μ_2 as the free proportional parameter, the following relations have to be fulfilled:

$$b_2 = \mu_2 H \frac{2a}{h} + \mu_0 H^2 \frac{2a}{b_B h^2}\left[\frac{1}{3}E\left(e - \frac{b^2}{a}\right) - \frac{hbe}{a}\right], \tag{36a}$$

$$e_2 = \mu_2 e_B + \frac{e}{2a}\left[hb_2 - \frac{b_0}{3}\left(\frac{e}{4a} + \frac{b}{a}E\right)\right], \tag{36b}$$

$$d_2 = \mu_2 d_B + \frac{d}{2a}\left[hb_2 - \frac{b_0}{3}\left(\frac{e}{4a} + \frac{b}{a}E\right)\right], \tag{36c}$$

where $E = 1 - hb_B/4a$; and with $\mu_2 \neq 0$. Thus

$$C_0 = \frac{b_2}{h}, \quad B_0 = e_2 - C_0, \quad D_0 = d_2. \tag{36d, e, f}$$

This approximation for the boundary conditions has one free parameter μ_2. The performance of this scheme is checked below.

Example 2. For checking the above fourth order scheme, let us consider the problem given in Example 1, with the following change in the boundary conditions:

$$\varphi_x + 2\varphi = 3P - 2 \quad \text{at } x = 0. \tag{37}$$

Table 1 summarizes some tests with the application of (36) to the boundary condition (37). The fourth order of the scheme was recovered when it was used with (27).

4. Numerical results. In this section, some problems that appear very often in the open literature are tested. The comparisons are done over a uniformly spaced mesh. The codes were programmed in FORTRAN as well as in PASCAL (for double check), and were run on the IBM-3081 and on the VAX-780. On the IBM machine, the programs have been run under the Q option (64 bit word) and a similar option have been taken when running them under the VAX machine. The errors are defined here as $\mathrm{MAX}_i\,|y_{\text{computed}} - y_{\text{exact}}|$.

Problem 1. The same problem as [20, Problem 2]:

$$y'' = Ky + d, \qquad y(0) = y(1) = 0$$

where $K = 400$, $d(t) = 200 + 2(100 + \pi^2)\cos(2\pi t)$. The exact solution is $y(t) = \cosh(20t - 10)/\cosh(10) - \cos^2(\pi t)$. The FD approximation at the grid point i is:

$$\frac{y_{i+1} - 2y_i + y_{i-1}}{h^2} - \left(1 + r_4\frac{kh^2}{12} + r_6\frac{k^2h^4}{360}\right)Ky$$
$$- \left(1 + r_4\frac{kh^2}{12} + r_6\frac{k^2h^4}{360}\right)d(t)$$
$$- r_4\frac{kh^2}{12}\left(1 + r_6\frac{kh^2}{30}\right)d''(t) - r_6\frac{h^4}{360}d^{(4)}(t) = 0$$

where the second order solutions are obtained for $r_4 = r_6 = 0$ while with $r_4 = 1$, $r_6 = 0$ and $r_4 = 1$, $r_6 = 0$ the respective fourth and sixth order solutions are recovered. The comparisons are given in Table 2. The results show that all methods produce more or less the same order of accuracy as they were designed for. However, the results of the present scheme are somewhat better than the others. In any event these results prove the fact that with the present scheme it is possible to get solutions which are very accurate.

TABLE 2
Maximum error comparisons for Problem 1.

Scheme	$N = 17$	$N = 33$	$N = 65$	$N = 129$
present, 2nd order	2.03 (−2)	5.55 (−3)	1.46 (−3)	3.67 (−4)
present, 4th order	9.80 (−4)	7.18 (−5)	4.81 (−6)	3.03 (−7)
present, 6th order	2.69 (−5)	4.98 (−7)	8.35 (−9)	1.32 (−10)
PASVA3 [17] 4th order	2.23 (−2)	1.43 (−3)	1.19 (−4)	9.99 (−6)
PASVA3 [17] 6th order	4.28 (−4)	8.39 (−6)	1.86 (−7)	4.04 (−9)
Imp. Acc. [10] 6th order	6.56 (−3)	1.19 (−4)	2.14 (−6)	3.91 (−8)

Problem 2. The same as [20, Problem 4]:

$$y'' = z, \quad z'' = d(t); \quad y(0) = y'(0) = y(1) = y'(1) = 0.$$

For $d(t) = e^t(t^4 + 14t^3 + 49t^2 + 32t - 12)$ the exact solution is $y(t) = t^2(1-t)^2 e^t$.

The FD approximation up to the fourth order at the mesh point i is

$$\frac{y_{i+1} - 2y_i + y_{i-1}}{h^2} - z = r_4 \frac{kh^2}{12}\left(d + r_6 \frac{h^2}{30} d''\right),$$

$$\frac{z_{i+1} - 2z_i + z_{i-1}}{h^2} - d = r_4 \frac{kh^2}{12}\left(d'' + r_6 \frac{h^2}{30} d^{(4)}\right).$$

The main difficulty is to get the appropriate boundary conditions for $z(t)$. Using the formulae in § 3, subject to the above boundary conditions for $y(t)$, one can get the following relation at the first grid point $i = 0$:

$$\frac{2y_1}{h^2} = \frac{2}{3} z_0 + \frac{1}{3} z_1 - r_4 \frac{h^2}{6}\left[\frac{d_0}{2} + r_6 \frac{h}{3}\left(d_0' + \frac{h}{4} d_0''\right)\right]$$

and a similar equation at $i = N$. Due to these boundary conditions it is necessary to solve simultaneously the two governing equations at every grid point. Here the block tridiagonal inversion procedure [14] is used. Comparisons of the maximum error are given in Table 3.

Problem 3. The same as [20, Problem 5]:

$$y'' = \beta(y - z); \qquad z'' = \alpha(z - y).$$

Here it was chosen also [20]: $s = 10$, $c = 10^{-3}$ and $\alpha = \beta = 2.5$. The FD approximation up to the fourth order is

$$\frac{y_{i+1} - 2y_i + y_{i-1}}{h^2} - K\beta(y - z) = 0,$$

$$\frac{z_{i+1} - 2z_i + z_{i-1}}{h^2} - K\alpha(z - y) = 0,$$

TABLE 3
Maximum error comparisons for Problem 2.

Scheme	$N=17$	$N=33$	$N=65$	$N=129$
present, 2nd order	5.52 (−2)	1.38 (−2)	3.46 (−3)	8.62 (−4)
present, 4th order	2.94 (−3)	1.84 (−4)	1.15 (−5)	7.19 (−7)
present, 6th order	9.93 (−4)	1.55 (−5)	2.43 (−7)	4.10 (−9)
PASVA3 [17] 4th order	7.70 (−3)	6.01 (−4)	4.64 (−5)	3.11 (−6)
PASVA3 [17] 6th order	2.11 (−3)	4.69 (−5)	1.07 (−6)	2.33 (−8)
Imp. Acc. [10] 6th order	4.41 (−3)	8.02 (−5)	1.46 (−6)	2.65 (−8)

where

$$K=1+r_4\frac{h^2}{12}(\alpha+\beta).$$

The boundary conditions are derived from the equations in § 3, to give

$$y_0=0,$$

$$y_N-y_{N-1}+\frac{h^3\beta c}{6}+\frac{h^2}{2}\beta K(y_N-z_N)=0,$$

$$z_0-z_1+\frac{h^3\alpha c}{6}+\frac{h^2}{2}\alpha K(z_0-y_0)=0,$$

$$z_N-z_{N-1}-hc-\frac{h^3\alpha c}{6}+\frac{h^2}{2}\alpha K(z_N-y_N)=0.$$

The maximum error is compared with Table 4. The block-tridiagonal procedure was used also here to get the solutions. When applying the deferred correction method with equal intervals, some wiggles are encountered as the values of α and β are increased. These wiggles are of the order of the maximum error. The possibility of applying the improved accuracy procedure [10] to the present scheme was also checked with this test problem. As it is presented in Table 4, it was found that there is not much of improvement by doing so. The reason for this may be due to the fact that both truncation errors are of the same order.

TABLE 4
Maximum error comparisons for Problem 3.

Scheme	$N=17$	$N=33$	$N=65$	$N=129$
present, 2nd order	6.11 (−2)	1.53 (−2)	3.82 (−3)	9.55 (−4)
present, 4th order	3.36 (−3)	2.10 (−4)	1.31 (−5)	8.19 (−7)
present + Imp. Acc. 4th order	2.12 (−3)	1.92 (−4)	1.02 (−5)	7.78 (−7)
PASVA3 [17] 4th order	5.62 (−2)	3.61 (−3)	2.31 (−4)	1.41 (−5)

5. Numerical presentation of a IBVP. The general concepts for getting high order numerical schemes for a BVP, that were established previously in the paper, can be

extended also to IBVP-initial boundary value problem, helping to solve them more accurately and in a stable manner.

Let us consider the following linear IBVP for $\Phi(x, t)$:

$$\frac{\partial \Phi}{\partial t} = a\frac{\partial^2 \Phi}{\partial x^2} + b\frac{\partial \Phi}{\partial x} + e\Phi - d \tag{38a}$$

with the boundary conditions

$$\Phi(x = L) = \Phi_L, \qquad \Phi(x = R) = \Phi_R. \tag{38b}$$

The Dirichlet boundary conditions were chosen for simplicity and without loss of generality. For solving (38), a uniformly spaced grid with an interval h is spread over Ω: $(L \leqq x \leqq R)$. The numerical solution will be obtained for the discrete time values t_k, $k = 1, 2, 3, \cdots$, beginning at $k = 0$ with the initial conditions for Φ:

$$\Phi(x, t_0) = \Phi_I(x). \tag{38c}$$

Let us denote the discrete values of Φ by $\Phi_i^n = \Phi(x_i, t_n)$, where $x_i = L + (i-1)h$. For the same reasons that were discussed previously, it is desired that the solution Φ^n at every time t_n will be obtained by a three (spatial) point's numerical scheme. In order to achieve high accuracy, other numerical methods use linear multistep techniques [11]. However, since one of the requirements here is that the numerical scheme will be as stable and simple as possible only one-step techniques are of interest here. Thus, the main assumption about the numerical solution of the IBVP at the nth time step is that it will uniquely depend on the solution from the $n-1$ time step:

$$A_i\Phi_{i-1}^n + B_i\Phi_i^n + C_i\Phi_{i+1}^n = S(\Phi^{n-1}), \tag{39}$$

where S is a linear operator acting on the solution at the $n-1$ time step. The coefficients A_i, B_i, C_i and operator S depend solely on the equation to be solved and not on the solution. The second assumption is that the operator S will be as similar as possible to the operator on the right hand side of (39). Thus it is assumed that

$$S(\Phi) = \bar{A}_i\Phi_{i-1} + \bar{B}_i\Phi_i + \bar{C}_i\Phi^{i+1}. \tag{40}$$

Following (4), (40) can be written at the point i also as follows:

$$\left[\alpha\Phi_{xx} + \beta\Phi_x + \gamma\Phi + \frac{h^2}{6}\left(\beta\Phi_{xxx} + \frac{\alpha}{2}\Phi_{xxxx}\right) + \cdots - d\right]^n - \left[\bar{\alpha}\Phi_{xx} + \bar{\beta}\Phi_x + \bar{\gamma}\Phi + \frac{h^2}{6}\left(\bar{\beta}\Phi_{xxx} + \frac{\bar{\alpha}}{2}\Phi_{xxxx}\right) + \cdots - d\right]^{n-1} = 0 \tag{41}$$

where the definition of the coefficients in the last equation is similar to the definitions given by (3). Using the last equation it is possible to derive expressions for these coefficients, subject to the requested scheme's order of accuracy. In order to bring the last equation to a similar form as that of the original IBVP, (38), the different functions at the time level n have to be expanded around the $n-1$ time level. Let us define

$$\Delta = t_n - t_{n-1}$$

and the time $\bar{t}$ as

$$\bar{t} = t_n + \bar{\vartheta}\Delta, \quad 0 \leqq \bar{\vartheta} \leqq 1, \quad \text{with } \vartheta = 1 - \bar{\vartheta}.$$

Now (41) can be rewritten around the time $\bar{t}$ as follows:

$$
\begin{aligned}
&(\alpha-\bar{\alpha})\Phi_{xx}+(\beta-\bar{\beta})\Phi_x+(\gamma-\bar{\gamma})\Phi+\frac{h^2}{6}(\beta-\bar{\beta})\Phi_{xxx}+\frac{h^2}{12}(\alpha-\bar{\alpha})\Phi_{xxxx}+\cdots \\
&\quad+\Delta\frac{\partial}{\partial t}\Big[(\vartheta\alpha+\bar{\vartheta}\bar{\alpha})\Phi_{xx}+(\vartheta\beta+\bar{\vartheta}\bar{\beta})\Phi_x+(\vartheta\gamma+\bar{\vartheta}\bar{\gamma})\Phi \\
&\qquad+\frac{h^2}{6}(\vartheta\beta+\bar{\vartheta}\bar{\beta})\Phi_{xxx}+\frac{h^2}{12}(\vartheta\alpha+\bar{\vartheta}\bar{\alpha})\Phi_{xxxx}+\cdots\Big] \\
&\quad+\frac{\Delta^2}{2}\frac{\partial^2}{\partial t^2}[(\vartheta^2\alpha-\bar{\vartheta}^2\bar{\alpha})\Phi_{xx}+\cdots]+\cdots=0
\end{aligned}
\tag{42}
$$

and it is clear that (42) presents, in general, an equation that should resemble (to certain order of h and Δ) the original (38a). It is reasonable to assume that the change with time of the coefficients appearing in that equation is slower than that of Φ. Therefore, eliminating the time derivatives from (42) using (38a), a new equation involving only x derivatives and powers of h and Δ is obtained. Hereafter, some numerical approximations which are based on (42) are derived and examined.

5.1. Second order approximations for the IBVP. The second order approximation is obtained by a suitable truncation of the infinite series in (42). Since this equation should be fulfilled exactly for every set of the parameters, the following system of equations govern the relations between the parameters to the order of h^2 and Δ^2:

$$\frac{\bar{\gamma}}{\gamma}=\frac{A}{\bar{A}}, \tag{43a}$$

$$\frac{\beta}{\gamma}=\frac{\bar{A}}{A}\bar{\beta}-\left(\frac{B}{A}+\frac{\bar{B}}{\bar{A}}\right), \tag{43b}$$

$$\frac{\bar{\beta}}{\gamma}=\frac{\beta}{\bar{A}}-\frac{C\bar{A}+\bar{C}A}{B\bar{A}+\bar{B}A}-\frac{A}{B\bar{A}+\bar{B}A}(A\alpha-\bar{A}\bar{\alpha}), \tag{43c}$$

$$
\begin{aligned}
&\left[B-\frac{A}{B\bar{A}+\bar{B}A}\left(\bar{A}-\bar{C}A-\frac{h^2}{6}A\right)\right]\frac{\alpha}{\gamma}+\left[\bar{B}+\frac{\bar{A}}{B\bar{A}+\bar{B}A}\left(\bar{A}-\bar{C}A-\frac{h^2}{6}A\right)\right]\frac{\bar{\alpha}}{\gamma} \\
&\quad=\Delta^2 ab\left(\bar{\vartheta}^2\frac{A}{\bar{A}}-\vartheta^2\right)+\left(\frac{B}{A}+\frac{\bar{B}}{\bar{A}}\right)\left(\frac{h^2}{6}+C\right) \\
&\qquad\cdot\left[\frac{B}{\bar{A}}-\frac{A}{\bar{A}}\frac{C\bar{A}+\bar{C}A}{B\bar{A}+\bar{B}A}\right]\left(\frac{\bar{A}}{A}-\bar{C}-\frac{h^2}{6}\right),
\end{aligned}
\tag{43d}
$$

$$
\begin{aligned}
&\left[\frac{h^2}{12}+C-\Delta Aab\frac{\bar{A}\vartheta^2-A\bar{\vartheta}^2}{B\bar{A}+\bar{B}A}\right]\frac{\bar{\alpha}}{\gamma}-\left[\frac{h^2}{12}+\bar{C}-\Delta^2\bar{A}ab\frac{\bar{A}\vartheta^2-A\bar{\vartheta}^2}{B\bar{A}+\bar{B}A}\right]\frac{\bar{\alpha}}{\gamma} \\
&\quad=\Delta^2\left[\frac{a}{2\bar{A}}(A\bar{A}^2-\bar{a}\vartheta^2)+b\vartheta^2\left(\frac{B}{A}+\frac{\bar{B}}{\bar{A}}\right)\right. \\
&\qquad\left.-\frac{b}{A\bar{A}}(A\bar{\vartheta}^2-\bar{A}\vartheta^2)\left(B-A\frac{C\bar{A}+\bar{C}A}{B\bar{A}+\bar{B}A}\right)\right],
\end{aligned}
\tag{43e}
$$

where

(44a,b) $$A=1+\Delta\vartheta e+\Delta^2\vartheta^2\frac{e^2}{2},\qquad B=\Delta\vartheta b(1+\Delta\vartheta e),$$

(44c,d) $$C=\Delta\vartheta\left[a+\frac{\Delta\vartheta}{2}(2ae+b^2)\right],\qquad \bar{A}=1-\Delta\bar{\vartheta}e+\Delta^2\bar{\vartheta}^2\frac{e^2}{2}$$

(44e,f) $$\bar{B}=\Delta\bar{\vartheta}b(1+\Delta\bar{\vartheta}e),\qquad \bar{C}=\Delta\bar{\vartheta}\left[a+\frac{\Delta\bar{\vartheta}}{2}(2ae+b^2)\right].$$

The system (43) has always a solution since the determinant D of this system has the following form:

$$D=(\bar{\vartheta}-\vartheta)\Bigg(1+\frac{h^2\Delta}{12}b-\frac{h^2\Delta^2}{12}eb-\Delta e\vartheta\bar{\vartheta}+\frac{\bar{A}-\bar{C}A-(h^2/6)A}{B\bar{A}+\bar{B}A}\left[\Delta a-\frac{\Delta}{2}e-\frac{\Delta^2}{2}b^2(\bar{\vartheta}^2+\vartheta\bar{\vartheta}+\vartheta^2)-\frac{\Delta^3}{2}eb\vartheta\bar{\vartheta}\right]\Bigg)$$

$$+\frac{h^2}{12}\frac{e}{b}\frac{\bar{A}-\bar{C}A-(h^2/6)A}{1+\Delta e(\vartheta-\bar{\vartheta}+\vartheta\bar{\vartheta})+\Delta^2/2\,e^2\vartheta\bar{\vartheta}}$$

and it can be verified that $D\neq 0$ for every ϑ in the range of $0\leqq\vartheta,\bar{\vartheta}\leqq 1$ with $a\neq 0$. Therefore, (43) and (44) define an infinite number of second order schemes for an IBVP, which may be obtained by changing the parameter ϑ between 0 to 1.

Let us formulate specifically some of these schemes:

(1) The Crank-Nicolson (CN) scheme [9], $\vartheta=\frac{1}{2}$. This is a well-known second order scheme and may be recovered from (43) very easily by assuming $\vartheta=0.5$:

(45a,b) $$\frac{\bar{\gamma}}{\gamma}=\frac{A}{\bar{A}},\qquad \frac{\bar{\beta}}{\gamma}=\frac{B}{\bar{A}},$$

(45c,d,e) $$\frac{\bar{\alpha}}{\gamma}=\frac{C}{\bar{A}},\qquad \frac{\alpha}{\gamma}=-\frac{\bar{C}}{\bar{A}},\qquad \frac{\beta}{\gamma}=-\frac{\bar{B}}{\bar{A}}.$$

(2) The backward implicit (BI) scheme, $\vartheta=1$. This scheme is called the BI scheme since the IBVP and the FD approximation are compared at the time t_{n-1}. The relations between α and β for $\vartheta=1$ are obtained from (43):

$$\frac{\alpha}{\gamma}\left[a(1+\Delta e)+\frac{\Delta}{2}b^2\right]+\frac{\beta}{\gamma}b\left(\Delta a+\frac{h^2}{12}\right)=-\frac{a}{2}\left[\Delta a-\frac{h^2}{6}(1+\Delta e)\right],$$

$$\frac{\alpha}{\gamma}b(1+\Delta e)+\frac{\beta}{\gamma}\left[a(1+\Delta e)+\frac{\Delta}{2}b^2\right]=-b\left[\Delta a-\frac{h^2}{6}(1+\Delta e)\right]$$

and the relations between the other parameters are

(46a,b) $$\frac{\bar{\gamma}}{\gamma}=1+\Delta e+\frac{(\Delta e)^2}{2},\qquad \frac{\alpha}{\gamma}\approx\frac{1}{2}\frac{h^2/6-\Delta a}{1+2\Delta e-(h^2/12)(b/a)^2},$$

(46c,d) $$\frac{\beta}{\gamma}\approx\frac{1}{2}\frac{b}{a}\frac{h^2/6-\Delta a}{1+2\Delta e-(h^2/12)(b/a)^2},\qquad \frac{\bar{\alpha}}{\gamma}\approx\frac{1}{2}\frac{h^2/6+\Delta a}{1+2\Delta e-(h^2/12)(b/a)^2},$$

(46e) $$\frac{\bar{\beta}}{\gamma}\approx\frac{1}{2}\frac{b}{a}\frac{h^2/6+\Delta a}{1+2\Delta e-(h^2/12)(b/a)^2}.$$

The case of $\vartheta = 0$ is very similar to that of $\vartheta = 1$. Since this scheme is created by comparing the IBVP to the FD approximation at the time t_n, it is called the forward implicit (FI) scheme, which is somewhat similar to the BI scheme and will not be derived here explicitly.

5.2. High order boundary conditions for the IBVP. Just as was done for the BVP, the boundary conditions for the IBVP should be given an appropriate FD approximation in order to be fitted generally to the accuracy of the scheme. Only the Neumann boundary conditions for the IBVP will be discussed here. Let us assume that at the point $i = 0$ the following boundary condition is imposed:

$$\frac{\partial \Phi}{\partial x} = f(t). \tag{47}$$

It is also assumed that the finite difference approximation at this point is:

$$B_i \Phi_i^n + C_i \Phi_{i+1}^n + D_i = \bar{B}_i \Phi_i^n + \bar{C}_i \Phi_{i+1}^n + \bar{D}_i.$$

Checking this scheme at the point $\bar{t}$, the following equation is obtained:

$$\begin{aligned} &\Phi\left[(\beta-\bar{\beta})-\left(\frac{eh^2}{2a}-\frac{ebh^3}{6a^2}\right)(C-\bar{C})+\cdots\right] \\ &+\Phi_t\left[\Delta(\beta\vartheta-\bar{\beta}\bar{\vartheta})-\Delta(C\vartheta-\bar{C}\bar{\vartheta})\left(\frac{eh^2}{2a}-\frac{ebh^3}{6a^2}\right)+\left(\frac{h^2}{2a}-\frac{bh^3}{6a^2}\right)(C-\bar{C})+\cdots\right] \\ &+\Phi_{tt}\left[\frac{\bar{\Delta}^2}{2}(\beta\vartheta^2-\bar{\beta}\bar{\vartheta}^2)\right. \\ &\left.-\frac{\Delta^2}{2}(C\vartheta^2-\bar{C}\bar{\vartheta}^2)\left(\frac{eh^2}{2a}-\frac{ebh^3}{6a^2}\right)+\left(\frac{h^2}{2a}-\frac{bh^3}{6a^2}\right)\Delta(\vartheta C+\bar{\vartheta}\bar{C})+\cdots\right]=0 \end{aligned} \tag{48}$$

where

$$\beta = B_i + C_i, \qquad \bar{\beta} = \bar{B}_i + \bar{C}_i,$$

and

$$\begin{aligned} D_i - \bar{D}_i = &\left(h-\frac{h^2 b}{2a}-\frac{h^3(e-b^2/a)}{6a}\right) \\ &\cdot[f(\bar{t})(C-\bar{C})+f_t(\bar{t})\Delta(\vartheta C+\bar{\vartheta}\bar{C})]+\frac{h^3(C-\bar{C})}{6a}+\cdots. \end{aligned}$$

For the CN scheme ($\vartheta = \frac{1}{2}$) this formulation is reduced to the following equation at the boundary point $i = 0$:

$$\frac{\Phi_0^n - \Phi_0^{n-1}}{\Delta} = \tfrac{1}{2}[\Xi(\Phi_0^n) + \Xi(\Phi_0^{n-1})]$$

where

$$\begin{aligned} \Xi(\Phi) = &\frac{e-2a/h^2-beh/3a}{1-bh/3a}\Phi_0+\frac{2a/h^2}{1-bh/3a}\Phi_1 \\ &+\frac{(b-2a/h^2-hb^2/3a-he/3a)f+bh/3af_t}{1-bh/3a}. \end{aligned}$$

Equation (48) can be applied also to the BI scheme ($\vartheta = 0$) with the following formula:

$$\frac{\Phi_0^n - \Phi_0^{n-1}}{\Delta} = \frac{\Delta}{2} b f_t^{n-1} + \left(1 + \frac{\Delta}{2} e\right) f^n + \left(1 + \frac{\Delta}{2} e\right) \Phi_0$$

$$+ \frac{a}{2}[(1 + \Delta e)\Phi_{xx}^{n-1} + \Phi_{xx}^n]$$

where

$$\Phi_{xx} = \frac{2}{h^2}(\Phi_1 - \Phi_0) - \frac{2}{h} f.$$

This approach can be extended to derive the FI formula at the boundary, by choosing $\vartheta = 0$.

5.3. Example. Let us consider the IBVP (38) with

$$b(x) = 2Pa(x), \quad e(x) = P^2 a(x) - Q, \quad d(x) = 0$$

and with the initial condition: $\Phi_I(x) = x\, e^{Px}$, and boundary conditions: $\Phi(L, t) = 0$, $\Phi(R, t) = e^{-(1+Qt)}$. This problem has the following exact solution for $L = 0$ and $R = 1$:

$$\Phi(x, t) = x\, e^{-(Px+Qt)}.$$

Tables 5 and 6 summarize the solutions for two different functions $a(x)$, which were obtained with the BI scheme. The second order in time and fourth order in space can be observed.

TABLE 5
Numerical solution's errors of the IBVP *of* § 5.3 *with* $a(x) = e^{-x}$ *at* $x = 0.5$.

Δt	$N = 26$	$N = 51$	$N = 101$	$N = 201$
0.02	0.673 (−7)	0.425 (−8)	0.268 (−9)	0.171 (−10)
0.04	0.268 (−6)	0.168 (−7)	0.109 (−8)	0.688 (−10)
0.08	0.105 (−5)	0.659 (−7)	0.413 (−8)	0.261 (−9)
0.16	0.401 (−5)	0.255 (−6)	0.161 (−7)	0.101 (−8)
0.32	0.159 (−4)	0.100 (−5)	0.627 (−7)	0.393 (−8)

TABLE 6
Numerical solution's errors of the IBVP *of* § 5.3 *with* $a(x) = \sin(x)$ *at* $x = 0.5$.

Δt	$N = 26$	$N = 51$	$N = 101$	$N = 201$
0.02	0.289 (−8)	0.188 (−9)	0.121 (−10)	0.763 (−12)
0.04	0.116 (−6)	0.737 (−9)	0.471 (−10)	0.305 (−11)
0.08	0.468 (−7)	0.310 (−8)	0.205 (−9)	0.129 (−10)
0.16	0.189 (−6)	0.122 (−7)	0.771 (−9)	0.490 (−10)
0.32	0.759 (−6)	0.485 (−7)	0.313 (−8)	0.208 (−9)

6. Conclusions. The present paper presents a general approach for deriving high order numerical schemes for solving BVP and IBVP. Most of the schemes contain enough free parameters so that it is possible to adjust the method according to some additional restrictions. Although most of the discussion was done for the linear case,

the same approach can be extended to the nonlinear case [12]. The high order accuracy of some typical numerical schemes was verified by several examples. The present approach is different and more general than that of [10]. More detailed comparisons will be made for the nonlinear version of this new strategy in a forthcoming paper [12].

REFERENCES

[1] H. B. KELLER (1975), *Numerical solution of boundary value problems for ordinary differential equations: survey and some recent results on difference methods*, in Numerical Solutions of Boundary Value Problems for O.D.E., A. K. Aziz, ed., Academic Press, New York, pp. 27–88.

[2] R. WEISS (1974), *The application of implicit Runge-Kutta and collocation methods to boundary value problems*, Math. Comp., 28, pp. 449–464.

[3] R. P. TEWARSON (1980), *On the use of splines for numerical solution of nonlinear two-point boundary value problems*, BIT, 20, pp. 223–232.

[4] L. FOX AND M. R. VALENCA (1980), *Some experiments with interval methods for two-point boundary value problems in ordinary differential equations*, BIT, 20, pp. 67–82.

[5] T. J. CHUNG (1978), *Finite Element Analysis in Fluid Dynamics*, McGraw-Hill, New York, pp. 103–151.

[6] R. W. HAMMING (1973), *Numerical Methods for Scientists and Engineers*, 2nd ed., McGraw-Hill, New York.

[7] H. B. KELLER (1968), *Numerical Methods for Two-Point Boundary Value Problems*, Blaisdell, Waltham, MA.

[8] B. V. NOUMEROV (1924), *A method of extrapolation of perturbation*, Mon. Not. R. Astr. Soc., 8, pp. 592–601.

[9] G. D. SMITH (1975), *Numerical Solution of Partial Differential Equations*, Oxford Univ. Press, Oxford.

[10] R. P. TEWARSON AND S. GUPTA (1982), *Improving the accuracy of finite difference methods for solving boundary value ordinary differential equations*, BIT, 22, pp. 353–360.

[11] M. R. BEAM AND R. F. WARMING (1980), *Alternating direction implicit method for parabolic equations with mixed derivatives*, this Journal, 1 (1), pp. 131–159.

[12] A. LIN (1985), *High order three points schemes for boundary value problems:* II, *Nonlinear problems*, J. Comput. Appl. Math., in press.

[13] H. B. KELLER AND V. PEREYRA (1979), *Difference methods and deferred corrections for ordinary boundary value problems*, SIAM J. Numer. Anal., 16 (2), pp. 241–259.

[14] A. LIN (1985), *Studies of the general strategy for solving nonlinear boundary value problems*, submitted.

[15] U. ASCHER, J. CHRISTIANSEN AND R. D. RUSSELL (1981), *Collocation software for boundary value* ODEs, ACM Trans. Mathematical Software, 7, pp. 209–229.

[16] G. F. CAREY AND D. L. HUMPHREY (1979), *Finite element mesh refinement algorithm using element residual*, in Codes for Boundary Value Differential Equations, B. Childs, M. Scott, J. W. Daniel, F. Denman and P. Nelson, eds., Lecture Notes in Computer Science 76, Springer-Verlag, New York, pp. 243–249.

[17] V. PEREYRA (1979), PASVA3: *An adaptive finite difference* FORTRAN *program for first-order nonlinear ordinary boundary problems*, in Codes for Boundary Value Differential Equations, B. Childs, M. Scott, J. W. Daniel, F. Denman and P. Nelson, eds., Lecture Notes in Computer Science, 76, Springer-Verlag, New York, pp. 67–88.

[18] R. BULIRSCH, J. STOER AND P. DEUFLHARD (1983), *Numerical solution of nonlinear two-point boundary value problems* I, Numer. Math. Handbook Series Approximation.

[19] H. B. KELLER (1975), *Numerical solution of boundary value problems for ordinary differential equations: survey and some recent results of difference methods*, in Numerical Solution of Boundary Value Problems for Ordinary Differential Equations, A. K. Aziz, ed., Academic Press, New York, pp. 555–557.

[20] M. LENTINI AND V. PEREYRA (1974), *A variable order finite difference method for nonlinear multipoint boundary value problems*, Math. Comp., 28 (128), pp. 981–1003.

SIAM J. SCI. STAT. COMPUT.
Vol. 7, No. 3, July 1986

017

AUTOMATIC GKS STABILITY ANALYSIS*

MICHAEL THUNÉ†

Abstract. A new algorithm is presented for automatic stability investigation according to the theory of Gustafsson, Kreiss and Sundström. It is more efficient than approaches tried earlier, by taking advantage of the special structure of the system of algebraic equations whose solutions govern stability.

The software system IBSTAB implements this algorithm, as well as procedures for checking von Neumann- and *P*-stability. IBSTAB combines symbolic formula manipulation and numerical routines. Furthermore, a user oriented problem description language is included. The software system is entirely FORTRAN based, the symbol manipulation parts being written in LISP, using the FORTRAN coded LISP-F3 interpreter.

Results are presented from tests, in which IBSTAB was addressed to problems with known results. Also, a study is presented, in which IBSTAB was used as a tool in the development of stable numerical boundary conditions for a problem in fluid dynamics.

Key words. stability, hyperbolic initial-boundary value problems, mathematial software, symbolic formula manipulation

AMS(MOS) subject classifications. 65M10, 65N10, 65V05

1. Introduction. This paper concerns automatic investigation of stability of difference methods for hyperbolic initial-boundary value problems. We consider linear first order systems in one space dimension. The crucial parameter on which stability is generally dependent is $\lambda \equiv \Delta t/\Delta x$, i.e. the relation between the step sizes Δt in time and Δx in space. The stability investigation will amount to finding a critical λ-value λ_c, such that the difference method is stable if we choose $\lambda \in [0, \lambda_c]$ and unstable otherwise.

To make such an investigation, there are some possibilities that might seem plausible at first thought. One is to run the difference scheme for some values of λ and look at the solutions to see if those λ-values gave a stable approximation. Another possibility is to perform a von Neumann analysis (i.e. a Fourier analysis, which is only applicable on pure initial value problems and on problems with periodic boundary conditions), disregarding the boundary data, and use that stability limit also for the mixed problem. The third immediate idea is to write the difference equation on one step form

$$u^{n+1} = Qu^n$$

and then study the eigenvalues of the matrix Q (which contains the coefficients of the difference formulas). It seems reasonable to think that the difference approximation is stable for all λ-values that give a spectral radius of Q, which is less than or equal to one.

However, from reasons that are discussed in [29], none of those three ways is sufficient if one wishes to investigate the stability of mixed problems. For that kind of problem some special theory is needed. Consider first the *energy method* [24]. It has the drawback of being in general applicable only on problems with nonhomogeneous boundary conditions (see [10], where an example is given, that demonstrates how unhomogeneous boundary conditions can affect stability). Furthermore, it only gives a sufficient stability condition. There is thus need for a more powerful theory. The *normal mode method* was developed by several authors, notably Godunov and

* Received by the editors June 21, 1984, and in revised form March 10, 1985.
† Department of Computer Sciences, Uppsala University, S-752 23 Uppsala, Sweden.

Ryabenkij [6], Kreiss [15], [16] and Osher [22], [23]. Finally a general stability theory based on normal mode analysis was presented by Gustafsson, Kreiss and Sundström [11], [12] for the class of initial-boundary value problems that we are discussing. This theory, henceforth called the GKS-theory, covers linear, first order hyperbolic systems in one space dimension. Since 1971, when the GKS-theory was first presented, related work has been done by Varah [30] for parabolic problems, by Strikwerda [27] for semidiscretized equations and by Michelson [18] for multidimensional problems.

The main topic of this paper is a new numerical algorithm for implementing the GKS-theory. We also present a software package, named IBSTAB, which contains this algorithm and combines it with routines for symbolic algebraic manipulations. The presence of symbolic analysis in IBSTAB is inspired by the software package DCG [5], where an artificial intelligence approach to mathematical software is used. Analogous to DCG, the IBSTAB system also includes a user oriented problem description language. Presently, IBSTAB exists in a pilot version and is not yet available for general distribution.

In order to make the paper self-contained, we give a review of the GKS-theory in § 2. Then § 3 gives an overview of IBSTAB after which a more detailed description is given in §§ 4–6. Finally, test results are presented in § 7 and future plans are discussed in § 8.

2. The GKS-theory.

2.1. The details of the GKS-theory will just be described briefly, as the topic of this paper is not the theory in itself but the implementation of it. The reader who wishes a complete presentation of the theory is referred to [11]. The presentation in that paper is made for homogeneous initial conditions, but the theory can be modified such that it holds for inhomogeneous initial conditions as well, as is shown e.g. in [10].

We will present the stability theory for a quarter-plane problem

$$\begin{aligned} &u_t = Au_x \quad 0 \leqq x < \infty, \quad 0 \leqq t, \\ &u \in \mathscr{L}^2(0, \infty) \end{aligned} \tag{2.1}$$

where

$$u = (u_1, \cdots, u_d)^T, \qquad A = d \times d\text{-matrix}$$

with suitable boundary conditions.

We take A to be constant. However, the theory also covers variable coefficients. For $A \equiv A(x)$ it is shown in [12] that it is sufficient to study the case $A \equiv A(0)$, i.e. it is sufficient to analyze a problem with a frozen coefficient.

Furthermore, if there are two boundaries, then the theory shows that each boundary can be analyzed separately, "removing" the other boundary. Thus, it is sufficient to study quarter-plane problems.

Now to the difference approximation of (2.1). The grid is defined by

$$x_j = j \cdot \Delta x, \qquad j = 0, 1, \cdots,$$

$$t_n = n \cdot \Delta t, \qquad n = 0, 1, \cdots,$$

and the basic (interior) difference scheme has the form

$$\begin{aligned} &\sum_{\nu=-1}^{s} Q_\nu u_j^{n-\nu} = 0, \qquad j = r, r+1, \cdots, \\ &Q_\nu = \sum_{j=-r}^{p} A_{\nu j} E^j, \qquad E u_j^n = u_{j+1}^n, \\ &u \in \ell_2(0, \infty). \end{aligned} \tag{2.2}$$

As Q_ν uses r points to the left, the basic approximation can not be used at $x_0, x_1, \cdots, x_{r-1}$, so there we will have to apply boundary conditions. These can be the conditions that are given for the pde problem, but they can also be difference schemes, which will then be called extra (or numerical) boundary conditions. The choice of extra boundary conditions is crucial for the stability.

By formally introducing $u_j^n = z^n \hat{u}_j$ in (2.2) we arrive at a resolvent equation. It is assumed that the coefficient matrices in the operators Q_ν can be diagonalized by one and the same transformation

$$T^{-1} A_{\nu j} T = D_{\nu j} = \operatorname{diag}(a_{\nu j}^{(1)}, \cdots, a_{\nu j}^{(d)}).$$

This assumption is not a severe restriction since in most cases the matrices $A_{\nu j}$ are powers of the coefficient matrix A, which, because of the hyperbolicity, can be diagonalized.

With the transformation $w_j = T^{-1} \hat{u}_j$ inserted into the resolvent equation, this equation takes the form

$$\sum_{\nu=-1}^{s} z^{s-\nu} \sum_{k=-r}^{p} D_{\nu k} E^k \cdot w_j = 0, \qquad j = r, r+1, \cdots. \tag{2.3}$$

Equation (2.3) represents d scalar difference equations for the components $w_j^{(\tau)}$ of w_j. We seek the solution w_j which is bounded as $j \to \infty$. This solution has the general form

$$w_j = \sum_{k=1}^{\omega} \left(\sum_{\nu=0}^{m_k - 1} \sigma_{k\nu} \cdot V_k \cdot j^\nu \right) \cdot \kappa_k^j \tag{2.4}$$

where m_k is the multiplicity of κ_k and $\sigma_{k\nu}$ are unknown scalar constants. V_k will be unit vectors. κ_k are the solutions such that $|\kappa_k| \leqq 1$ of the *characteristic equations*

$$\sum_{\nu=-1}^{s} z^{s-\nu} \sum_{k=-r}^{p} a_{\nu k}^{(\tau)} (\kappa^{(\tau)})^{k+r} = 0, \qquad \tau = 1, 2, \cdots, d \tag{2.5}$$

corresponding to (2.3).

The solution (2.4) is substituted into the boundary conditions. This gives a linear system of equations for the unknown coefficients $\sigma_{k\nu}$ in (2.4):

$$M\sigma = 0, \qquad \boldsymbol{\sigma} = (\sigma_1, \cdots, \sigma_N)$$

Strong stability (cf. [11, Definition 3.3]) is established iff the condition

$$\det M \neq 0, \qquad |z| \geqq 1$$

is fulfilled. In analyzing stability we look for solutions that violate this condition. Thus, the *determinant condition* is

$$\det M(\kappa_1, \cdots, \kappa_N, z) = 0. \tag{2.6}$$

If (2.6) is satisfied for some value z; $|z| > 1$, $\max |\kappa_k| < 1$, then the scheme is *unstable.* The case $\det M = 0$, $|z| = 1$, can be separated into three cases.

i) All the corresponding κ_k are less than one in magnitude. Then the scheme is classified as *weakly stable* (cf. [11, Def. 3.2]).

ii) There is some κ_k such that $|\kappa_k| = 1$. If all such κ_k are multiple roots of (2.5), then the scheme is classified as *weakly stable.*

iii) At least one κ_k is on the unit circle and is a single root of (2.5). In that case the scheme is *unstable.*

2.2. The practical use of the GKS-theory requires solving a system of polynomial equations:

$$\begin{aligned} &c_1(\kappa^{(1)}, z) = 0, \\ &\vdots \\ &c_d(\kappa^{(d)}, z) = 0, \\ &g(\kappa_1, \kappa_2, \cdots, \kappa_N, z) = 0. \end{aligned}$$

Here c_j, $j = 1, \cdots, d$ are the characteristic equations and g is the determinant condition obtained from the boundary conditions.

In the system we have $d+1$ equations and $N+1$ unknowns, $d \leqq N$. Now, for a fixed z, $c_j(\kappa^{(j)}, z)$ will be a polynomial in $\kappa^{(j)}$ only, having μ_j solutions $\kappa_1^{(j)}, \cdots, \kappa_{\mu_j}^{(j)}$, such that $|\kappa_k^{(j)}| < 1$. All those solutions, taken together for $j = 1, \cdots, d$, are denoted $\kappa_1, \cdots, \kappa_N$. In order to get a system of $N+1$ equations with $N+1$ unknowns we just let each c_j be repeated μ_j times, at the same time renaming the equations. The system of polynomial equations to be solved in order to investigate stability will then be:

$$\begin{aligned} &f_1(\kappa_1, z) = 0, \\ &\vdots \\ &f_N(\kappa_N, z) = 0, \\ &g(\kappa_1, \cdots, \kappa_N, z) = 0. \end{aligned} \tag{2.7}$$

The stability investigation means solving (2.7) for all solutions with $|z| \geqq 1$. This is, despite the simple structure of the f_j-equations, a formidable task. Stability investigations that have been published for special schemes (e.g. [20], [25], [21], [13], [26]) show that already for small problems a lot of work is required to pursue a GKS-analysis.

To simplify the analysis, some researchers have tried to derive new criteria, based on the GKS-theory but more convenient for practical use. Those criteria have only been sufficient, so they do not replace the general theory.

The most far-reaching work along these lines has been done by Goldberg and Tadmor [7], [8], [9].

Their efforts improve the analytical tools for stability investigation. However, the problem remains of solving (2.7) for cases that those criteria do not cover. Some work has been done to develop automatic techniques for that purpose.

Oliger [20] used reduction methods to reduce (2.7) to *one* polynomial of one unknown. This equation was then solved by an ordinary numerical root finder. There were two problems with this approach. First, the reduction was complicated. Secondly, the degree of the resulting polynomial was very high and finding its roots was therefore very time consuming.

Another approach would be to attack (2.7) with some iterative method directly, without reducing the system. Remember that our problem is: find all solutions to (2.7) such that $|z| \geqq 1$. To be able to do this it is necessary to invent some way of finding good initial guesses to all such solutions. When we have a good initial guess, then we can use, e.g., Newton's method to find the corresponding solution.

One way of finding initial guesses is to use a continuation method. This approach was tried by Coughran [4] and also, in a smaller investigation, by Swenson [28]. Both experienced trouble at points where several continuation paths are crossing. Coughran also noticed that a complete stability investigation using a continuation method was very time consuming. The timing was furthermore sensitive to the choice of initial splitting in the continuation method. This is of course a serious drawback if the splitting is to be chosen automatically.

To conclude: In order to investigate GKS-stability some automatic procedure is needed and methods tried in the past have shown serious drawbacks. This is the motivation for the work that has been done within the IBSTAB project.

3. IBSTAB—an overview.

3.1. The central point of IBSTAB is that it contains a new method for finding initial guesses to the solutions of (2.7). The approaches that were described in the last section could be applied to any system of polynomial equations. However, (2.7) has a very special structure, each f_j containing only one κ_j. By taking advantage of that structure it has been possible to develop the efficient numerical algorithm that is the heart of IBSTAB.

The *basic idea* of the method is: Only look for solutions where the z-component is close to the unit circle.

Suppose that for some $\lambda = \lambda_0$ all solutions are such that $|z| \leqq 1$. Furthermore, suppose that a slightly increased $\lambda = \lambda_1 = \lambda_0 + \delta\lambda_0$ does only give slightly different solutions. Then for λ_1 all solutions will be such that $|z| \leqq 1 + \delta$, where δ is small. As we are only interested in the case where $|z| \geqq 1$, it would consequently be sufficient for us to look for solutions with z in the close neighbourhood of the unit circle. By making some very weak assumptions on the underlying difference approximation we can show that it is possible to find λ_0 and that the solutions for subsequent λ-values will only be perturbations of those that we get for $\lambda = \lambda_0$. This shows that the basic idea is reasonable.

How can this basic idea be used in practice?

For a fixed λ-value, we try to find initial guesses to the solutions of (2.7), by searching for such guesses with $|z| = 1 + \xi$. We take z-values on the "search circle" $|z| = 1 + \xi$. For each fixed z we solve for κ_j from f_j, $j = 1, \cdots, N$. This is where *the special structure of* (2.7) is used. We thus get $\alpha = (\kappa_1, \cdots, \kappa_N, z)$, such that $f_j(\alpha) = 0$, $j = 1, \cdots, N$. If, in addition, $g(\alpha)$ is small, then α is taken as initial guess to some iteration method for solving (2.7). (IBSTAB uses a modified version of the IMSL-routine ZSYSTM, which implements Brown's method [3], [14].)

It should be noted that the main advantage of this algorithm is that we only have to search in *one* variable. If we would have to search in the $N+1$ variables $\kappa_1, \cdots, \kappa_N, z$, then the number of possibilities would in general be immense and the method would be impractical. But by using the structure of (2.7) we can solve for $\kappa_1, \cdots, \kappa_N$ *exactly* (or with high accuracy by some iteration method) from the characteristic equations, as those are polynomials in κ for a given z. This allows us to search in z only and this leads to an efficient algorithm for analyzing stability.

In order that the algorithm find all critical solutions of (2.7), what assumptions are necessary on the difference approximation? We will denote the difference approximation at inner points by $D(\lambda)$. Our assumptions are:

1. $D(0)$ is a stable approximation to the ordinary differential equation $u_t = 0$.
2. No higher time level is involved in $D(\lambda)$, $\lambda > 0$, than in $D(0)$.

The first assumption is natural. The second, however, is a *restriction*, but not a severe one as most schemes currently in use fulfill it.

These assumptions are necessary in order to assure that the mathematical premises of the IBSTAB algorithm are fulfilled. The premises are:

i) The system of polynomial equations has the structure of (2.7).

ii) (from first assumption). For $\lambda = 0$ all solutions of (2.7) are such that $|z| \leqq 1$.

iii) (from second assumption). For $\lambda > 0$ all z-values are perturbations of the z-values obtained for $\lambda = 0$.

Premise ii) is used to get a startvalue $\lambda = \lambda_0 = 0$, such that for λ_0 all solutions fulfill $|z| \leqq 1$. Premise iii) will then assure that for slightly increased λ-values the solutions will only change slightly. These are exactly the two facts that are needed for the basic idea to be reasonable.

It is important that $|z| = 1 + \xi$ is chosen as "search circle" (instead of $|z| = 1$, which might seem to be a more natural choice). The reason is that when the von Neumann condition is fulfilled, which is assumed in the GKS-theory, then for $|\kappa| = 1$ we must have $|z| \leqq 1$. Thus, in taking $|z| = 1 + \xi$, $\xi > 0$, we are sure that the κ-values split nicely into two groups, one with $|\kappa| < 1$ and one with $|\kappa| > 1$. This will simplify the implementation of the IBSTAB algorithm, making it easy to pick the right κ-values, i.e. those with $|\kappa| < 1$, for each characteristic equation.

3.2. After this introductory presentation of the numerical algorithm we give an overview of the entire IBSTAB system. As mentioned there are also parts for symbolic algebraic manipulations and for problem description.

There are two difficulties in the GKS-theory, that motivate the use of software tools for the investigation. The main difficulty is of course the solution of (2.7), which we have already discussed.

The other difficulty is the derivation of (2.7) from the underlying difference approximation. This can be a tedious and error-prone work. Consequently a program for automatic derivation of (2.7) would be of great help.

In the construction of IBSTAB the aim has been to apply the ideas of mathematical software used in the software package, named DCG, which has been presented by Engquist and Smedsaas [5]. The IBSTAB system thus combines symbolic manipulations and numerical routines. Furthermore, it is intended to allow the user to present the problem in a "natural" manner that will not include the writing of FORTRAN subroutines or functions. Therefore IBSTAB contains an extended version of the problem specification language used in DCG, which makes it possible to present the stability problem in a notation that hopefully seems natural to a numerical analyst.

The overall structure of IBSTAB is presented in Fig. 3.1.

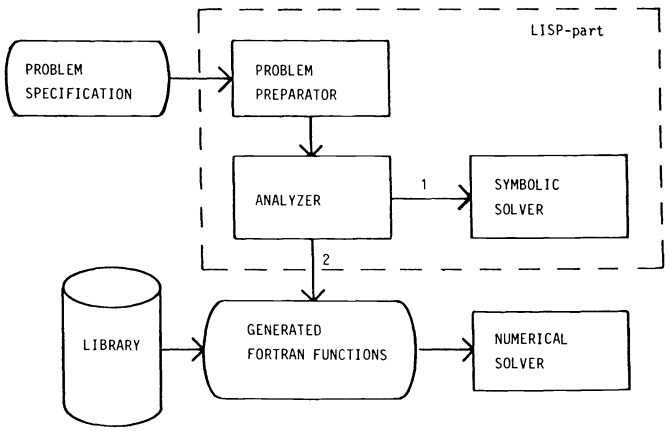


FIG. 3.1

The symbol manipulation parts of the system are written in LISP, using the LISP-F3 interpreter [19], and the numerical part is written in FORTRAN. LISP-F3 is coded in FORTRAN and thus IBSTAB is completely FORTRAN based. Another advantage in using LISP-F3 is that it gives the possibility to make external calls to

FORTRAN routines from LISP. This will be used e.g. when diagonalizing the matrices of the difference equations (in order to have premise i) of the last paragraph fulfilled).

Now the parts of IBSTAB will be introduced briefly. The *problem specification* is given by the user. It is read by the *problem preparator*, which also generates the nonlinear system of equations (2.7). The main task of the *analyzer* is to decide whether this system should be solved by a direct or by a numerical method. If the problem is simple enough to be solved analytically, then the *symbolic solver* takes over, making the stability analysis using symbolic manipulations. If, on the other hand, the analysis shows that the numerical algorithm must be used, then the analyzer

- symbolically calculates the Jacobian of (2.7);
- generates FORTRAN functions for both (2.7) and its Jacobian.

The *generated FORTRAN functions* together with problem independent routines from a *library* form the *numerical solver*, which will now take over and perform the stability analysis.

It should be pointed out that IBSTAB treats not only GKS-stability but also checks the von Neumann stability (which is an underlying assumption in the GKS-theory).

Furthermore, if GKS-stability is assumed, then P-stability [1] can be checked in a separate pass, rerunning IBSTAB. However the P-stability investigation is not the main purpose of IBSTAB and is therefore not treated as generally as the GKS-stability check.

4. The numerical solver.

4.1. We will now go into the details of the new numerical algorithm for automatic GKS stability analysis.

Assuming that the reader is familiar with the basic ideas, we first give a more complete description of the algorithm, which we will then discuss and motivate. For the description we use a quasi-ALGOL notation:

ALGORITHM 4.1.
for $\lambda = \lambda_1$ **step** λ_{step} **until** λ_{stop} **do**
1. Compute ε.
2. Check the von Neumann condition.
3. **for** $\theta = 0$ **step** θ_{step} **until** 2π **do**
3.1. $z = (1+\xi) \cdot e^{i\theta}$.
3.2. Compute κ_k from the characteristic equations to get a "search point" $\alpha = (\kappa_1, \cdots, \kappa_N, z)$.
3.3. Compute GABS $= |g(\alpha)|$.
3.4. **if** GABS $= 0$ **then**
if instability **then** stop.
else
3.5. **if** GABS $\leqq \varepsilon$ **then**
3.5.1. Solve (4.1) using Brown's method with α as initial guess.
3.5.2. **if** convergence **then**
if instability **then** stop.
3.6. Compute new θ_{step}.
3.7. **if** θ_{step} "small" **then**
3.7.1. Solve (4.1) using Brown's method with α as initial guess.
3.7.2. **if** convergence **then**
if instability **then** stop.
3.8. **endfor** θ.
4. Compute new λ_{step}.
5. **endfor** λ.

4.2. It is of course important that the stepsize λ_{step} is in each step chosen so that no z will be more than slightly perturbed. We shall here develop an algorithm for that choice. In order to do that we write the system (2.7) as a vector equation

$$G(\lambda, \alpha) = 0, \tag{4.1}$$

where $\alpha = (\kappa_1, \cdots, \kappa_N, z)$, $G = (f_1, \cdots, f_N, g)^T$.

Let $\alpha_{\delta\lambda}$ be a solution to (4.1) for the λ-value $\lambda + \delta\lambda$, i.e.

$$G(\lambda + \delta\lambda, \alpha_{\delta\lambda}) = 0,$$

and take $\alpha_{\delta\lambda}$ to be a perturbation of α_0. A Taylor expansion gives, for small values of $\delta\lambda$,

$$G_\lambda(\lambda, \alpha_0) \cdot \delta\lambda \approx -J_\alpha(\lambda, \alpha_0)(\alpha_{\delta\lambda} - \alpha_0) \tag{4.2}$$

where G_λ denotes the gradient vector with respect to λ. J_α is the Jacobian matrix with respect to α. We now introduce the parameter Δ. We want $\alpha_{\delta\lambda}$ to fulfill $|z| \leqq 1 + \Delta$, where Δ is chosen small enough that Algorithm 4.1 works. Assuming that α_0 fulfills $|z| \leqq 1$ we thus impose the condition $\|\alpha_{\delta\lambda} - \alpha_0\| \leqq \Delta$. If we use the latter condition in (4.2) then we get the following approximate bound on λ_{step}:

$$\lambda_{\text{step}} \leqq \Delta \cdot \min_{\alpha} (\|J_\alpha(\lambda, \alpha)\|_\infty / \|G_\lambda(\lambda, \alpha)\|_\infty) \tag{4.3}$$

where the minimization is made over those solutions α which were found for the previous λ-value.

For efficiency we wish λ_{step} to be as large as possible, so we take equality in (4.3) as our formula for λ_{step}. As the Taylor analysis breaks down if λ_{step} is not small, we also impose a maximum limit

$$\lambda_{\text{step}} \leqq \bar{\lambda}_{\text{step}}.$$

The choice $\lambda_{\text{step}} = \bar{\lambda}_{\text{step}}$ is used not only when the formula gives too large values, but also when no solutions were found for the previous λ-value.

4.3. The choice of stepsize θ_{step} for the argument of z is governed by the following analysis. For z on the circle

$$z = (1 + \xi) \cdot e^{i\theta}, \qquad 0 \leqq \theta < 2\pi$$

we can regard the system $G(\alpha)$ as a function of θ instead of z, i.e. we take $\alpha = (\kappa_1, \cdots, \kappa_N, \theta)$.

We take α such that $G(\alpha) \neq 0$ and we look for $\delta\alpha$ such that $G(\alpha + \delta\alpha) = 0$. With a linearization we get $\delta\alpha$ to be the ordinary Newton step

$$\delta\alpha \approx -J^1(\alpha) \cdot G(\alpha)$$

where J is the Jacobian of G.

We look for $\delta\theta$, which is the last component of $\delta\alpha$.

In general $\delta\theta$ is complex valued. However, θ_{step} should be real and furthermore the angular change is given by $\text{Re}\,(\delta\theta)$, whereas $\text{Im}\,(\delta\theta)$ represents a change in magnitude of z. Thus a reasonable choice of θ_{step} is

$$\theta_{\text{step}} = |\text{Re}\,(\delta\theta)|. \tag{4.4}$$

The choice (4.4) is used together with a maximum limit

$$\theta_{\text{step}} \leqq \bar{\theta}_{\text{step}}.$$

In Algorithm 4.1 we also, in step 3.7, introduced a lower limit of θ_{step}, saying that if θ_{step} is "small" then Brown's method should be used. By "small" we shall mean

$$|\theta_{\text{step}}| \leqq 2 \cdot \frac{|g(\alpha)|\mu}{\varepsilon} \quad \text{if} \quad \left|\frac{\partial f_j(\alpha)}{\partial \kappa_j}\right| > c, \qquad j = 1, \cdots, N.$$

This is fulfilled if the derivatives involved in $\delta\theta$ are large. μ is taken to be a constant, small enough to avoid unnecessary use of Brown's method. c is also a small constant, such that $|\partial f_j(\alpha)/\partial \kappa_j| \leqq c$ means that $\partial f_j/\partial \kappa_j$ is "almost singular" at α. If $|\partial f_j(\alpha)/\partial \kappa_j| \leqq c$ for some j, then θ_{step} is also considered "small" and Brown's method is called. In practice the computation is done in the following way:

- Compute θ_{step} using (4.4) (if possible);
- if $|g(\alpha)| > \varepsilon$ then check if θ_{step} "small";
- if $\theta_{\text{step}} > \bar{\theta}_{\text{step}}$ then set $\theta_{\text{step}} = \bar{\theta}_{\text{step}}$.

4.4. We want the parameter ε to be an indicator of whether the residual $|g(\alpha)|$ of the determinant condition is small enough to make it probable that α is close to a solution to (2.7). It is chosen to be an average size of g according to the following procedure:

- For λ_1 we use

$$\varepsilon = \frac{1}{2} \cdot \sum_{j=1}^{4} |g(\alpha_j)|/4$$

  where α_j is the search point with z-value $z_j = (1+\xi)(i)^j$
- For the subsequent λ-values we take

$$\varepsilon = \tfrac{1}{2} \cdot (\text{average of the } g\text{-residuals at the search points generated for the preceding } \lambda\text{-value}).$$

4.5. We will now justify Algorithm 4.1 by stating two propositions. For that purpose we introduce the following notation. Let $\alpha = (\kappa_1, \cdots, \kappa_N, z)$ be a given search point, i.e. a point such that

$$\begin{aligned} &f_j(\kappa_j, z) = 0, \qquad j = 1, \cdots, N, \\ &z = (1+\xi)e^{i\theta}. \end{aligned} \tag{4.5}$$

A point α that fulfills (4.5), but that is not guaranteed to be among the set of search points is called a presumptive search point. Furthermore, let $\alpha^* = (\kappa_1^*, \cdots, \kappa_N^*, z^* = |z^*|e^{i\theta^*})$ be a solution to (2.7) and introduce the distance

$$\operatorname{dist}(\alpha^*, \alpha) = \left[\sum_{j=1}^{N} |\kappa_j^* - \kappa_j|^2 + |z^* - z|^2\right]^{1/2}.$$

We can now state the propositions.

PROPOSITION 4.1. *Assume*:

a) *Brown's method has convergence radius R, such that*

$$R \geqq \gamma > 0$$

where γ is a fixed constant.

b) *All derivatives of f_k, $k = 1, \cdots, N$, and g are majorized by a constant $\bar{K}$ at every point β such that*

$$\operatorname{dist}(\alpha^*, \beta) \leqq \delta.$$

Then there is δ_0 such that for $\delta \leqq \delta_0$ IBSTAB *will find α^* if α^* is the only solution such that*

$$\text{dist}(\alpha^*, \alpha) \leqq \delta, \qquad |\theta^* - \theta| \leqq \delta.$$

PROPOSITION 4.2. *We can choose Δ small enough that if*

$$1 \leqq |z^*| \leqq 1 + \Delta$$

then there is a presumptive search point α fulfilling

$$\text{dist}(\alpha^*, \alpha) \leqq \delta_0, \qquad |\theta^* - \theta| \leqq \delta_0.$$

For a proof of Proposition 4.1, see [29]. The second proposition follows trivially from the fact that when z^* is close to z, then κ_j^* is a perturbation of κ_j (cf. [29]).

The first proposition shows that there is a convergence region around every search point. Now, assume that for $\lambda = \lambda_j$ all solutions α^* fulfill $|z^*| \leqq 1$. By our way of choosing λ_{step} we have that for $\lambda = \lambda_{j+1}$ any α^* such that $|z^*| \geqq 1$ must fulfill $1 \leqq |z^*| \leqq 1 + \Delta$. Then Proposition 4.2 says that all such critical solutions are within the convergence region of a presumptive search point.

Some remarks are needed:

- We are interested in finding all solutions in a neighbourhood of the unit circle with $1 \leqq |z^*| \leqq 1 + \Delta$. Now, we have *not* shown that all such solutions will be within the convergence region of a true search point.
- The assumptions imply cases for which the algorithm might not work. Pinpointing such "critical" cases is one of the benefits of the kind of justification given here.

The uncertainties that are pointed out in the remarks must be covered for by extensive testing of the algorithm. Test results are presented in § 7.

4.6. It now seems appropriate to discuss how to choose Δ. Proposition 4.2, which is where Δ was used, is a poor guide for that choice.

For that reason we will try the following approach (using the notation from § 4.2). Choose Δ such that if, for $\lambda = \lambda_{j+1}$, we use α_0 as initial guess to Brown's method, we will get convergence to $\alpha_{\delta\lambda}$. This means that Δ must be chosen on basis of the convergence properties of Brown's method. Looking at a local convergence theorem given by Brent [2], we find that a bound on the convergence radius R of Brown's method is

$$R \geqq 1/(3 \cdot N \cdot L \cdot \|J^{-1}\|),$$

where N is the number of equations, L is a Lipschitz constant for the jacobian J and $\|J^{-1}\|$ is the norm of J^{-1} taken in the solution point.

Thus, a reasonable choice of Δ, assuming that J is well-conditioned, would be

$$\Delta = 1/(3N).$$

This is the formula that is used in IBSTAB. To show that it is satisfactory, we refer to the test results.

4.7. After this presentation of the numerical algorithm we will just give some short remarks on its implementation. First, it has turned out in practice to be suitable to impose lower bounds on λ_{step} and θ_{step}

$$\lambda_{\text{step}} \geqq \underline{\lambda}_{\text{step}}, \qquad \theta_{\text{step}} \geqq \underline{\theta}_{\text{step}}$$

to avoid unnecessarily small steps.

Secondly, we have found that when a solution has been computed, then it is necessary to take a rather large θ_{step} in order to get out of the region of convergence of that solution. This is done by taking

$$\theta_{\text{step}} = \bar{\theta}_{\text{step}}/2 \qquad \text{(cf. § 4.3)}$$

every time a solution is encountered.

Finally let us assume that the difference approximation being investigated is found to be stable for $\lambda \leqq \lambda_L$, but that it is unstable for the next λ-value, $\lambda = \lambda_R$. Then IBSTAB continues with a refinement step, where the "uncertainty interval" $]\lambda_L, \lambda_R[$ is reduced by bisection. Here *either* von Neumann- or GKS-stability is checked. The refinement goes on until $\lambda_R - \lambda_L \leqq \text{TOL}$, where TOL is a given tolerance.

4.8. We will end the description by discussing how to modify the algorithm in order to implement it on a parallel processing computer. One possibility to do that efficiently would be to have all (or groups of) the z-values for a given λ examined simultaneously. This could be done if we chose a fixed step size around the circle $|z| = 1 + \xi$, taking some "minimal" value of θ_{step} and calling on the "system solver" only when $|g| \leqq \varepsilon$. If all the processors flagged stability, then one processor could calculate a new λ-value etc. If instead we consider a vector machine, then, still keeping a fixed "minimal" θ_{step}, picking the search points for which Brown's method should be called could be done in a couple of "vector sweeps". The details of such implementations have not been considered yet.

5. The LISP-part of IBSTAB. The numerical algorithm is in IBSTAB surrounded by routines for symbolic analysis. Those routines were introduced in § 3.2. As IBSTAB does only exist in a pilot version and is not yet available for general distribution, we will here just make some short comments on the LISP part of the software package.

The main task of the *analyzer* is to check whether the system (2.7) is simple enough to be solved by the symbolic solver. The criteria for this are;

i) There are only two equations in the system, i.e., there is only one characteristic equation.

ii) The second equation, i.e., the determinant condition, is of at most first degree in z and fourth degree in κ and the coefficient for z is independent of κ.

iii) Substituting z in the characteristic equation will yield at most a fourth degree polynomial in κ.

The kind of modifications and improvements that might be considered for the LISP part of IBSTAB would primarily be changes in order to enlarge the class of problems that could be treated without using the numerical solver. One such possibility is to include the criteria of Goldberg and Tadmor, that we discussed in § 2.2. Another way could be to attach to IBSTAB a data base containing known stability results. Whenever a problem was presented to IBSTAB, the system would then begin by comparing it to the problems in the data base. This latter idea is rather loose and I have not considered any details of how the data base should be constructed.

The modification, that would give the most far-reaching enlargement of the class of problems that could be treated analytically, is to use some better method for solving systems of polynomial equations by symbolic algebraic manipulations. It is the opinion of the present author, that the most promising method is the one developed by Lazard [17]. However, careful consideration would certainly be needed to make a useful implementation of the method. It is also unclear how efficient any implementation might be if it were to be used for problems of realistic size (i.e., problems that are larger than those that can currently be treated by the symbolic solver). Putting some effort into the investigation of those questions could possibly be worthwhile.

6. The problem description language. As stated in § 3.2 one of the intentions with IBSTAB is that it should have a user oriented problem description language. The user should be able to specify the problem in a manner that is natural to her. To illustrate how this goal has been approached we give a short example.

We wish to investigate the stability of a leap-frog approximation to

$$v_t = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} v_x \qquad 0 \leqq x < \infty, \quad 0 \leqq t,$$

$$v^I(0, t) = 0$$

with extra (numerical) boundary condition

$$w_0^n = 2w_1^n - w_2^n,$$

where w_j^n is the approximation to $v^{II}(x_j, t_n)$. The investigation should be made for $0 \leqq \lambda \leqq 2$. The IBSTAB notation for this would be, e.g.,

```
DIFFERENCEEQS
    U(J, N+1) - U(J, N-1) - L*A*(U(J+1, N) - U(J-1, N)) = 0.0
BOUNDARYCONDS
    B1*U(0, N) - B2*U(1, N) + B3*U(2, N) = 0.0
PARAMETERS
    DEPVAR = U;
    NDE = 2;
    NBCND = 2;
    DCOEFF = A: 0.0 1.0
                1.0 0.0;
    BCOEFF = B1: 1.0 0.0
                 0.0 1.0,
             B2: 0.0 0.0
                 0.0 2.0,
             B3: 0.0 0.0
                 0.0 1.0;
    LSTOP = 2.0
END
```

This example shows the structure of the problem description language. It is based on self-explaining section names. Expressions must be given with right-hand side identical to zero. The difference equations and boundary conditions should be presented in vector form. The name L for λ is fixed but the names of the dependent variable (DEPVAR) and of difference equation- and boundary condition coefficients (DCOEFF and BCOEFF) can be chosen freely. As shown by the example those are specified in the PARAMETERS section, where also the values of the coefficients are given. The example is not exhaustive: there are also sections for describing the polynomial equations if one has already derived those by hand. In this case one must use the notation Z for z, $K1$ for κ_1, $K2$ for κ_2, etc.

It seems, that for a person working in engineering or numerical mathematics the notation adopted in IBSTAB is much more natural than e.g. a conventional FORTRAN notation. The size of the description program is also considerably smaller than for a corresponding FORTRAN code.

7. IBSTAB in practice. We have now reached the point, where the efficiency of the numerical algorithm for automatic GKS stability analysis shall be demonstrated

by experimental results. First we will show how IBSTAB managed to solve test examples for which the correct answers were already known. Then a real-life experiment will be presented, where IBSTAB was addressed to a previously unsolved problem. In this case the result was checked by running the investigated scheme using the DCG system.

The choice of test examples is always critical. Any algorithm can probably be shown to be superior by making a suitable choice of test data. We have not chosen our test examples in such a biased way. On the other hand, we have not intentionally tried to design test examples that would check the critical cases that were indicated in Proposition 4.1. Our principle has been to try IBSTAB on problems for which the results were known and furthermore, to choose the problems so that they cover a wide range of problem types with respect to order of approximation and number of pde's. In this context it should be noted that there is not vast literature on known results, so the choice of test problems has been limited.

For each test some statistics have been collected:

N_λ = total number of λ-values,

N_z = total number of z-values,

N_{BM} = total number of calls to Brown's method,

$\overline{N_z} = N_z / N_\lambda$.

$\overline{N_{BM}} = N_{BM} / N_\lambda$,

CPU-time (in minutes).

The values of N_λ, $\overline{N_z}$, $\overline{N_{BM}}$ and CPU-time will be reported. The stability result is given as a pair of λ-values λ_L, λ_R. The scheme is stable for $\lambda \leqq \lambda_L$ and unstable for $\lambda \geqq \lambda_R$.

Another point of interest is the actual choice of the "open" parameters that were introduced in § 4. Those values are listed below. For each parameter we include a reference to the section where it was defined. The values used in the current implementation of IBSTAB are:

$$\lambda_1 = 0.06 \qquad (4.1)$$

$$\xi = 0.001 \qquad (4.1)$$

$$\bar{\lambda}_{\text{step}} = 0.2 \qquad (4.2)$$

$$\lambda_{\text{step}} = 0.05 \qquad (4.7)$$

$$\bar{\theta}_{\text{step}} = 0.1 \qquad (4.3)$$

$$\theta_{\text{step}} = 0.01 \qquad (4.7)$$

$$\mu = 0.04 \qquad (4.3)$$

$$\text{TOL} = 0.025 \qquad (4.7)$$

All tests runs were made on the BASF 7/68 at Uppsala University Computer Center, using the WATFIV compiler.

7.1. Test problem I.

$$u_t = \begin{pmatrix} 0 & a(x) \\ a(x) & 0 \end{pmatrix} u_x, \quad u = \begin{pmatrix} u^I \\ u^{II} \end{pmatrix}, \quad 0 \leqq x < \infty, \quad t \geqq 0$$

$$u^{II}(0, t) = 0.$$

This is the wave equation written as a first order system. We apply the leapfrog scheme. With linear extrapolation as extra boundary condition and with $a(0) = 1$ (cf. § 2.1, where the treatment of variable coefficients was discussed) the scheme is weakly stable for $\lambda < 1$. For $\lambda \geqq 1$ the von Neumann condition is not fulfilled.

Result.

$$\lambda_L = 0.98500, \qquad \lambda_R = 1.01000,$$

$$N_\lambda = 22,$$

$$\overline{N_z} = 67,$$

$$\overline{N_{BM}} = 28,$$

$$\text{CPU-time} = 2.62.$$

7.2. Test problem II.

$$u_t = a(x, t)u_x \quad 0 \leqq x \leqq 1, \quad t \geqq 0, \quad a(0, t) = -1$$

$$u(0, t) = 0.$$

We apply the fourth order Kreiss-Oliger scheme [20] and take the extra boundary conditions analyzed by Sloan [26] and Oliger [20].

Result:

a) Right quarter-plane problem:

$$\lambda_L = 0.71000, \qquad \lambda_R = 0.73500,$$

$$N_\lambda = 16,$$

$$\overline{N_z} = 71,$$

$$\overline{N_{BM}} = 31,$$

$$\text{CPU-time} = 1.41.$$

b) Left quarter-plane problem:

$$\lambda_L = 0.66000, \qquad \lambda_R = 0.68500,$$

$$N_\lambda = 15,$$

$$\overline{N_z} = 128,$$

$$\overline{N_{BM}} = 118,$$

$$\text{CPU-time} = 6.66.$$

This confirms Sloan's analysis.

7.3. Test problem III.

$$u_t = a(x, t)u_x \quad 0 \leqq x < \infty, \quad t \geqq 0, \quad a(0, t) = 1.$$

We use an approximation and boundary conditions studied by Oliger [21]. It uses a coarse grid in the inner part of the region and a fine grid near the boundary. The corresponding spatial step sizes are denoted Δx_c and Δx_f.

On the coarse grid we take the fourth order Kreiss-Oliger scheme (cf. Test problem II) and on the fine grid, in the interval $[0, 2\Delta x_c]$, we use the leapfrog approximation.

We define the parameters $\lambda_f = \Delta t/\Delta x_f$ and $\lambda_c = \Delta t/\Delta x_c$ and the scale translation factor $S = \Delta x_c/\Delta x_f$. Consequently we have $\lambda_f = S \cdot \lambda_c$. In our test, we used $S = 5$.

Oliger [21] was able to show theoretically that for $S \geqq 2$ the approximation is unstable for all values of λ_c. With θ denoting $\arg(z)$ he proved that there are two intervals $[\theta_2, \theta_3]$ and $[\theta_4, \theta_5]$ in each of which for some θ^* the determinant condition is violated.

Result:

$$\lambda_L = 0.03000, \qquad \lambda_R = 0.04500,$$
$$N_\lambda = 3,$$
$$\overline{N_z} = 57,$$
$$\overline{N_{BM}} = 54,$$
$$\text{CPU-time} = 0.48.$$

We notice that IBSTAB failed to find the critical solution for $\lambda = 0.03$. However, a search point α was generated with $\arg(z)$ very close to $\theta^* \in [\theta_2, \theta_3]$ and at that search point Brown's method was called. Thus the failure is due to the fact that Brown's method evidently has a very small convergence radius around the solution α^*.

In [29] this test is discussed in more detail. The conclusion is that IBSTAB failed but that the failure is not alarming. The erroneous stability interval $\lambda \in [0, \lambda_L]$ is very small. IBSTAB was in fact for $\lambda \geqq 0.045$ able to confirm Oliger's theoretical analysis. It is also reassuring that the search procedure even for the difficult λ-values managed to generate good initial guesses to Brown's method.

7.4. Test problem IV.

$$\begin{pmatrix} \rho \\ u \\ p \end{pmatrix}_t + \begin{pmatrix} \hat{u} & \hat{\rho} & 0 \\ 0 & \hat{u} & \hat{\rho}^{-1} \\ 0 & \gamma \cdot \hat{p} & \hat{u} \end{pmatrix} \begin{pmatrix} \rho \\ u \\ p \end{pmatrix}_x = 0, \qquad 0 \leqq x < \infty, \quad t \geqq 0.$$

These are the linearized Euler equations. We study the subsonic inflow problem with extra boundary condition

$$[\hat{\rho} c u - p]_0^{n+1} = 2[\hat{\rho} c u - p]_1^n - [\hat{\rho} c u - p]_2^{n-1}.$$

We choose $\hat{u}$, $\hat{\rho}$ and c so as to give, according to the analysis of Gustafsson and Oliger [13], GKS-stability for $\lambda \leqq 2$.

Result:

$$\lambda_L = 1.98314, \qquad \lambda_R = 2.00338,$$
$$N_\lambda = 33,$$
$$\overline{N_z} = 117,$$
$$\overline{N_{BM}} = 99,$$
$$\text{CPU-time} = 13.56.$$

7.5. More details on the test examples are given in [29], where also additional tests are presented. We now make some general comments on the test results.

It should be emphasised that all the tests were made using one and the same choice of open parameters. No trimming was made to improve the results. The statistics show that the CPU-time was very modest, especially considering that we used a nonoptimizing compiler and considering the complexity of the problem to be solved. However, the number of calls to Brown's method was rather large. The ratio $\overline{N_{BM}}/\overline{N_z}$ ranges from 0.42, for Test problem I, to 0.95, for Test problem III. This is probably a price that has to be paid in order to get a robust algorithm, even if those ratios may possibly be reduced by making an optimal choice of open parameters.

7.6. After having been successful on the test problems with known results, IBSTAB was addressed to a real-life problem, where it was used as a tool in choosing numerical boundary conditions. The stability limits obtained by IBSTAB were double checked by running the approximation using the software package DCG.

The problem was given by Bertil Gustafsson and has the following background. Shock propagation in a compressible fluid is governed by a system of conservation laws

$$U_t + F_x(U) = 0$$

where the components of $U = (\rho, u\rho, E)^T$ are density, momentum and energy. Assume that a frontal shock is propagating to the left through a fluid at rest, and that the flow behind this shock contains several other shocks. A possible solution method for such a problem is the von Neumann-Richtmyer difference scheme [24] combined with shock fitting for the frontal shock. The scheme uses a staggered grid as shown in Fig. 7.1.

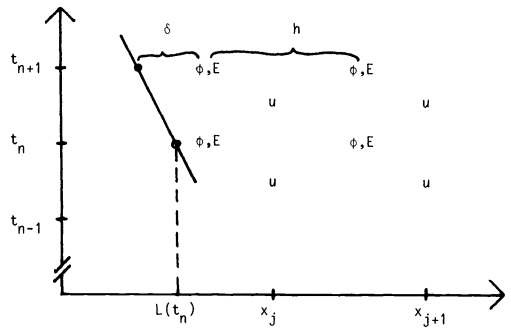


FIG. 7.1

The Rankine-Hugoniot conditions are used to define the state at the frontal shock. This requires one additional condition, which is obtained by using an extrapolation procedure for u. A plane $P(x, t)$ in the (x, t, u)-space is defined by using the u-values at $(x_j, t_{n-1/2})$, $(x_j, t_{n+1/2})$, $(L(t_n), t_n)$, where $L(t)$ denotes the shock position at time t. The u-value at the shock for $t = t_{n+1}$ is then taken as $P(L(t_{n+1}), t_{n+1})$.

A proper model problem is obtained by linearizing the problem and freezing the coefficients. If the variables are renumbered, and if it is assumed that the shock is stationary, then the problem reduces to

$$\begin{bmatrix} \phi_j^{n+1} \\ \psi_j^{n+1} \end{bmatrix} = \begin{bmatrix} \phi_j^n \\ \psi_j^n \end{bmatrix} + k \cdot \begin{bmatrix} D_+\psi_j^n \\ D_-\phi_j^{n+1} \end{bmatrix}.$$

The problem is how to choose a numerical boundary condition for ψ at $x = 0$. The procedure described above leads for the model problem to the boundary condition

$$(1+\theta)\psi_0^{n+1} = (1+\theta)\psi_0^n + \theta(\psi_1^{n+1} - \psi_1^n) + \alpha(\phi_0^{n+1} - \phi_0^n), \qquad -0.5 < \theta \leqq 0.5, \quad 0 \leqq \alpha \leqq 10.$$

The question to be answered by IBSTAB was: for what values of θ and α is the approximation stable?

The suggested approximation of the model problem yields the following system of polynomial equations:

$$(z-1)^2 \cdot \kappa - \lambda^2 z(\kappa - 1)^2 = 0,$$

$$(1+\theta)(1-z) - \theta(1-z)\kappa - \alpha\lambda(1-\kappa) = 0.$$

In the investigation we chose fixed α-values $\alpha = 1.1$ and $\alpha = 1$ and varied θ for each of those. Results are shown in Table 1, where (N) means von Neumann- and (G) means GKS-instability. For $\theta = 0$ IBSTAB was able to use the symbolic solver. In that case the user interactively prescribes the λ-values for which the investigation should be made. Then only $\lambda \leqq 0.99$ was tested, as $\lambda = 1$ is the limit for von Neumann stability. Thus we got no λ_R-value in that case.

TABLE 1

	θ	λ_L	λ_R
$\alpha = 1.1$	−0.49	0.66823	0.68358 (G)
	−0.4	0.75554	0.76808 (G)
	−0.3	0.79157	0.81657 (G)
	−0.2	0.90934	0.93434 (G)
	−0.1	0.96078	0.98578 (G)
	0	0.99	* (N)
	0.1	0.98536	1.00058 (N)
	0.2	0.99046	1.00483 (N)
	0.3	0.98877	1.00985 (N)
	0.4	0.98500	1.01000 (N)
	0.5	0.98500	1.01000 (N)
$\alpha = 1$	−0.49	0.71281	0.72653 (G)
	−0.4	0.77663	0.80163 (G)
	−0.3	0.87560	0.90060 (G)
	−0.2	0.93528	0.96028 (G)
	−0.1	0.99196	1.01696 (N)
	0	0.99	* (N)
	0.1	0.98017	1.00517 (N)
	0.2	0.97936	1.00436 (N)
	0.3	0.98500	1.01000 (N)
	0.4	0.98564	1.01064 (N)
	0.5	0.98500	1.01000 (N)

A parallel test was made by running the scheme with $\alpha = 1$, using the DCG system. The results confirm those obtained by IBSTAB. What do the results mean? The definition of θ is

$$\theta = \delta / h,$$

where δ and h are explained in Fig. 7.1. More exactly $\delta = x_{j-1/2} - L(t_{n+1})$. That means that θ corresponds to the position of the shock at $t = t_{n+1}$, which is x_{j-1} for $\theta = 0.5$ and x_j for $\theta = -0.5$. The extrapolation procedure described in the previous paragraph will be worse the closer the shock is to x_j, i.e. the closer θ is to -0.5. This is mirrored in the stability results, as the stability limit becomes more and more restrictive as $\theta \to -0.5$ from the right.

8. Future plans. In IBSTAB we have obtained an efficient tool for automatic stability analysis. But still there are several paths along which IBSTAB can be extended.

One extension would be to elaborate the P-stability check. The tests already made within the IBSTAB project are promising and the work seems worth continuing.

Another extension, which is perhaps the most urgent, is to implement Michelson's theory for problems in several dimensions to be able to cover a larger class of real-life problems. The theory is only developed for special cases but nevertheless it would be

worth implementing. It should be noted that even for cases that the theory does not cover the uniform Kreiss condition [18] could be formally derived. This condition is then not sufficient but necessary for stability.

A third extension concerns problems where the parameters vary within a certain range. For those problems it would be nice if the user was allowed to prescribe a range of coefficient values for which the stability investigation should be made in one single run.

Note. As has been stressed several times in this paper, the complete IBSTAB system does only exist in a pilot version at Uppsala University. However, after the submission of the paper, the problem independent part of the numerical solver has become available for general distribution and can be ordered from the author. In this "public" version of IBSTAB, Brown's method has been replaced by Powell's hybrid method. This was motivated by the result for test problem III, which was the only test case, where the pilot version failed. With the new version, a correct stability limit is obtained for that problem as well.

Acknowledgments. I am indebted to Bertil Gustafsson for many discussions on the work described in this paper. I also wish to thank Tom Smedsaas for his suggestions on software design.

REFERENCES

[1] R. M. BEAM, R. F. WARMING AND H. C. YEE, *Stability analysis of numerical boundary conditions and implicit difference approximations for hyperbolic equations*, J. Comp. Phys., 48 (1982), pp. 200–222.

[2] R. P. BRENT, *Some efficient algorithms for solving systems of nonlinear equations*, SIAM J. Numer. Anal., 10 (1973), pp. 327–343.

[3] K. M. BROWN, *Computer oriented algorithms for solving systems of simultaneous nonlinear algebraic equations*, in Numerical Solution of Systems of Nonlinear Algebraic Equations, G. D. Byrne and C. A. Hall, eds., Academic Press, New York, 1973.

[4] W. M. COUGHRAN, JR, *On the approximate solution of hyperbolic initial-boundary value problems*, Thesis, Dept. Computer Science, Report No. STAN-CS-80-806, Stanford Univ., Stanford, CA, 1980.

[5] B. ENGQUIST AND T. SMEDSAAS, *Automatic computer code generation for hyperbolic and parabolic differential equations*, this Journal, 1 (1980), pp. 249–269.

[6] S. K. GODUNOV AND V. S. RYABENKIJ, *Spectral criteria for the stability of boundary problems for non-self-adjoint difference equations*, Uspekhi Mat. Nauk., 18, 3 (1963). (In Russian.)

[7] M. GOLDBERG AND E. TADMOR, *Scheme independent stability criteria for difference approximations of hyperbolic initial-boundary value problems.* I, Math. Comp., 32, (1978), pp. 1097–1107.

[8] ———, *Scheme-independent stability criteria for difference approximations of hyperbolic initial-boundary value problems.* II, Math. Comp., 36, (1981), pp. 603–626.

[9] ———, *Convenient stability criteria for difference approximations of hyperbolic initial boundary value problems*, 1983 AMS-SIAM Summer Seminar, "Large-Scale Computations in Fluid Mechanics", Scripps Institution of Oceanography, University of California, San Diego, to appear.

[10] B. GUSTAFSSON, *Stability analysis for initial-boundary value problems*, Numerical Methods, Proceedings, Caracas 1982, V. Pereyra and A. Reinoza, eds., Springer-Verlag, Berlin, 1983.

[11] B. GUSTAFSSON, H.-O. KREISS AND A. SUNDSTRÖM, *Stability theory of difference approximations for mixed initial boundary value problems.* II, Math. Comp., 26 (1972), pp. 649–686.

[12] ———, *Stability theory of difference approximations for mixed initial boundary value problems.* II, Uppsala University, Dept. of Computer Sciences, Report No. 30, (1971).

[13] B. GUSTAFSSON AND J. OLIGER, *Stable boundary approximations for implicit time discretizations for gas dynamics*, this Journal, 3 (1982), pp. 408–421.

[14] IMSL *Library*, edition 8, Reference Manual, 1980.

[15] H.-O. KREISS, *Difference approximations for the initial-boundary value problem for hyperbolic differential equations*, in Numerical Solution of Nonlinear Differential Equations (proceedings), John Wiley, New York, 1966, pp. 141–166.

[16] ———, *Stability theory for difference approximations of mixed initial boundary value problems.* I, Math. Comp., 22 (1968), pp. 703–714.

[17] D. LAZARD, *Résolution des systèmes d'équations algébriques*, Theor. Comp. Sci., 15 (1981), pp. 77-110.
[18] D. MICHELSON, *Stability theory of difference approximations for multidimensional initial-boundary value problems*, Math. Comp., 40 (1983), pp. 1-45.
[19] M. NORDSTRÖM, LISP-F3 *User's Guide*, Dept. Computer Sciences, Report No. DLU 78/4, Uppsala University, 1978.
[20] J. OLIGER, *Fourth order difference methods for the initial boundary-value problem for hyperbolic equations*, Math. Comp., 28, (1974), pp. 15-25.
[21] ———, *Hybrid difference methods for the initial boundary-value problems for hyperbolic equations*, Math. Comp., 30 (1976), pp. 724-738.
[22] S. OSHER, *Systems of difference equations with general homogeneous boundary conditions*, Trans. Amer. Math. Soc., 137 (1969), pp. 177-201.
[23] ———, *Stability of difference approximations of dissipative type for mixed initial-boundary value problems.* I, Math. Comp., 23 (1969), pp. 335-340.
[24] R. D. RICHTMYER AND K. W. MORTON, *Difference Methods for Initial-Value Problems*, John Wiley, New York, 1967.
[25] G. SKÖLLERMO, *How the boundary conditions affect the stability and accuracy of some implicit methods for hyperbolic equations*, Dept. Computer Sciences, Report No. 62, Uppsala University, 1975.
[26] D. M. SLOAN, *Boundary conditions for a fourth order hyperbolic difference scheme*, Math. Comp., 41 (1983), pp. 1-11.
[27] J. STRIKWERDA, *Initial boundary value problems for the method of lines*, J. Comp. Phys., 34 (1980), pp. 94-107.
[28] I. SWENSON, *Undersökning av en inbäddningsmetod för lösning av icke-lineära ekvationssystem i samband med stabilitetsundersökning av differensapproximationer av begynnelse-randvärdes-problem*, Dept. Computer Sciences, Internal Report No. 83, Uppsala University, 1983. (In Swedish.)
[29] M. THUNÉ, IBSTAB—*A software system for automatic stability analysis of difference methods for hyperbolic initial-boundary value problems*, Thesis, Dept. Computer Sciences, Report No. 93, Uppsala University, 1984.
[30] J. M. VARAH, *Stability of difference approximations to the mixed initial boundary value problems for parabolic systems*, SIAM J. Numer. Anal., 8 (1971), pp. 598-615.

SIAM J. SCI. STAT. COMPUT.
Vol. 7, No. 3, July 1986

018

THE NUMERICAL CALCULATION OF TRAVELING WAVE SOLUTIONS OF NONLINEAR PARABOLIC EQUATIONS*

THOMAS HAGSTROM† AND H. B. KELLER‡

Abstract. Traveling wave solutions have been studied for a variety of nonlinear parabolic problems. In the initial value approach to such problems the initial data at infinity determines the wave that propagates. The numerical simulation of such problems is thus quite difficult. If the domain is replaced by a finite one, to facilitate numerical computations, then appropriate boundary conditions on the "artificial" boundaries must depend upon the initial data in the discarded region. In this work we derive such boundary conditions, based on the Laplace transform of the linearized problems at $\pm\infty$, and illustrate their utility by presenting a numerical solution of Fisher's equation which has been proposed as a model in genetics.

Key words. artificial boundary conditions, parabolic traveling waves

AMS(MOS) subject classifications. 65M99, 35K15

1. Introduction. We develop numerical methods for computing traveling wave solutions of Cauchy problems for nonlinear parabolic equations posed on infinite spatial domains. In particular we consider the general one space dimensional case:

$$\begin{aligned} &\text{a)} \quad u_t = f(u_{xx}, u_x, u), \quad -\infty < x < \infty, \quad t > 0, \\ (1.1)\quad &\text{b)} \quad \lim_{x\to\pm\infty} u(x, t) = \varphi_\pm, \\ &\text{c)} \quad u(x, 0) = u_0(x). \end{aligned}$$

The equation (1.1a) is parabolic if, as we assume:

$$(1.1)\quad \text{d)} \quad \frac{\partial}{\partial a} f(a, b, c) \geqq 1 \quad \forall a, b, c \in \mathbb{R}.$$

Further we require that the constant states at $\pm\infty$ are compatible with smooth steady states, but that changes in their values effect the wave; thus:

$$(1.2)\quad \text{a)} \quad f(0, 0, \varphi_\pm) = 0, \qquad \text{b)} \quad f_u(0, 0, \varphi_\pm) \neq 0.$$

Finally the initial data is required to satisfy (1.1b); that is:

$$(1.2)\quad \text{c)} \quad \lim_{x\to\pm\infty} u_0(x) = \varphi_\pm.$$

In particular we wish to compute the evolution of initial data into traveling waves, that is into solutions of the form $u = w(x - ct)$.

Hagan [3], [4] has presented an extensive analysis of problem (1.1). We paraphrase some of his results below:

(i) Nonmonotonic waves (i.e. $w'(\xi)$ not of one sign) are unstable in general.

* Received by the editors May 24, 1984, and in revised form July 1, 1985. This research was sponsored in part by the U.S. Department of Energy under contract DE-AS03-76SF-00767, and by the U.S. Army under contract DAAG29-80-C-0041.

† Department of Applied Mathematics and Statistics, State University of New York at Stony Brook, Stony Brook, New York 11794.

‡ Department of Applied Mathematics, California Institute of Technology, Pasadena, California 91125.

(ii) The stability of monotonic waves of speed c can, in general, be determined by an examination of their trajectories in the phase plane of:

$$w' = v, \qquad f(v', v, w) + cv = 0. \tag{1.3}$$

(iii) If traveling waves exist, a large class of initial data, $u_0(x)$, satisfying (1.1b) will evolve to the same traveling wave.

(iv) Infinitely many wavespeeds, c, may be allowed. In this case, the traveling wave which evolves depends on the behavior of the initial data at infinity.

The numerical solution of (1.1) requires a finite computational domain. One way to obtain such a domain is to introduce artificial boundaries at the points $x = \tau_{\pm}$, $\tau_+ > \tau_-$. The derivation of such boundary conditions is the main subject of this work. A general theory of boundary conditions at an artificial boundary is given by the authors in [6]. This theory is not directly applicable to time dependent problems in unbounded spatial domains such as (1.1). However, a Laplace transformation in time yields a problem of the right form. In § 2, proper boundary conditions are derived for the transformed problem and they are inverted to yield conditions on the direct problem.

We note that use of the proper boundary conditions is crucial whenever (iv) holds. Then, the "naive" conditions, replacing (1.1b) by:

$$u(\tau_{\pm}, t) = \varphi_{\pm}, \tag{1.4}$$

must, in general, fail to lead to the correct long time solution.

In § 3 a specific problem of the form (1.1) is introduced: the Cauchy problem for Fisher's equation. It has traveling wave solutions of all speeds $c \geqq 2$. Gazdag and Canosa [1] present a numerical solution of Fisher's equation using boundary conditions analogous to (1.4). As predicted by the theory, their solution always evolved to the traveling wave of minimum speed, $c = 2$. In order to calculate traveling waves with speeds $c \geqq 2$, we employ the boundary conditions derived in § 2. The numerical solution evolves to the correct traveling wave for a variety of choices of initial data.

We note that the method of deriving boundary conditions presented here can be applied to other time dependent problems, including some problems of hyperbolic type. For other examples the reader is referred to Gustafsson and Kreiss [2] and Hagstrom [5].

2. Construction of the boundary conditions. We construct boundary conditions at the right-hand boundary, $x = \tau_+$. The construction at the left will be analogous. Although the general results of [6] are not directly applicable to this problem, they do motivate our approach and, hence, we summarize them below.

For a linear problem the exact boundary condition at the artificial boundary is characterized in the following way: the Cauchy data at the artificial boundary, that is the function along with the correct number of its normal derivatives, should be an element of a certain affine set. This affine set is determined by the subspace of Cauchy data which leads to solutions of the homogeneous problem in the tail, $x \geqq \tau_+$, combined with the trace of any particular solution of the inhomogeneous problem. For certain nonlinear problems whose solution is required to have a smooth limit at infinity, the existence of an exact nonlinear boundary condition is established. Here it is necessary to assume that the solution at the artificial boundary is sufficiently close to its limiting value. An asymptotic expansion of the boundary condition is derived, the first term of which consists of the boundary condition for the problem linearized about the limiting solution.

Given this last fact about nonlinear problems, it is reasonable to analyze the problem in the tail linearized about φ_+. We assume a coordinate system moving to the

right with speed c and thus we consider:

$$\begin{aligned} &\text{a)} \quad v(x,t) = u(x,t) - \varphi_+, \qquad x \geqq \tau_+, \\ &\text{b)} \quad v_t = f_1 v_{xx} + f_2 v_x + c v_x + f_3 v, \\ &\text{c)} \quad v(x,0) = u_0(x) - \varphi_+, \\ &\text{d)} \quad \lim_{x\to\infty} v(x,t) = 0. \end{aligned} \tag{2.1}$$

Here the constants f_i are given by:

$$f_1 = \frac{\partial f}{\partial(u_{xx})}(0,0,\varphi_+), \quad f_2 = \frac{\partial f}{\partial(u_x)}(0,0,\varphi_+), \quad f_3 = \frac{\partial f}{\partial u}(0,0,\varphi_+). \tag{2.2}$$

Following the discussion above, to derive boundary conditions from (2.1) two problems must be solved; boundary conditions for the homogeneous problem, (2.1b, d) combined with zero initial data, must be found as well as a particular solution of (2.1b, d) which satisfies (2.1c). The homogeneous problem is considered first.

2.1. Boundary conditions for the homogeneous problem. We introduce the temporal Laplace transform:

$$\hat{w}(x,s) = \int_0^\infty e^{-st} w(x,t)\, dt.$$

If w is a solution of (2.1b, d) with zero initial data, then $\hat{w}$ satisfies:

$$\begin{aligned} &\text{a)} \quad f_1 \hat{w}_{xx} + (f_2 + c)\hat{w}_x + (f_3 - s)\hat{w} = 0, \\ &\text{b)} \quad \lim_{x\to\infty} \hat{w}(x,s) = 0. \end{aligned} \tag{2.3}$$

Equation (2.3a) has the basic exponential solutions:

$$\hat{w} = e^{\kappa_\pm(s;c)x}, \tag{2.4}$$

where $\kappa_\pm(s;c)$ are given by:

$$\kappa_\pm = \frac{-(f_2+c)}{2f_1} \pm \frac{1}{2f_1}[(f_2+c)^2 + 4f_1(s-f_3)]^{1/2}. \tag{2.5}$$

For $\mathrm{Re}\,(s)$ sufficiently large, $\mathrm{Re}\,(\kappa_+) \geqq 0$ and $\mathrm{Re}\,(\kappa_-) \leqq 0$, since $f_1 \geqq 1$. Hence, the admissible solution has exponent κ_- and it satisfies:

$$\hat{w}_x(\tau_+; s) = \kappa_-(s;c)\hat{w}(\tau_+; s). \tag{2.6}$$

This can be rewritten as:

$$\frac{\hat{w}_x(\tau_+; s)}{\kappa_-(s;c)} = \hat{w}(\tau_+; s). \tag{2.7}$$

Using the convolution formulas and the expression for the inverse transform of $1/\kappa_-$ (see, e.g., Oberhettinger and Badii [9]), (2.7) yields for $w(\tau_+, t)$ the condition:

$$-\sqrt{f_1}\int_0^t \left\{ e^{[f_3-\alpha^2](t-p)} \left[\frac{1}{\sqrt{\pi(t-p)}} - \alpha\, e^{\alpha^2(t-p)}\, \mathrm{Erfc}\,(\alpha\sqrt{t-p}) \right] w_x(\tau_+, p) \right\} dp = w(\tau_+, t) \tag{2.8}$$

where we have introduced:

$$\alpha \equiv \frac{f_2 + c}{2\sqrt{f_1}}.$$

2.2. Particular solution. We now find particular solutions in the case that $u_0(x) - \varphi_+$ is expressed as a finite sum of exponentials:

$$u_0(x) - \varphi_+ \equiv \sum_{j=1}^{N} d_j e^{-\alpha_j(x-\tau_+)}, \qquad \alpha_j > 0, \quad x \geqq \tau_+. \tag{2.9}$$

Hagan's analysis [3] shows that $u_0(x) - \varphi_+$ must decay at least exponentially for traveling wave solutions to exist. From (2.5), with $s = 0$, we see that traveling waves of speed c have exponential decay rates given by:

$$g_\pm(c) = -\frac{f_2 + c}{2f_1} \pm \frac{1}{2f_1}[(f_2 + c)^2 - 4f_1 f_3]^{1/2}. \tag{2.10}$$

Thus for any exponent, $-\alpha_j$, there exists a unique speed c_j such that:

$$-\alpha_j = g_-(c_j) \quad \text{or} \quad -\alpha_j = g_+(c_j). \tag{2.11a}$$

This speed is given by:

$$c_j = \frac{f_3}{\alpha_j} + f_1\alpha_j - f_2. \tag{2.11b}$$

Hence, each exponential in (2.9) can be associated with a unique traveling wave, from which a particular solution can be found. This gives the particular solution corresponding to the initial data (2.9):

$$v_p(x, t) = \sum_{j=1}^{N} d_j e^{-\alpha_j([x-\tau_+]-(c-c_j)t)}. \tag{2.12}$$

Combining (2.8) and (2.12) yields the following linearized boundary condition on $u \equiv w - v_p$ at $x = \tau_+$:

$$\begin{aligned} -\sqrt{f_1}\int_0^t \left\{ e^{[f_3-\alpha^2](t-p)} \left[\frac{1}{\sqrt{\pi(t-p)}} - \alpha\, e^{\alpha^2(t-p)} \operatorname{Erfc}(\alpha\sqrt{t-p}) \right] \right. \\ \left. \cdot \left(u_x(\tau_+, p) + \sum_{j=1}^{N} \alpha_j d_j e^{\alpha_j(c_j-c)p} \right) \right\} dp \\ = u(\tau_+, t) - \varphi_+ - \sum_{j=1}^{N} d_j e^{\alpha_j(c_j-c)t}. \end{aligned} \tag{2.13}$$

Here of course:

$$\begin{aligned} &\text{a)} \quad \alpha = \frac{f_2 + c}{2\sqrt{f_1}}, \\ &\text{b)} \quad u_0(x) = \varphi_+ + \sum_{j=1}^{N} d_j e^{-\alpha_j(x-\tau_+)}, \qquad x \geqq \tau_+. \end{aligned} \tag{2.14}$$

2.3. Conditions at the left boundary. A similar boundary condition can be derived at the left-hand boundary, $x = \tau_-$. In transform variables, a solution to the linearized,

homogeneous problem on $(-\infty, \tau_-]$ must satisfy:

$$\frac{\hat{w}_x(\tau_-; s)}{\bar{\kappa}_+(s; c)} = \hat{w}(\tau_-; s), \tag{2.15}$$

where

$$\bar{\kappa}_+ = \frac{-(\bar{f}_2 + c)}{2\bar{f}_1} + \frac{1}{2\bar{f}_1}[(\bar{f}_2 + c)^2 + 4\bar{f}_1(s - \bar{f}_3)]^{1/2} \tag{2.16}$$

and

$$\bar{f}_1 = \frac{\partial f}{\partial(u_{xx})}(0, 0, \varphi_-), \quad \bar{f}_2 = \frac{\partial f}{\partial(u_x)}(0, 0, \varphi_-), \quad \bar{f}_3 = \frac{\partial f}{\partial u}(0, 0, \varphi_-). \tag{2.17}$$

The inverse transform of (2.15) is given by:

(2.18)

$$\sqrt{\bar{f}_1}\int_0^t \left\{ e^{[\bar{f}_3 - \bar{\alpha}^2](t-p)}\left[\frac{1}{\sqrt{\pi(t-p)}} + \bar{\alpha}\, e^{\bar{\alpha}^2(t-p)}\,\mathrm{Erfc}\,(-\bar{\alpha}\sqrt{t-p})\right] w_x(\tau_-, p)\right\} dp = w(\tau_-, t)$$

with $\bar{\alpha} = (\bar{f}_2 + c)/1\sqrt{\bar{f}_1}$. To find a particular solution, we assume that

$$u_0(x) - \varphi_- = \sum_{j=1}^{M} \bar{d}_j\, e^{\bar{\alpha}_j(x - \tau_-)}, \qquad \bar{\alpha}_j > 0, \quad x \leqq \tau_-.$$

Each exponent, $\bar{\alpha}_j$, can be uniquely associated with a linear traveling wave of speed $\bar{c}_j$ through (2.16), with $s = 0$:

$$\bar{c}_j = -\bar{f}_2 - \bar{f}_1\bar{\alpha}_j - \frac{\bar{f}_3}{\bar{\alpha}_j}. \tag{2.19}$$

This leads to the particular solution:

$$\bar{V}_p(x, t) = \varphi_- + \sum_{j=1}^{M} \bar{d}_j\, e^{-\bar{\alpha}_j\tau_-}\, e^{\bar{\alpha}_j(x + (c - \bar{c}_j)t)}. \tag{2.20}$$

Combining (2.20) with (2.18) yields a linear boundary condition at τ_-, analogous to (2.13):

$$\begin{aligned}\sqrt{\bar{f}_1}\int_0^t e^{[\bar{f}_3 - \bar{\alpha}^2](t-p)}\left[\frac{1}{\sqrt{\pi(t-p)}} + \bar{\alpha}\, e^{\bar{\alpha}^2(t-p)}\,\mathrm{Erfc}\,(-\bar{\alpha}\sqrt{t-p})\right] \\ \cdot \left(u_x(\tau_-, p) - \sum_{j=1}^{M} \bar{\alpha}_j\bar{d}_j\, e^{\bar{\alpha}_j(c - \bar{c}_j)p}\right) dp \\ = u(\tau_-, t) - \varphi_- - \sum_{j=1}^{M} \bar{d}_j\, e^{\bar{\alpha}_j(c - c_j)t}.\end{aligned} \tag{2.21}$$

Here

$$\begin{aligned} &\text{a)} \quad \bar{\alpha} = \frac{\bar{f}_2 + c}{2\sqrt{\bar{f}_1}}, \\ &\text{b)} \quad u_0(x) = \varphi_- + \sum_{j=1}^{M} \bar{d}_j\, e^{\bar{\alpha}_j(x - \tau_-)}, \qquad x \leqq \tau_-. \end{aligned} \tag{2.22}$$

3. Application to Fisher's equation. We now apply the results of the preceding section to Fisher's equation:

$$\begin{aligned} &\text{a)} \quad u_t = u_{xx} + u(1-u), \qquad x \in (-\infty, \infty), \quad t \geqq 0, \\ (3.1) \qquad &\text{b)} \quad \lim_{x\to\infty} u(x,t) = 0, \qquad \lim_{x\to-\infty} u(x,t) = 1, \\ &\text{c)} \quad u(x,0) = u_0(x). \end{aligned}$$

Problem (3.1) is used as a model of the propagation of an advantageous gene. For a discussion of this application see, for example, Moran [8]. It is a special case of (1.1) and various statements concerning the behavior of its solution are consequences of Hagan's [3] general analysis. (They were first proved by Larson [7].)

(i) There exist traveling wave solutions of all wavespeeds $c \geqq 2$.

(ii) All positive initial data, $u_0(x)$, decaying at least exponentially as $x \to \infty$ evolves to a unique traveling wave.

(iii) If $u_0(x) \sim e^{-\beta x}$ as $x \to \infty$, then the solution evolves to a wave of speed $c(\beta)$ given by:

$$(3.2) \qquad c(\beta) = \begin{cases} \dfrac{1+\beta^2}{\beta}, & \beta \leqq 1, \\ 2, & \beta \geqq 1. \end{cases}$$

The linearized boundary conditions, (2.13) and (2.21), are easily specialized to this problem. As in § 2, we introduce a coordinate system moving to the right with speed c and choose τ_+ and τ_- as our artificial boundary locations. We assume that $u_0(x)$ can be represented as a finite sum of exponentials in the tails:

$$(3.3) \qquad \begin{aligned} u_0(x) &= \sum_{j=1}^{N} d_j e^{-\alpha_j(x-\tau_+)}, \qquad x \geqq \tau_+, \\ u_0(x) &= \sum_{j=1}^{M} \bar{d}_j e^{\bar{\alpha}_j(x-\tau_-)} + 1, \qquad x \leqq \tau_-. \end{aligned}$$

The boundary conditions we impose are:

$$(3.4) \qquad \begin{aligned} &\text{a)} \quad -\int_0^t e^{(1-c^2/4)(t-p)} \left[\frac{1}{\sqrt{\pi(t-p)}} - \frac{c}{2} e^{c^2(t-p)/4} \operatorname{Erfc}\left(\frac{c}{2}\sqrt{t-p} \right) \right] \\ &\qquad \cdot \left[u_x(\tau_+, p) + \sum_{j=1}^{N} \alpha_j d_j e^{(1+\alpha_j^2 - \alpha_j c)p} \right] dp \\ &\qquad = u(\tau_+, t) - \sum_{j=1}^{N} d_j e^{(1+\alpha_j^2-\alpha_j c)t}, \\ &\text{b)} \quad \int_0^t e^{-(1+c^2/4)(t-p)} \left[\frac{1}{\sqrt{\pi(t-p)}} + \frac{c}{2} e^{c^2(t-p)/4} \operatorname{Erfc}\left(-\frac{c}{2}\sqrt{t-p} \right) \right] \\ &\qquad \cdot \left[u_x(\tau_-, p) - \sum_{j=1}^{M} \bar{\alpha}_j \bar{d}_j e^{(\bar{\alpha}_j^2 + \bar{\alpha}_j c - 1)p} \right] dp \\ &\qquad = u(\tau_-, t) - 1 - \sum_{j=1}^{M} \bar{d}_j e^{(\bar{\alpha}_j^2 + \bar{\alpha}_j c - 1)t}. \end{aligned}$$

We note from (3.3) that the true solution should evolve to a wave of speed $c(\beta)$ given

by (3.2) where

$$\beta = \min_j \{\alpha_j\}. \tag{3.5}$$

In certain circumstances the particular terms in (3.4a) have a large exponential growth in time. As this could be a source of error in a numerical computation, the integrals involving them were done exactly. This allows us to rewrite the right boundary condition:

$$\begin{aligned} &-\int_0^t e^{(1-c^2/4)(t-p)}\left[\frac{1}{\sqrt{\pi(t-p)}}-\frac{c}{2}e^{c^2(t-p)/4}\operatorname{Erfc}\left(\frac{c}{2}\sqrt{t-p}\right)\right]u_x(\tau_+, p)\,dp \\ &\quad = u(\tau_+, t)+\sum_{j=1}^{N} f_j(t). \end{aligned} \tag{3.4a'}$$

Here

$$f_j(t) = \alpha_j d_j\left[\frac{-c}{2(1-k_j)}e^t \operatorname{Erfc}\left(\frac{c}{2}\sqrt{t}\right)\right.$$

$$\left.+\frac{[k_j+c^2/4-1]^{1/2}}{(1-k_j)}e^{k_j t}\operatorname{Erfc}\left(\sqrt{\left(k_j+\frac{c^2}{4}-1\right)t}\right)+h_j(t)\right],$$

$$h_j(t)=\begin{cases} 0, & \alpha_j=\dfrac{c}{2}+\sqrt{\dfrac{c^2}{4}+k_j-1}, \\ -2\dfrac{\sqrt{c^2/4+k_j-1}}{(1-k_j)}e^{k_j t}, & \alpha_j=\dfrac{c}{2}-\sqrt{\dfrac{c^2}{4}+k_j-1}, \end{cases} \tag{3.6}$$

and

$$k_j = 1+\alpha_j^2-\alpha_j c.$$

We note that (3.4a') explicitly contains the different evolution of initial data with large and small decay rates.

Presented below are the results of some numerical computations of solutions of (3.1) using the boundary conditions (3.4a', b). A uniform grid was introduced and spatial derivatives were replaced by centered finite differences. The method was implicit in time and stable for the ratio of the time step to the grid size sufficiently small. With h denoting the spatial mesh width and k the time step we used:

$$\begin{aligned} &u(x, t+k)-kD_+D_-u(x,t+k)-ku(x, t+k)(1-u(x, t+k)) \\ &\quad = u(x, t-k)+kD_+D_-u(x, t-k)+u(x, t-k)(1-u(x, t-k))+2ckD_0u(x, t). \end{aligned}$$

Here D_+D_- and D_0 are standard spatial difference operators:

$$D_+D_-f(x)=\frac{(f(x+h)-2f(x)+f(x-h))}{h^2}, \qquad D_0f(x)=\frac{(f(x+h)-f(x-h))}{2h}.$$

At each step the nonlinear system of difference equations was solved using Newton's method with an explicit step taken to generate the initial guess.

The boundaries were located midway between gridpoints and the integrals there were approximated by the trapezoid rule (away from the singularity). So, for example,

at the right-hand boundary we used:

$$-\left[\frac{k}{2}(K(t)+K_s(t))w(0)+k\sum_{j=1}^{(t/k-2)}(K(t-jk)+K_s(t-jk))w(jk)\right.$$
$$+kK(k)w(t-k)+\frac{k}{2}K(0)w(t)+\frac{k}{2}K_s(k)w(t-k)$$
$$\left.+\int_{t-k}^{t}\{(t-p)w(t-k)\,e^{(1-c^2/4)}-(t-k-p)w(t)\}\frac{dp}{\sqrt{\pi(t-p)}}\right]$$
$$=\frac{1}{2}\left\{u\left(\tau_+ +\frac{h}{2},t\right)+u\left(\tau_+ -\frac{h}{2},t\right)\right\}+\sum_{j=1}^{N}f_j(t).$$

Here, K and K_s are respectively the singular and nonsingular parts of the kernel in (3.4a'), the f_j are defined in (3.6) and $w(t)$ is given by:

$$w(t)=\frac{1}{h}\left[u\left(\tau_+ +\frac{h}{2},t\right)-u\left(\tau_+ -\frac{h}{2},t\right)\right].$$

The remaining integral was done analytically.

For all cases described below the grid ranged between -12 and 12 and contained 171 points. The time step is .025, well within the stable region in all cases. Initial conditions were generated in the following way: expansions in the tail, (3.3), were input and smoothly connected (two continuous derivatives) by a combination of polynomial and exponential functions. The computations shown were performed on a VAX 11/780 at the University of Wisconsin at Madison, though others were done on the IBM 4341 of the Applied Mathematics Department at the California Institute of Technology.

Figure 1 shows the evolution, in a coordinate system moving with speed 4, of initial data which decays, at both $\pm\infty$, at a rate compatible with a wave of speed 4.

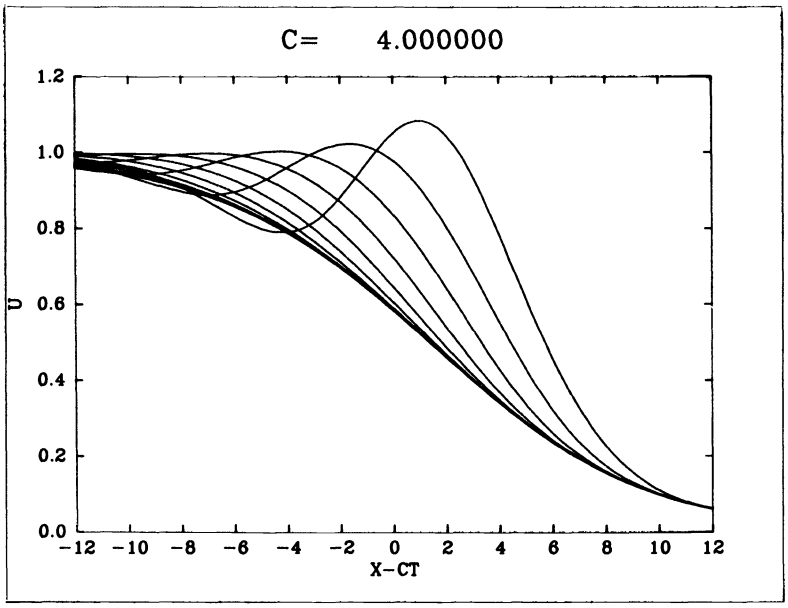


FIG. 1

The initial data and solutions at intervals of 25 time steps are displayed. A steady state is reached which must be moving with speed 4. Figure 2 contains the final state (solid line) of Fig. 1. This is the solution at $t = 6.875$. Plotted with it is the traveling wave solution of speed 4. This solution was found by a second order centered finite difference solution of the relevant steady problem on the same mesh as used in the time dependent case. The solution was required to agree with the time dependent one at the right-hand boundary while the condition:

$$u_x = \bar{\kappa}_+(0; 4)(u - 1)$$

was used at τ_-. (Note that $\bar{\kappa}_+(s; c)$ is defined in (2.16).) The agreement between the two solutions is seen to be excellent.

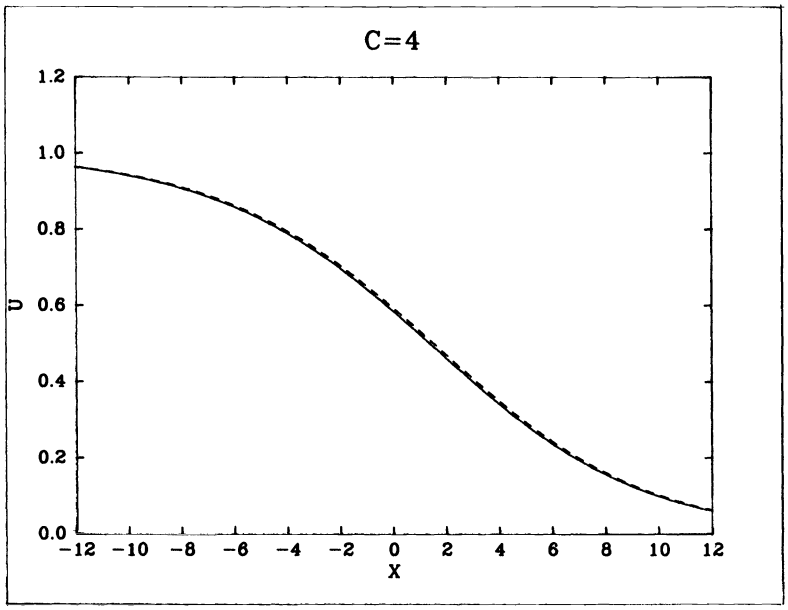


FIG. 2

We note that the boundary condition,

$$u(\tau_+, t) = \text{constant}, \tag{3.7}$$

leads to good results when the speed of the coordinate system is the same as the speed of the final state. For a more complicated problem, however, this might not be known in advance. Indeed, it might be the goal of the computation to determine it. As shown in Fig. 3, our conditions avoid this difficulty. This is the computed evolution in a coordinate system moving with speed 3 of the same initial data used to generate Fig. 1. The wave is seen to move to the right and, in fact, moves with relative speed 1. This is confirmed in Fig. 4, a comparison of the solution at $t = 6.875$ (solid line) and the wave of speed 4 of Fig. 2 translated to the right a distance of 6.875. We believe the small error at the right boundary is due to the use of linearized boundary conditions.

Figure 5 displays the computed evolution, in a coordinate system moving with speed 4, of initial data with two decay rates in the right tail: one compatible with a wave of speed 4, the other compatible with a wave of speed 3. Here, the speed 4 part decayed at the large rate while the speed 3 part decayed at the slow rate. As predicted by the theory, a speed 3 wave is eventually reached.

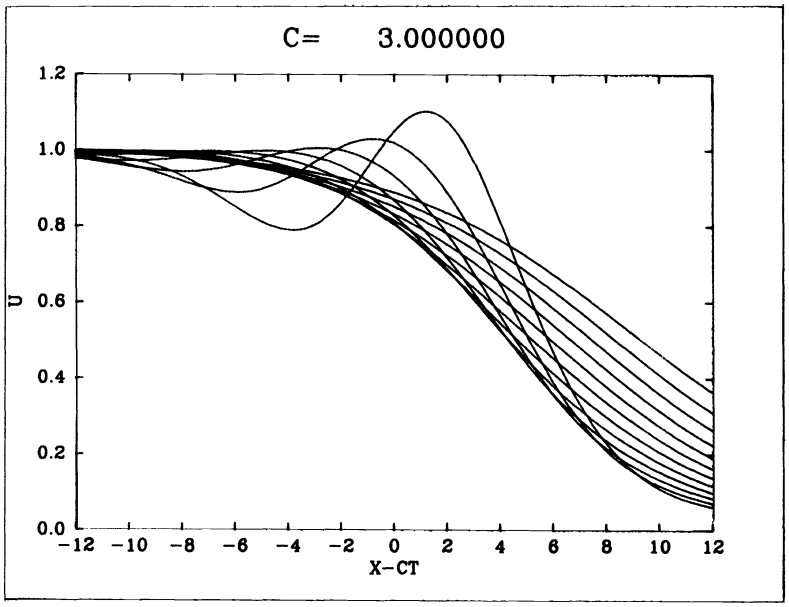


FIG. 3

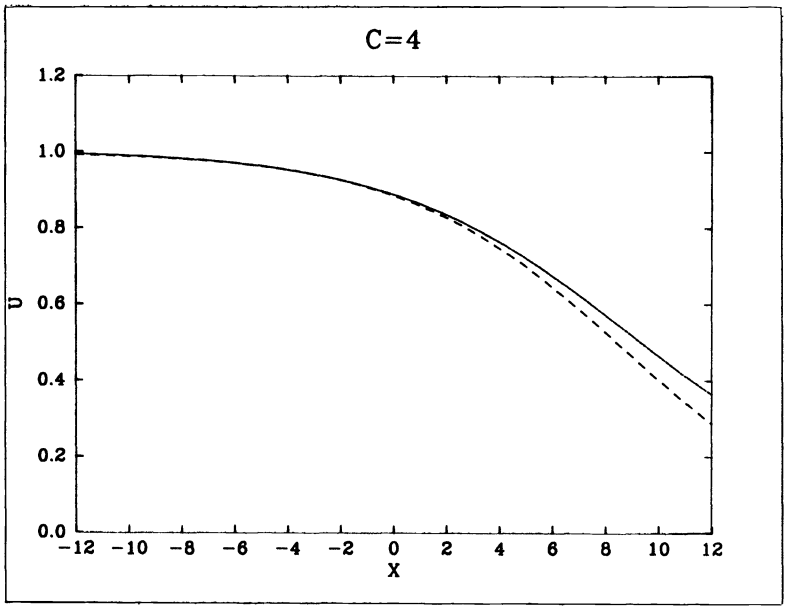


FIG. 4

We note that, as it is the initial data in the right tail which determines the wavespeed, it is the right-hand boundary condition which is important. Various choices for the left-hand boundary condition, for example $u =$ constant and $u_x = 0$, were tried and led to good results.

In summary, we have shown that our boundary conditions consistently lead to correct long-time results while other simpler conditions do not. We believe that their

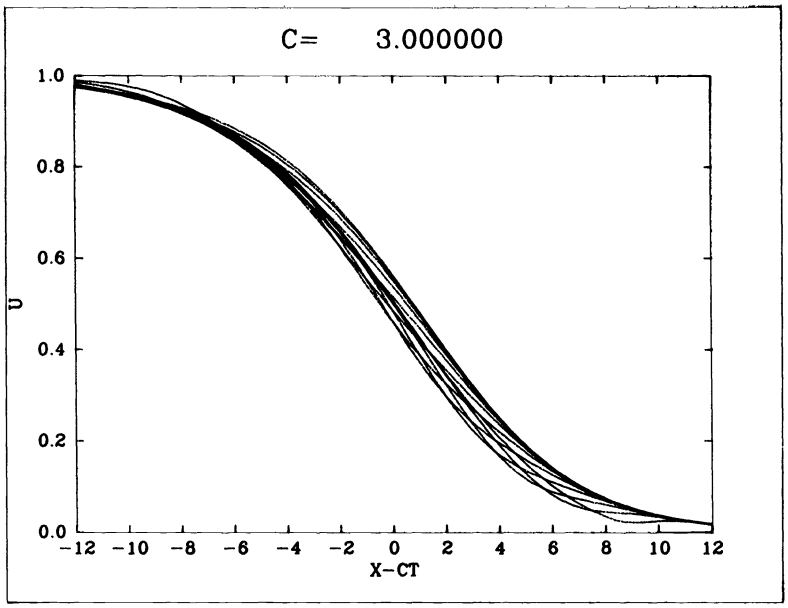


FIG. 5

generalization to more complicated problems, where the final state is not known a priori, will also give reliable results. It should be noted, however, that this has not yet been proved even in the simple case described here.

Acknowledgment. The authors thank Prof. J. D. Murray for bringing this problem to our attention.

REFERENCES

[1] J. GAZDAG AND J. CANOSA, *Numerical solution of Fisher's equation*, J. Appl. Probl., 11 (1974), pp. 445-457.

[2] B. GUSTAFSSON AND H.-O. KREISS, *Boundary conditions for time dependent problems with an artificial boundary*, J. Comput. Phys., 30 (1979), pp. 333-351.

[3] P. HAGAN, *Traveling wave and multiple traveling wave solutions of parabolic equations*, SIAM J. Math. Anal., 5 (1982), pp. 717-738.

[4] ———, *The instability of non-monotonic wave solutions of parabolic equations*, Stud. Appl. Math. (1981), pp. 57-88.

[5] T. HAGSTROM, *Reduction of unbounded domains to bounded domains for partial differential equation problems*, Ph.D. thesis, California Institute of Technology, Pasadena, 1983.

[6] T. HAGSTROM AND H. B. KELLER, *Exact boundary conditions at an artificial boundary for partial differential equations in cylinders*, SIAM J. Math. Anal., to appear.

[7] D. A. LARSON, *Transient bounds and time-asymptotic behavior of solutions to nonlinear equations of Fisher type*, SIAM J. Appl. Math., 34 (1978), pp. 93-103.

[8] P. MORAN, *The Statistical Processes of Evolutionary Theory*, Clarendon Press, Oxford, 1962, pp. 178-186.

[9] F. OBERHETTINGER AND L. BADII, *Tables of Laplace Transforms*, Springer-Verlag, New York, 1973.

SIAM J. SCI. STAT. COMPUT.
Vol. 7, No. 3, July 1986

019

THE NUMERICAL SCHWARZ ALTERNATING PROCEDURE AND SOR*

LOUIS W. EHRLICH†

Abstract. The Numerical Schwarz Alternating Procedure is accelerated on several sample regions using a block SOR technique where the relaxation factor is determined using the power method to approximate the appropriate eigenvalue.

Key words. Schwarz alternating, SOR, power method, relaxation.

1. Introduction. Almost a hundred years ago, H. A. Schwarz [10] introduced an idea that has been used over the years to solve or prove the existence of solutions of Dirichlet problems for Laplace's equation over regions composed of regular geometric shapes pieced together. In effect, one decomposes the region into these regular regions with overlapping or abutting boundaries. The equations are solved in each of the regular regions alternately, with the values on the "interior" boundaries of the regular regions determined iteratively.

To this writer's knowledge, the first application of this approach to numerically solving problems was discussed by K. Miller [8]. The idea then apparently lay dormant for about 15 years. Suddenly, recently, applications burst forth [1], [3], [4], [5], [6], [9], [12]. The reason for this is probably the emergence of parallel computing, but the emergence of the personal computer may also have been a factor.

In the literature, the idea is called the Numerial Schwarz Algorithm [8], [9], domain decomposition [3], [6], [13], domain partitioning [1], [12], etc. It apparently is implemented in several ways. One can overlap the regular regions [4], [9], or one can abut the regions with no overlap [1], [12]. Each of the regions can be solved numerically, independent of adjacent regions. In some cases finite differences are used [4], [5], [9], and in others, finite elements [1], [3], [6], [12]. The interconnection of the regions may be through grid points common to both (abutment) or through some sort of interpolation (abutment or overlap).

Since each iteration requires solutions of the Laplace equation in certain regions, a variety of suggestions have been given to solve the resulting system. In [1], [3], [6], [12], conjugate gradient type techniques are suggested. In [9], an incomplete-factorization technique was suggested. It was also pointed out in [9] that, as usually implemented, the method is a block Gauss-Seidel method. As such, it should be amenable to acceleration. This idea was mentioned in [4]. Also in [4], [5] the idea of decomposing the region into regions that fast Poisson solvers can handle was presented. We have used that idea here in conjunction with using a block SOR method. It should be pointed out that some of these ideas may appear in [7], but that report is unavailable to this author.

2. Numerical implementation.

a. Overlap. To fix ideas, let us consider the Dirichlet problem in a rectangle using finite differences. We propose to split or decompose the rectangle into two overlapping rectangles as per Fig. 1, where (a) is the right boundary of rectangle (A), and (b) is the left boundary of rectangle (B).

* Received by the editors March 25, 1985, and in revised form July 15, 1985.
† Applied Physics Laboratory, The Johns Hopkins University, Laurel, Maryland 20707.

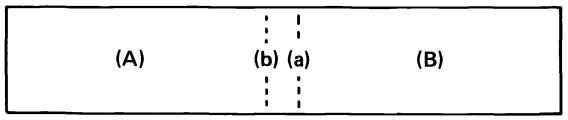


FIG. 1. *Region overlap.*

The usual approach is the following. Since one of the rectangles will have at least one boundary within the other, guess the solution along this boundary (say (a)) and solve rectangle A. Alternate to the other rectangle (B) and solve using the values along its unknown boundary (b) that were computed in (A). If the grid points do not coincide, some sort of interpolation scheme can be used. In any event, using the values which were just computed in (A) leads us to the Gauss-Seidel approach rather than the Jacobian method of using the values of the previous iteration. Repeat.

This is effectively an iterative process for the unknowns which are on the interior "boundary" lines. Considered as a block method, it has the required properties to apply block SOR. The problem is determining the optimum relaxation factor. This can be done by applying the power method to the problem to determine the largest eigenvalue of the block Jacobi matrix. Actually the block Gauss-Seidel method was used since its eigenvalues are the squares of those of the Jacobi and hence should lead to faster convergence of the power method due to better separation of the eigenvalues.

b. Abutting. Here we assume the region is split into two (or more) regions with only boundaries overlapping, as in Fig. 2.

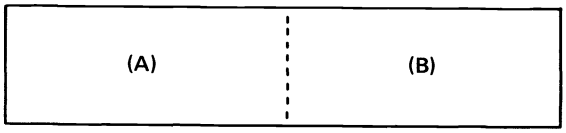


FIG. 2. *Region abutment.*

This leads to a natural decomposition of the resulting linear system, i.e.,

$$\begin{aligned} Ax_1 + Ix_a &= b_1, \\ Ix_1 + Dx_a + Ix_2 &= b_a, \\ Ix_a + Ax_2 &= b_2, \end{aligned} \tag{1}$$

where x_a is the solution along the common boundary to (A) and (B). Eliminating x_1 and x_2, we end up with a linear system for x_a involving the Schur complement matrix [2]. [1] and [12] discuss this and suggest a conjugate gradient approach. Equation (1) also lends itself to a block SOR method which we are proposing here as another consideration.

3. Numerical results. We considered first the Laplace equation in a rectangle. A 32×64 grid was superimposed over the region with the usual 5 point finite difference equation. Both overlapping and abutment were tried. The rectangular subregions during each iteration were solved using a direct solver—POIS by Swarztrauber and Sweet [11], thus avoiding any interplay between the iterative method of the Schwarz technique and that of solving the Laplace equation. The block SOR relaxation factor, ω, was determined in each case by using the power method on the Gauss-Seidel iteration

technique to find μ, the spectral radius. The factor was then determined from

$$\omega = \frac{2}{(1+\sqrt{1-\mu})}.$$

Table 1 contains some of the results. Iteration continued until two successive approximations differed by less than 10^{-6}.

TABLE 1

Rows of overlap	BSOR ($\omega = 1$) Iters.	Power method Iters.	Power method (ω)	BSOR (ω) Iters.
1 (32)*	119	7	1.52	27
1 (15)	103	7	1.50	25
1 (48)	97	7	1.49	24
2	66	6	1.40	20
3	35	5	1.27	14
15	7	3	1.02	6
30	4	3	1.00	4

* Abutments at row indicated in ( ).

Several things are noticeable from the table. First, the greater the ovelap, the more rapid the convergence, dramatically (see also [4]). Second, with minimal overlap, the BSOR method becomes cost effective. Clearly, if storage permits, the region should be solved as a single region. However, parallel computation may affect this decision. Also, small personal computers may be used in solving the various regions.

To illustrate the surprising effectiveness of the Numerical Schwarz Alternating process, the region in Fig. 3 was considered.

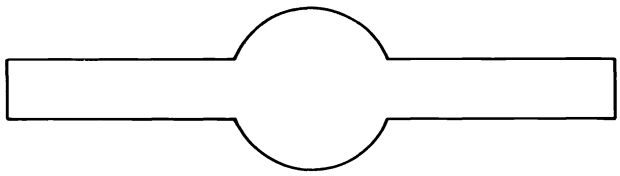


FIG. 3. *Sample region.*

This region was decomposed two different ways and the effectiveness of the Alternating technique compared. Figures 4 and 5 illustrate the two decompositions. Tables 2 and 3 compare the effectiveness of the methods.

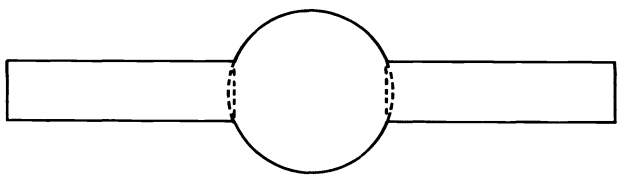


FIG. 4. *First decomposition of sample region.*

In the rectangles, rectangular finite difference equations were used and solved by the direct method cited above. In the circle, the Laplace equation was written in polar coordinates and solved using polar finite differences and the direct method of PWSPLR

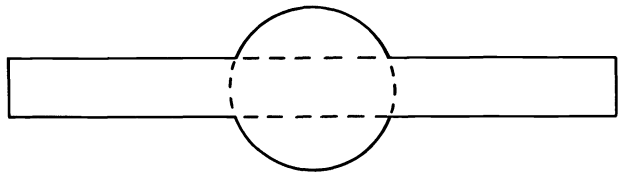


FIG. 5. *Second decomposition of sample region.*

TABLE 2
($\varepsilon = 10^{-10}$)

Figure	$\omega = 1$ Iter.	Time sec.	Power method Iter.	ω	ω Iters.	Time sec.
4	44	5.85	4	1.21	22+4	3.43
5	9	1.33	3	1.009	8+3	1.63

TABLE 3
($\varepsilon = 10^{-6}$)

Rect. M	mesh N	Polar o	mesh r	$\omega = 1$ Iter.	Time sec.	Power meth. Iter.	ω	ω Iter.	Time sec.
Data for Fig. 4									
16	16	60	16	26	3.46	7	1.2	13+7	2.66
32	32	60	32	24	7.51	3	1.19	12+3	4.7
32	32	60	64	24	12.9	3	1.19	12+3	8.04
64	64	60	64	24	20.4	3	1.18	12+3	12.77
Data for Fig. 5									
16	64	60	16	7	1.02	3	1.01	6+3	1.31
32	127	60	32	7	2.77	3	1.01	6+3	3·56
32	127	60	64	7	4.4	3	1.01	6+3	5.6
64	127	60	64	6	5.2	3	1.01	6+3	7.8

[11]. The approximations to the "interior" boundaries were determined by linear interpolation, but any higher interpolation scheme would probably have worked as well. Iteration continued until values along the "interior" boundaries agreed to within ε as noted in the tables. Comparison of the solutions of Figs. 4 and 5 shows 3 and 4 significant digit agreement for the finest mesh used.

4. Comments. The Numerical Schwarz Alternating technique appears to be a very effective method to solve a problem in pieces of a region if one cannot handle the entire region at once. Also, speedup may be possible by parallel computation of the pieces simultaneously. This report suggests another approach to piecewise computation, i.e., block SOR.

REFERENCES

[1] P. E. BJORSTAD AND O. B. WIDLUND, *Solving elliptic problems on regions partitioned into substructures,* in Elliptic Problem Solvers II, G. Birkhoff and A. Schoenstadt, eds., Academic Press, New York, 1984, pp. 245-255.

[2] R. W. COTTLE, *Manifestations of the Schur complement*, Linear Algebra Appl., 8 (1974), pp. 189-211.
[3] Q. V. DIHN, R. GLOWINSKI AND J. PÉRIAUX, *Solving elliptic problems by domain decomposition methods with applications*, in Elliptic Problem Solvers II, G. Birkhoff and A. Schoenstadt, eds., Academic Press, New York, 1984, pp. 395-426.
[4] G. ENDEN, U. DINNAR AND M. ISRAELI, *A numerical simulation of* 2-D *flow in a T-type bifurcation*, presented to Third International Conference on Numerical Methods in Laminar and Turbulent Flow, Univ. Washington, Seattle, 1983.
[5] G. ENDEN, M. ISRAELI AND U. DINNAR, *A numerical simulation of the flow in a T-type bifurcation and its application to "end to side" fistula*, submitted to ASME J. Biomechanical Engrg., 1984.
[6] R. GLOWINSKI, Q. V. DINH AND J. PRIAUX, *Domain decomposition methods for nonlinear problems in fluid dynamics*, Computer Meth. Appl. Mech. Engng., 40 (1983), pp. 27-109.
[7] M. ISRAELI AND E. WASSERSTROM, ILAM National Conference, Weizmann Institute, Rehovot, Israel, 1981 (Abstract). (In Hebrew.)
[8] K. MILLER, *Numerical analogs to the Schwarz alternating procedure*, Numer. Math. 7 (1965), pp. 91-103.
[9] G. RODRIGUE AND J. SIMON, *A generalization of the numerical Schwarz algorithm*, Lawrence Livermore Lab. Report UCRL-90030, Livermore, CA, 1983.
[10] H. A. SCHWARZ, *Gesammelte Mathematische abhandlungen*, Springer, Berlin, 2 (1890), pp. 133-134.
[11] P. N. SCWARZTRAUBER AND R. SWEET, Report NCAR-TN/IA-1-9, NCAR, Boulder, CO, 1975.
[12] O. B. WIDLUND, *Iterative methods for elliptic problems on regions partitioned into substructures and the biharmonic Dirichlet problem*, in Computing Methods in Applied Sciences and Engineering, VI, R. Glowinski and J. L. Lions, eds., North-Holland, Amsterdam, 1984, pp. 33-45.
[13] Y. A. KUZNETSOV, *Matrix computational processes in subspaces*, in Computing Methods in Applied Sciences and Engineering, VI, R. Glowinski, J. L. Lions, eds., North-Holland, Amsterdam, 1984.

SIAM J. SCI. STAT. COMPUT.
Vol. 7, No. 3, July 1986

020

ON THE SCOPE OF THE METHOD OF MODIFIED EQUATIONS*

D. F. GRIFFITHS† AND J. M. SANZ-SERNA‡

Abstract. A rigorous analysis is presented for the method of modified equations whereby its range of applicability and its shortcomings are delineated. Numerous examples from different areas are presented and the theoretical findings are confirmed throughout by computational experiments.

Key words. modified equations, difference approximations, local truncation errors, stability

AMS (MOS) subject classifications. 65L05, 65L10, 65M05, 65M10

1. Introduction. Modified equations have been a commonly used tool in the study of difference schemes. Because of the lack of any theoretical foundation, this use has been accompanied by constant difficulties and results derived from modified equations have sometimes been regarded with apprehension. As a result a situation has arisen where authors either disregard entirely the technique or have an unjustified faith in its scope. The aim of the present paper is to investigate carefully the foundation and applicability of the method in the hope of clarifying the situation.

To our best knowledge the method of modified equations was first used by Garabedian [4] in the analysis of SOR iterations. Few papers have been devoted to studying the method (Hirt [11], Warming and Hyett [34], Wilders [35], also Morton [16]). On the other hand the technique has been extensively employed in the literature, see e.g. [1], [6], [7], [8], [10], [14], [19], [20], [26], [33] and [36]. By and large, applications have concentrated on the investigation of dispersive and dissipative properties of partial difference schemes. A nonstandard example is given by Duncan and Griffiths [3]. One of the referees has rightly pointed out the analogy between the idea of modified equation and the backward error analysis of Wilkinson.

A summary of the paper is as follows. The main ideas are introduced in § 2, in the context of a concrete example. This is followed in § 3 by the discussion of a wide range of applications to both ordinary and partial differential equations. The theoretical analyses are backed throughout by numerical illustrations. We place the method in a wider context in § 4, by making comparisons with other forms of analysis. Our findings are summarized in § 5.

In keeping with the aim of the paper, the examples included, mostly simple, have been chosen to provide insight into the various aspects of the method; the presentation of new real-life applications is completely outside the scope of the article.

2. Modified equations. This section introduces, in a rigorous way, the concept of modified equation. For simplicity, the ideas are presented in the case of a model problem which exhibits all the important features of the more general situation. In fact, it is not difficult to rewrite the material below in the language of any of the general discretization theories (e.g. [31], [29], [33], [23]) and in particular, [37 § 2.4]).

We consider the scalar initial value problem

$$u(0) = \eta, \tag{2.1a}$$

$$\frac{du}{dt} = f(u), \qquad 0 \leqq t \leqq T, \tag{2.1b}$$

* Received by the editors September 18, 1984, and in revised form, February 20, 1985.

† Department of Mathematical Sciences, The University, Dundee DD1 4HN, Scotland.

‡ Departamento de Ecuaciones Funcionales, Facultad de Ciencias, Universidad de Valladolid, Valladolid, Spain.

where $f(u)$ is smooth and Lipschitz continuous in $-\infty < u < \infty$, with Lipschitz constant L. These hypotheses ensure the existence and the uniqueness of a smooth solution. The problem (2.1) is discretized by means of Euler's rule

$$U_0 = \eta + \delta, \tag{2.2a}$$

$$(U_{n+1} - U_n)/h = f(U_n), \qquad n = 0, 1, \cdots, N-1. \tag{2.2b}$$

Here N is a positive integer, $h = T/N$ and δ caters for a possible error in the starting value. For simplicity, the effects of round-off errors are not considered in this paper.

Some of the basic, elementary steps of the analysis of (2.2) ([12], [9], [5]) will now be presented for later reference. A crucial part of the analysis is the estimation of the size of the global errors

$$e_n = Y_n - U_n, \tag{2.3}$$

where $Y_n = u(t_n)$ is the value of the theoretical solution at the grid-point $t_n = nh$. In more concrete terms we are interested in the quantity

$$e = \max \{|e_n|: n = 0, 1, \cdots, N\}. \tag{2.4}$$

Note that U_n, Y_n, e_n, e, δ depend on the parameter h but this dependence does not appear in the notation. The standard approach to the study of e is the following indirect one (and this includes both the derivation of bounds for e for a given, fixed h and the investigation of the behaviour of e as h tends to zero).

First the auxiliary *local truncation errors*

$$l_0 = Y_0 - (\eta + \delta), \tag{2.5a}$$

$$l_{n+1} = (Y_{n+1} - Y_n)/h - f(Y_n), \qquad n = 0, 1, \cdots, N-1 \tag{2.5b}$$

are introduced. A simple Taylor expansion taking into account that $Y_n = u(t_n)$ reveals that, for $n > 0$, l_n can be bounded by $\frac{1}{2}hB_2$, where B_2 is a bound for $|u''(t)|$, $0 \leqq t \leqq T$. Thus

$$l = \max \{|l_n|: n = 0, 1, \cdots, N\} \tag{2.6}$$

is $O(h + \delta)$ as $h \to 0$.

Then, the *stability* of the discretization is established, i.e. it is shown that

$$e \leqq Cl, \tag{2.7}$$

where C is a postive constant which depends on T and L but not on h. This bound is derived by subtracting (2.5) from (2.2) and applying induction w.r.t. n. No property of Y_n is required in the derivation, i.e. the fact that $Y_n = u(t_n)$ is not used at this stage. From (2.7) e is also $O(h + \delta)$ and one says that (2.2) possesses first order rate of convergence.

Remark. Some authors [13] prefer to write (2.2b) in the undivided form

$$U_{n+1} - U_n = hf(U_n).$$

Accordingly they define the local truncation error for $n > 0$, to be

$$Y_{n+1} - Y_n - hf(Y_n)$$

rather than (2.5b). With this definition the local errors are $O(h^2 + \delta)$ while the global errors, whose definition remains unchanged, are $O(h + \delta)$. In this paper, finite difference schemes are always written in divided form, i.e. in the form resulting from the replacement of derivatives by divided differences.

The introduction of modified equations aims at describing the behaviour of the numerical solution U_n. This will now be illustrated in the context of (2.1), (2.2). We consider the *modified* problem

$$z(0) = \eta + \delta, \tag{2.8a}$$

$$(1+\tfrac{1}{2}hf'(z))z' = f(z), \qquad 0 \leqq t \leqq T. \tag{2.8b}$$

(The derivation of modified problems is considered in the next section. No motivation for (2.8) will be provided at this stage.) The standard theory of continuous dependence on the parameters shows that, at least for h small, (2.8) has a unique solution $z(t)$. (Notice again that $z(t)$ depends on h.) We claim that $z(t_n)$ is a better approximation to the numerical solution U_n than $u(t_n)$. Our task is to bound the quantity e given by (2.4)-(2.3), where now $Y_n = z(t_n)$. In order to do so, we resort to the indirect approach above. We still define l_n, l by (2.5)-(2.6) with $Y_n = z(t_n)$ and observe that (2.7) is still valid, since, as noted before, the derivation of the stability bound does not use any information on Y_n. However, now

$$l_0 = Y_0 - U_0 = z(0) - U_0 = 0$$

while, for $n = 0, 1, \cdots, N-1$,

$$\begin{aligned} l_{n+1} &= (Y_{n+1} - Y_n)/h - f(Y_n) = [z(t_{n+1}) - z(t_n)]/h - f(z_n) \\ &= z'(t_n) + (h/2)z''(t_n) + (h^2/6)z'''(\theta_n) - f(z_n), \end{aligned}$$

where $t_n < \theta_n < t_{n+1}$. On using (2.8b)

$$\begin{aligned} l_{n+1} &= z'(t_n) + (h/2)z''(t_n) + (h^2/6)z'''(\theta_n) - z'(t_n) - (h/2)f'(z(t_n))z'(t_n) \\ &= (h/2)[z''(t_n) - f'(z(t_n))z'(t_n)] + (h^2/6)z'''(\theta_n) \end{aligned}$$

which, on using the equation obtained by differentiation of (2.8b), leads to

$$l_{n+1} = (h/2)^2[f'(z(t_n))z''(t_n) + f''(z(t_n))(z'(t_n))^2] + (h^2/6)z'''(\theta_n). \tag{2.9}$$

Now z, z', z'', z''' can be bounded independently of h because of the continuous dependence of the solutions of (2.8) on the parameter h. We conclude that now $e = O(h^2)$ and say that the modified problem (2.8) describes the behaviour of the solution of (2.2) with second order of correctness. This will be now illustrated by means of an example.

The problem (2.8) is easily integrated to yield

$$\int_{\eta+\delta}^{z(t)} \frac{dv}{f(v)} + \frac{h}{2} \ln \frac{|f(z(t))|}{|f(\eta+\delta)|} = t. \tag{2.10}$$

In what follows we set $f(u) = u^2$. This does not strictly satisfy the hypotheses above in that $f(u)$ is Lipschitz continuous for $-M < u < M$, M finite but not for $-\infty < u < \infty$, however this poses no difficulty (see e.g. [25, p. 24]). We further set $T = .99$, $\eta = 1$, $\delta = 0$ with theoretical solution $u(t) = 1/(1-t)$. From (2.10), the modified solution $z(t)$ is given by

$$1 - \frac{1}{z} + h \ln z = t.$$

Figure 1 depicts $z(t)$, $u(t)$ and the Euler points U_n when $h = T/4$, $T/16$. It is clear that the values computed by the difference scheme are much closer to the values $z(t_n)$ than to the values $u(t_n)$. Moreover the agreement between $z(t_n)$ and U_n is very good, even for the coarser grid.

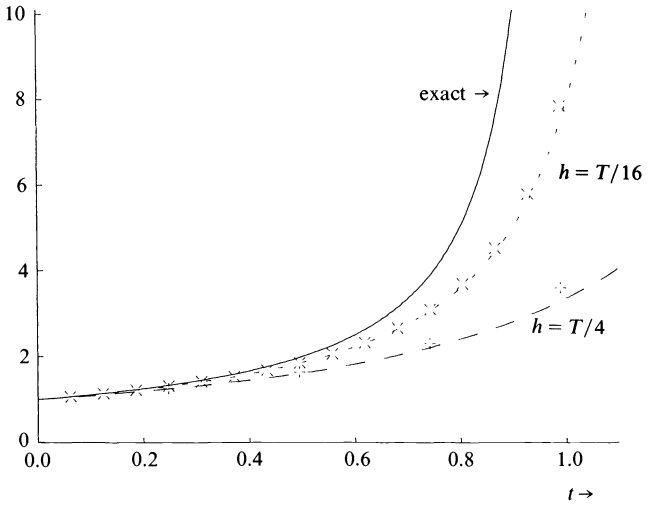


FIG. 1. *Exact solution of* $u' = u^2$, $0 \leqq t \leqq T$ $(=0.99)$ *(full line), together with solution of modified equation (broken lines) and numerical solution by Euler's method* $(+$ *and* $\times)$ *for* $h = T/4$ *and* $h = T/16$.

It should be noted that the modified equation continues to describe the behaviour of the Euler solution even for $nh \geqq 1$, when the theoretical solution $u(t)$ ceases to exist (cf. [24]). This is illustrated in Fig. 2. One can actually derive bounds for $U_n - z(t_n)$, $nh \geqq 1$, but this point will not be pursued further. The following points summarize the main ideas and are useful in preventing the pitfalls which may arise from an indiscriminate application of modified problems.

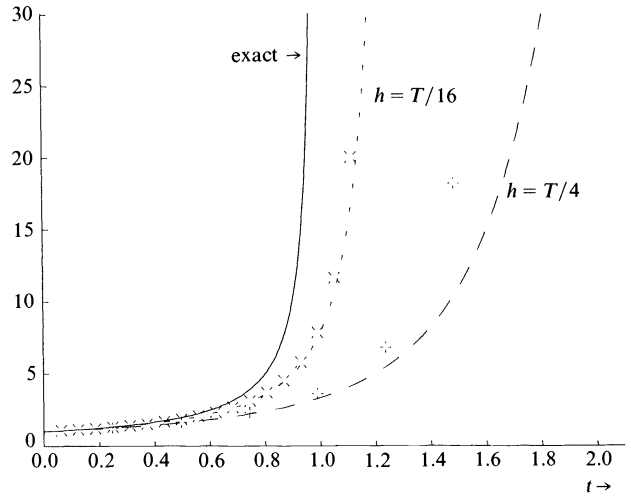


FIG. 2. *Solutions as in Fig.* 1 *but for an extended time interval.*

(i) A modified problem correct of order p is a problem depending on the parameter h with the property that its solution z has a local discretization error $O(h^p)$; i.e. it satisfies, except for $O(h^p)$ terms, the discrete equations defining the numerical method. In order to prove that the local discretization error l is $O(h^p)$, it is not enough to show that $l \leqq h^p B_p$, where B_p depends on the derivatives of z. In fact $z = z(h)$ and one must

also check that B_p remains bounded as $h \to 0$. This point was illustrated in the argument which follows (2.9).

It is perhaps worth pointing out that for a numerical method with qth order of convergence, the original problem being solved provides already a modified problem correct of order q.

(ii) It should be observed that even though it is customary in the literature to talk about modified *equations*, it is essential to consider modified *problems*, i.e. the modified equation should be supplemented by the necessary initial/boundary conditions (such as (2.8a) and care should be exercised in checking that the modified solution satisfies, except for $O(h^p)$ terms, the initial/boundary discrete equations (such as (2.2a)) which supplement the main scheme (such as (2.2b)).

(iii) The stability of the numerical method is an essential ingredient in guaranteeing the success of the method of modified problems. Without stability the bounds for local errors cannot be transferred to the global error $z - U$. The concept of stability used here refers to the $h \to 0$, nh fixed case (0-stability in ODEs [13], Lax stability in PDEs [19], [17], [18]) and not to the $n \to \infty$, fixed h case (weak stability in [13], contractivity [2]).

The importance of the points (i)-(iii) above will be borne out by the examples in the next section.

The idea of comparing the numerical solution U with a function close to but different from the theoretical solution u goes back to Strang [30]. See also [27], [22] and [37, Chap. 1].

3. The construction of modified problems: examples and counterexamples. In this section examples of modified problems are constructed, which illustrate the range of applicability of the technique.

(A) In our first example we return to (2.1)-(2.2). In order to construct a modified problem, correct of order two, the values of a smooth function $w(t)$ are substituted in (2.5):

$$l_0 = w(0) - (\eta + \delta),$$

$$l_{n+1} = (w(t_{n+1}) - w(t_n))/h - f(w(t_n)), \qquad n = 0, 1, \cdots, N-1.$$

The possible dependence of $w(t)$ on h is not reflected in our notation. On Taylor expanding, we obtain

$$l_{n+1} = w'(t_n) + \frac{h}{2} w''(t_n) + \frac{h^2}{6} w'''(t_n) + \cdots - f(w(t_n))$$

and the requirement that $l = O(h^2)$ implies that $w(t)$ should satisfy

$$w(0) = \eta + \delta + O(h^2),$$

$$w'(t) + \frac{h}{2} w''(t) = f(w(t)) + O(h^2).$$

In particular, the equations

$$w(0) = \eta + \delta, \tag{3.1a}$$

$$w' + \frac{h}{2} w'' = f(w) \tag{3.1b}$$

appear to be good candidates for the role of modified problem with second order of correctness. However two difficulties have to be addressed. First the missing initial

value $w'(0)$ needs to be specified. Secondly, as $h \to 0$ the equation (3.1a) is singularly perturbed and there is a danger of w'' increasing without bound. Such a growth would destroy the $O(h^2)$ bound on l, as noted in § 2(i). The success of the modified problem approach depends on extracting a regularly perturbed problem from (3.1). A means of achieving this is by a suitable choice of $w'(0)$ to accompany (3.1). The difficulties inherent in this approach do not manifest themselves in this example and the study of this technique is deferred until the next example.

A second means of regularizing (3.1) is now presented.

Differentiation of (3.1b) leads to

$$w'' + \frac{h}{2} w''' = f'(w) w'.$$

Upon eliminating w'' between this equation and (3.1b), we obtain

$$\left(1 + \frac{h}{2} f'(w)\right) w' - \frac{h^2}{4} w''' = f(w).$$

The solutions of this equation we are interested in, namely those whose derivatives remain bounded as $h \to 0$, differ by $O(h^2)$ from those of

$$\left(1 + \frac{h}{2} f'(z)\right) z' = f(z), \tag{3.2}$$

an equation which is not singularly perturbed. It was rigorously shown in § 2 that Euler's method provides a second order approximation to (3.2).

In practice, and for a more general problem, the steps leading up to a modified problem need not be performed rigorously. One would begin by replacing the grid values in the discrete equations by those of a smooth function w. Then, on performing a Taylor expansion and discarding powers of h higher than the pth, one would arrive at an equation involving high derivatives of w. Finally, and as far as possible, higher derivatives would be eliminated by combining this equation with those resulting from its differentiation (while systematically deleting terms which involve powers of h above the pth).

Once a candidate for a modified problem has been obtained by mere formal manipulation, the local error of its solution z should be rigorously shown to be small in order to conclude that z models the behaviour of the numerical solution provided by a stable scheme (cf. (i)-(iii) § 2).

An instance is provided by the equation

$$z' = \left(1 - \frac{h}{2} f'(z)\right) f(z), \tag{3.3}$$

which results from formal inversion up to $O(h^2)$ of the factor $1 + (h/2) f'(z)$ in (3.2). One easily shows that solutions of (3.3) with $z(0) = \eta + \delta$ possess a local error $l \leqq Ch^2$ (C independent of h), thereby providing a new modified problem for (2.2). This demonstrates that modified problems correct of order p are, by no means, unique.

(B) We retain the initial value problem (2.1), but this time discretize it by means of the backward Euler rule

$$U_0 = \eta + \delta,$$
$$(U_{n+1} - U_n)/h = f(U_{n+1}), \qquad n = 0, 1, \cdots, N-1.$$

On proceeding as at the beginning of the previous example we arrive at the following analogue of (3.1)

$$w(0) = \eta + \delta, \tag{3.4a}$$

$$w' - \frac{h}{2} w'' = f(w). \tag{3.4b}$$

We now discuss the regularization of (3.2) by means of a suitable choice of $w'(0)$. To avoid any unwelcome detail, we only consider the case $\eta = 1$, $\delta = 0$, $f(u) = \lambda u$. The family of solutions of (3.4a)-(3.4b) is given by

$$w(t) = (1+\alpha)\, e^{r_+ t} - \alpha\, e^{r_- t},$$
$$r_\pm = (-1/h)[\pm\sqrt{1-2\lambda h} - 1],$$

so that $r_\pm = \lambda + O(h)$, $r_- = 2/h + O(1)$ and the derivatives of w will increase as $h \to 0$ unless the missing starting value $w'(0)$ is chosen to guarantee that $\alpha = 0$, i.e. $w'(0) = r_+$. When $w'(0) \neq r_+$, solutions of (3.4) do not describe up to $O(h^2)$ the behaviour of the numerical solution, even though (3.4) was obtained by insisting that the expansion of the local error should only contain terms involving factors h^s, $s \geqq 2$ (cf. (i) of § 2).

This is illustrated numerically in Table 1, where $\lambda = 1$, $t = \frac{1}{2}$ and $w'(0) = \lambda$ (a reasonable choice, since this coincides with $u'(0)$) and $w'(0) = r_+$. The theoretical solution has $u(\frac{1}{2}) \simeq 1.649$.

TABLE 1

h	Numerical	Modified $w'(0) = \lambda$	Modified $w'(0) = r_+$
$\frac{1}{4}$	1.778	.93	1.796
$\frac{1}{8}$	1.706	−7.32	1.709
$\frac{1}{16}$	1.676	−5,907.49	1.676

Had Euler's rule been used, the roots r_+, r_- would have satisfied $r_+ = \lambda + O(h)$, $r_- = -2/h + O(1)$ and then the study of the size of the derivatives of $\exp(r_- t)$ would have been rather delicate due to a boundary layer at $t = 0$.

(C) This and the following example show the importance of considering modified *problems* rather than modified *equations*, i.e. proper account must be taken of all side conditions (§ 2(ii)).

We again consider the problem (2.1), but this time discretized by the leap-frog scheme

$$U_0 = \eta, \tag{3.5a}$$

$$(U_1 - U_0)/h = f(U_0), \tag{3.5b}$$

$$(U_{n+2} - U_n)/2h = f(U_{n+1}), \qquad n = 0, 1, \cdots, N-2, \tag{3.5c}$$

where the additional starting value U_1 is obtained by Euler's method. The scheme (3.5) possesses second order of convergence and therefore the original problem (2.1) provides a modified problem with second order of correctness. We now seek a modified problem of third order of correctness. On proceeding as in the derivation of (3.2), we

obtain the equation

$$\left[1+\frac{h^2}{6}(f''(z)f(z)+f'(z)^2)\right]z'=f(z) \tag{3.6}$$

whose solutions satisfy (3.5c) except for an $O(h^3)$ local discretization error. In this sense, (3.6) is a modified *equation* correct of *third* order for the leap-frog scheme. However solutions of (3.6) with $z(0)=\eta$ only satisfy (3.5b) with *second* order of correctness. Therefore a modified problem based on (3.6) cannot attain third order of correctness. A numerical example with $f(u)=u$, $u(0)=z(0)=1$, $t=1$ is presented in Table 2, which shows that the approximation provided by z has only second order of accuracy. In fact, no smooth function w of t and the parameter h can satisfy $w(t_n)-U_n=O(h^3)$, since the theory of asymptotic expansions of global errors [9] shows that $u(t_n)-U_n=h^3[\phi(t_n)+(-1)^n\psi(t_n)]+O(h^4)$, with ϕ and ψ smooth functions, which leads to a disparity between even and odd grid values of U_n. (This disparity is evident in the table.) A means of describing the behaviour of U_n may be found in [21] (cf. [28]).

TABLE 2

h	$(u-U)/h^2$	$(z-U)/h^2$
$\frac{1}{3}$	1.13	.69
$\frac{1}{4}$	0.99	.55
$\frac{1}{7}$	1.21	.76
$\frac{1}{8}$	1.03	.58
$\frac{1}{15}$	1.22	.77
$\frac{1}{16}$	1.04	.58

(D) The two-point boundary value problem

$$u(0)=0, \tag{3.7a}$$

$$-u''+u=0, \qquad 0\leqq t\leqq 1, \tag{3.7b}$$

$$u'(1)=1, \tag{3.7c}$$

is discretized by

$$U_0=0, \tag{3.8a}$$

$$-(U_{n-1}-2U_n+U_{n+1})/h^2+U_n=0, \qquad n=1,2,\cdots,N-1, \tag{3.8b}$$

$$(U_N-U_{N-1})/h=1, \tag{3.8c}$$

where $h=1/N$, N a positive integer. We observe that (3.8b) approximates (3.7b) with second order accuracy, while (3.8c) is only a first order accurate replacement of (3.7c). Consequently we obtain the following modified problem, which has second order of correctness

$$z(0)=0, \tag{3.9a}$$

$$-z''+z=0, \qquad 0\leqq t\leqq 1, \tag{3.9b}$$

$$z'(1)-(h/2)z(1)=1, \tag{3.9c}$$

where the last equation has been derived by Taylor expanding (3.8c) and using (3.9b) to eliminate z''. Table 3, which shows values at $t=1$, provides illustration of the fact that (3.8) is first order accurate, while (3.9) coincides with (3.8) up to second order.

TABLE 3

h	(Exact-numerical)/h	(Modified-numerical)/h
$\frac{1}{4}$	−.287	+.034
$\frac{1}{8}$	−.289	+.015
$\frac{1}{16}$	−.290	+.007

(E) Our final example is given by the following periodic initial value problem for the heat equation:

$$u(x,0)=u_0(x), \qquad -\infty<x<\infty, \tag{3.10a}$$

$$u(x,t)=u(x+1,t), \qquad -\infty<x<\infty, \quad t>0, \tag{3.10b}$$

$$u_t=u_{xx}, \qquad -\infty<x<\infty, \qquad t>0, \tag{3.10c}$$

together with the discretization

$$U_j^0=u_0(jh), \qquad j=0,\pm1,\pm2,\cdots, \tag{3.11a}$$

$$U_j^n=U_{j+J}^n \quad n=1,2,\cdots, \quad j=0,\pm1,\pm2,\cdots, \tag{3.11b}$$

$$(U_j^{n+1}-U_j^n)/k=(U_{j-1}^n-2U_j^n+U_{j+1}^n)/h^2, \quad n=0,1,\cdots, \quad j=0,\pm1,\pm2,\cdots. \tag{3.11c}$$

Here u_0 is 1-periodic, $h=1/J$, J a positive integer and $k=rh^2$, with r a positive parameter. We now present in detail the construction of a modified problem of second order of correctness (in k), so as to show the additional novelties involved in dealing with PDEs. A smooth function $w(x,t)$ is substituted in (3.11c) to yield

$$l_j^{n+1}=(w_j^{n+1}-w_j^n)/k-(w_{j-1}^n-2w_j^n+w_{j+1}^n)/h^2,$$

where $w_j^n=w(jh,nk)$. On Taylor expanding, we obtain

$$l_j^{n+1}=(w_t-w_{xx})+\frac{k}{2}\left(w_{tt}-\frac{1}{6r}w_{xxxx}\right)+\cdots \tag{3.12}$$

which leads to

$$w_t=w_{xx}-\frac{k}{2}\left(w_{tt}-\frac{1}{6r}w_{xxxx}\right) \tag{3.13}$$

as a candidate for modified equation. Again (3.13) contains a small parameter in front of the highest derivatives and, because of its high order, requires more side conditions than can be derived from (3.11a)-(3.11b). Differentiation of (3.13), first with respect to t and then with respect to x twice, yields

$$w_{tt}=w_{xxt}-\frac{k}{2}\left(w_{ttt}-\frac{1}{6r}w_{xxxxt}\right),$$

$$w_{xxt}=w_{xxxx}-\frac{k}{2}\left(w_{xxtt}-\frac{1}{6r}D_x^6 w\right).$$

These equations can now be used to eliminate w_{tt} from (3.13) and, on discarding terms involving k^2, we arrive at the equation

$$z_t=\left(1-\frac{k}{2}\left(1-\frac{1}{6r}\right)D_x^2\right)z_{xx}. \tag{3.14}$$

We notice in passing that the form

$$\left(1+\frac{k}{2}\left(1-\frac{1}{6r}\right)D_x^2\right)z_t = z_{xx} \tag{3.15}$$

resulting from formally inverting the operator in brackets on the right of (3.14), may also be considered. This alternative form seems advantageous in the case of initial boundary value problems, since it does not increase the number of required boundary conditions. See below.

We now discuss whether (3.14), together with

$$z(x, 0) = u_0(x), \qquad -\infty < x < \infty, \tag{3.16a}$$

$$z(x+1, t) = z(x, t), \qquad -\infty < x < \infty, \quad t > 0, \tag{3.16b}$$

provides a modified problem for the study of (3.11). The Fourier transform of (3.14) is given by

$$(d/dt)\hat{z}(m, t) = -\left(1+\frac{k}{2}\left(1-\frac{1}{6r}\right)4m^2\pi^2\right)4m^2\pi^2\hat{z}(m, t)$$

where m is the wave number ($m = 0, \pm 1, \pm 2, \cdots$). This leads to

$$\hat{z}(m, t) = \hat{z}(m, 0)\exp[\sigma(m)t] \tag{3.17}$$

where $\sigma(m)$ is the symbol of (3.14)

$$\sigma(m) = -\left(1+\frac{k}{2}\left(1-\frac{1}{6r}\right)4m^2\pi^2\right)4m^2\pi^2.$$

Three ranges of the parameter r should be studied separately.

(i) $\frac{1}{6} \leqq r \leqq \frac{1}{2}$. When r has been fixed within this range the exponential term in (3.17) is bounded for all $t \geqq 0$ uniformly in m and k. Therefore the solutions of (3.14)-(3.16) are bounded together with their derivatives uniformly in k. This fact combined with the stability of the scheme (3.11) allows us to conclude that we are dealing with a problem of second order of correctness. (Note that for $r = \frac{1}{6}$ (3.14) reduces to the original equation (3.10c), in agreement with the fact that, for this value of r, (3.11) is convergent for order $O(k^2)$ [15].) Table 4 provides a numerical illustration of the approximation at

$$x = \tfrac{1}{2}, \quad t = \tfrac{1}{4}, \quad u_0(x) = \sum_{l=1}^{\infty} \frac{(-1)^l}{l^6}\cos 2\pi l x, \quad u(\tfrac{1}{2}, \tfrac{1}{4}) = 5.172 \times 10^{-5}$$

when $r = \frac{1}{4}$.

TABLE 4

h	Numerical $\times 10^5$	Modified $\times 10^5$
$\frac{1}{4}$	26.327	1.875
$\frac{1}{8}$	4.351	4.013
$\frac{1}{16}$	4.851	4.854
$\frac{1}{32}$	5.091	5.091

(ii) $\frac{1}{2} < r$. In this range the exponential term in (3.17) is still bounded. However the scheme is now *unstable* and bounds for the local discretization error do not lead

to bounds for global errors. As a result (3.14)-(3.16) do not model the behaviour of (3.11). This is borne out by Table 5, which is analogous to Table 4 except for the fact that now $r=\frac{2}{3}$.

TABLE 5

h	Numerical $\times 10^5$	Modified $\times 10^5$
$\frac{1}{4}$	-2.05×10^4	0.012
$\frac{1}{8}$	-3.13×10^7	1.129
$\frac{1}{16}$	-4.57×10^{20}	3.535
$\frac{1}{32}$	1.55×10^{39}	4.703

(iii) $0<r<\frac{1}{6}$. In this range the scheme is stable, but the exponential term in (3.17) cannot be bounded uniformly in m, k. Thus (3.14)-(3.16) is not well-posed uniformly in k. (Even for a fixed $k>0$, (3.14)-(3.16) cannot be solved for arbitrary initial data, due to the unboundedness of the exponential term as m varies, a situation similar to that for the backward heat equation. In particular (3.14)-(3.16) does not possess a solution for the initial datum employed in Tables 4-5.)

When attention is restricted to initial data u_0 containing only a prescribed finite number M of harmonics, (3.14), (3.16) may still be of some value, since the exponential in (3.17) is bounded for $m\leqq M$ and k sufficiently small $k\leqq k_0(M)$. This remark has often been expressed in the literature by saying that modified equations are valid only when the product mk is small [34], [16], [33], [20].

Table 6 refers to the initial condition

$$u_0=\sum_{l=1}^{M}\frac{(-1)^l}{l^6}\cos 2\pi lx,$$

$x=\frac{1}{2}$, $t=\frac{1}{4}$, $r=\frac{1}{8}$, $M=5$, $u(\frac{1}{2},\frac{1}{4})=5.172\times 10^{-5}$. As M is increased, the value of h must be decreased accordingly in order to attain a prescribed level of accuracy.

TABLE 6

h	Numerical $\times 10^5$	Modified $\times 10^5$
$\frac{1}{4}$	34.474	2.277×10^{31}
$\frac{1}{8}$	5.927	5.872
$\frac{1}{16}$	5.342	5.339
$\frac{1}{32}$	5.214	5.213

To conclude this example we point out that the alternative modified problem (3.15)-(3.16) is uniformly well-posed, as $k\to 0$, if and only if r lies in the range $0<r\leqq\frac{1}{6}$. Therefore (3.14), (3.15) complement each other and allow a study of the scheme in the entire stable range $0<r\leqq\frac{1}{2}$.

4. Related techniques. The modified equation approach is closely related to other commonly employed means of analysis. We first consider the use of variational equations to study the behaviour of the global error $u-U$. For the sake of simplicity, attention is restricted to the model situation (2.1)-(2.2) with $\delta=0$. It is well known [9] that $U_n=u(t_n)+hv(t_n)+O(h^2)$, where the function $v(t)$ does not depend on h

and satisfies

$$v(0)=0, \tag{4.1a}$$

$$v'=f'(u(t))v-\tfrac{1}{2}u''(t), \qquad 0\leqq t\leqq T. \tag{4.1b}$$

Thus $y(t)=u(t)+hv(t)$ provides a model for the description of the Euler solution accurate to $O(h^2)$. However the determination of $y(t)$ requires successively the solution of the original problem (2.1) and that of the variational problem (4.1). The modified problem approach, on the other hand, involves only the solution of the single problem (2.8). This latter approach is therefore more convenient in practice, where often only qualitative information on the behaviour of U is of interest. Nevertheless the two approaches are closely related, as borne out by the fact that (2.8) can be rigorously derived from (4.1) as follows. On using (2.1b), we can rewrite (4.1b) as

$$v'=f'(u)v-\tfrac{1}{2}f'(u)u'.$$

Hence, since $y=u+hv$,

$$\begin{aligned} y'=u'+hv'&=f(u)+hf'(u)v-\frac{h}{2}f'(u)u' \\ &=f(y)-\frac{h}{2}f'(y)y'+O(h^2), \end{aligned} \tag{4.2}$$

where in the final step we have made use of the smoothness of u and v. Deletion of the $O(h^2)$ remainder in (4.2) can only lead to an $O(h^2)$ change in the solutions and yields (2.8).

This close relationship between the variational and modified equation approaches merely reflects the fact that they are based on the same information, namely the leading terms in the expansion of the local error. This remark applies equally to any strongly stable [29] linear multistep method. For stable linear multistep methods having roots $r\neq 1$, $|r|=1$ the situation is more delicate [9] due to the effect of choice of starting values. (See example C) above.)

The observation that modified problems make use only of the leading terms of the expansion of the local error applies generally, and is primarily responsible for restricting the scope of the method. A further illustration is given in the context of the heat equation example in the previous section. The scheme (3.11) was used there only insofar as to derive (3.12). In turn the modified equations (3.14), (3.15) were based solely on the terms displayed in (3.12); consequently they would serve as modified equations for any scheme that gave rise to the same terms. On the other hand the amplification factor [15]

$$\xi(m)=1-4r\sin^2 m\pi h,$$

where the wave number m is an integer, provides a complete characterization of the scheme and may therefore be used to deduce all its properties. In particular the first order of local accuracy ($r\neq\frac{1}{6}$) is a consequence of the expansion

$$\frac{\xi-\exp\left[-(2m\pi)^2k\right]}{k}=-\frac{k}{2}(2m\pi)^4\left(1-\frac{1}{6r}\right)+O(k^2m^6), \tag{4.3}$$

This expression is simply the Fourier transform of (3.12) when w is a solution of (3.10c). The $O(k)$ term in the right of (4.3) is the Fourier transform of the leading term of the local truncation error, which is the only information required to construct

the modified problems. This is reinforced by noting that

$$-\frac{k^2}{2}(2m\pi)^4\left(1-\frac{1}{6r}\right) = \exp\left[\frac{k^2}{2}(2m\pi)^4\left(1-\frac{1}{6r}\right)\right]-1+O(k^4m^8)$$

$$= \exp\left[-(2m\pi)^2k\right]\left\{\exp\left[\frac{k^2}{2}(2m\pi)^4\left(1-\frac{1}{6r}\right)\right]-1\right\}$$

$$+O(k^3m^6),$$

which, together with (4.3), leads to

$$\frac{\xi-\exp(\sigma k)}{k}=O(k^2m^6),$$

where $\sigma=\sigma(m)$ is the symbol of the modified equation (3.14). In other words the symbol $\sigma(m)$ and consequently the modified equation itself, can be derived from the terms displayed in (4.3) without having to resort to the original difference equations.

The study of the stability of the scheme (both for $k\to 0$, t fixed and k fixed, $t\to\infty$) requires complete knowledge of $\xi(m)$, $|hm|\leqq\pi$, information which cannot be deduced from the leading terms of the expansion of $\xi(m)$ around $mh=0$. Consequently, properties such as stability cannot be ascertained from a study of modified problems (cf. Table 5). Therefore the cases reported in the literature where analysis of a modified problem has resulted in the correct stability limits must be regarded as coincidence. These attempts have, by and large, been restricted to cases where the stability had previously been analysed by different means and the stability limits were thus known beforehand.

It may be useful to point out that although the derivation of modified equations only takes into account the behaviour of the numerical scheme for mh small, *well-posed* modified problems describe accurately the numerical solution provided by (Lax) *stable* schemes even if the solution contains all wave numbers (cf. Table 4). The reason for this is that in any initial datum in (say) L^2 the high frequencies are represented with amplitudes which tend to zero as the wave number increases. "It does no harm for these higher harmonics to be falsified" both by the scheme and by the modified equation "provided only that they do not become amplified to such an extent as to be no longer negligible" (see [19, p. 11]. The quoted sentences have been taken from this reference.)

5. Conclusions. The following conclusions have emerged from our study of the method of modified problems.

(i) The construction of a modified problem correct of order p may be undertaken in a purely formal manner. Having arrived at a suitable candidate it is necessary to verify that its solution satisfies the discrete equations except for an $O(h^p)$ remainder. In doing so it is imperative to ensure that any derivatives appearing in the remainder are bounded as $h\to 0$ (cf. Table 1).

(ii) Side conditions in both the original problem and its discretization must be incorporated into the analysis (cf. examples C) and D)).

(iii) Stability as $h\to 0$ of the discrete method being analyzed is an essential prerequisite for the success of the analysis. Without stability, estimates of the local truncation error do not imply estimates of the global error (cf. Table 5).

(iv) Since only a limited amount of information on the scheme is used in constructing a modified problem, such problems cannot provide a full description of the scheme. In particular stability properties, both for h fixed, $t\to\infty$ and $h\to 0$, t fixed, cannot be deduced from a modified problem.

(v) It has often been asserted in the literature that modified partial differential equations provide a valid description of the numerical solution only when the product (wave number) $\times h$ is small. However our analysis has revealed that this is not necessarily the case and that solutions to initial data containing all harmonics can be described, provided that the candidate modified problem satisfies (i)-(iii) above (cf. Table 4 and last paragraph of § 4).

Acknowledgment. The authors would like to express their gratitude to the British Council in Spain for their financial assistance.

REFERENCES

[1] R. C. Y. CHIN AND G. W. HEDSTROM, *A dispersion analysis of difference schemes: Tables of generalized Airy functions*, Math. Comp., 32 (1978), pp. 1163-1170.

[2] K. DEKKER AND J. G. VERWER, *Stability of Runge-Kutta Methods for Stiff Nonlinear Differential Equations*, North-Holland, Amsterdam, 1984.

[3] D. B. DUNCAN AND D. F. GRIFFITHS, *The study of a Petrov-Galerkin method for first order hyperbolic equations*, Comp. Meth. Appl. Mech. Eng., 45 (1984), pp. 147-166.

[4] P. R. GARABEDIAN, *Estimation of the relaxation factor for small mesh sizes*, Mathematical Tables and other Aids for Computation, 10 (1956), pp. 183-185.

[5] C. W. GEAR, *Numerical Initial Value Problems in Ordinary Differential Equations*, Prentice-Hall, Englewood Cliffs, NJ, 1971.

[6] D. F. GRIFFITHS, *Explicit methods for multidimensional advection-diffusion problems*, Proc. VI Congreso de Ecuaciones Diferenciales y Aplicaciones, Universidad de Zaragoza, 1983, pp. 64-80.

[7] A. HARTEN, J. M. HYMAN AND P. D. LAX, Appendix by B. KREYFITZ, *On finite-difference approximations and entropy conditions for shocks*, Comm. Pure Appl. Math., 29 (1976), pp. 297-322.

[8] G. W. HEDSTROM, *Models of difference schemes for* $u_t + u_x = 0$ *by partial differential equations*, Math. Comp., 29 (1975), pp. 969-977.

[9] P. HENRICI, *Discrete Variable Methods in Ordinary Differential Equations*, John Wiley, New York, 1963.

[10] A. HINDMARSH, P. M. GRESHO AND D. F. GRIFFITHS, *The stability of explicit Euler time-integration for certain finite difference approximations of the multi-dimensional advection-diffusion equation*, Int. J. Numer. Meth. Fluids., 4 (1984), pp. 853-897.

[11] C. W. HIRT, *Heuristic stability theory of finite difference equations*, J. Comp. Phys., 2 (1968), pp. 339-355.

[12] E. ISAACSON AND H. B. KELLER, *Analysis of Numerical Methods*, John Wiley, New York, 1966.

[13] J. D. LAMBERT, *Computational Methods in Ordinary Differential Equations*, John Wiley, London, 1973.

[14] A. LERAT, *Numerical shock structure and nonlinear corrections for difference schemes in conservation form*, Proc. 6th International Conference on Numerical Methods in Fluid Dynamics, H. Cabannes et al., eds., Springer-Verlag, Berlin, 1979.

[15] A. R. MITCHELL AND D. F. GRIFFITHS, *The Finite Difference Method in Partial Differential Equations*, John Wiley and Sons, Chichester, 1980.

[16] K. W. MORTON, *Initial value problems by finite difference and other methods*, State of the Art in Numerical Analysis, D. A. H. Jacobs, ed., Academic Press, London, 1977, pp. 699-756.

[17] C. PALENCIA AND J. M. SANZ-SERNA, *Equivalence theorems for incomplete spaces: an appraisal*, IMA J. Numer. Anal., 4 (1984), pp. 109-115.

[18] ———, *An extension of the Lax-Richtmeyer theory*, Numer. Math., 44 (1984), pp. 279-283.

[19] R. D. RICHTMEYER AND K. W. MORTON, *Difference Methods for Initial-Value Problems*, John Wiley, Interscience, London, 1967.

[20] P. J. ROACHE, *Computational Fluid Dynamics*, Hermosa, Albuquerque, NM, 1976.

[21] J. M. SANZ-SERNA, *Studies in numerical nonlinear instability* I: *Why do leap-frog schemes go unstable*, this Journal, 6 (1985), pp. 923-938.

[22] ———, *Convergence of the Lambert-McLeod trajectory solver and of the* CELF *method*, Numer. Math., 45 (1984), pp. 173-182.

[23] J. M. SANZ-SERNA AND C. PALENCIA, *A general equivalence theorem in the theory of discretization methods*, Math. Comp., to appear.

[24] J. M. SANZ-SERNA AND J. G. VERWER, *A Study of the recursion* $y_{n+1} = y_n + \tau y_n^m$, Centrum voor Wiskunde en Informatica, Report NM-R8403, Amsterdam, 1984.

[25] L. F. SHAMPINE AND M. K. GORDON, *Computer Solution of Ordinary Differential Equations*, W. H. Freeman, San Francisco, CA, 1975.

[26] Y. I. SHOKIN, *The Method of Differential Approximation*, Springer-Verlag, Berlin, 1983.
[27] M. N. SPIJKER, *Equivalence theorems for non-linear finite-difference methods*, Lecture Notes in Mathematics, 395, R. Ansorge and W. Tornig, eds., Springer-Verlag, Berlin, 1974, pp. 109–122.
[28] H. J. STETTER, *Symmetric two-step algorithms for ordinary differential equations*, Computing, 5 (1970), pp. 267–280.
[29] ———, *Analysis of Discretization Methods for Ordinary Differential Equations*, Springer-Verlag, Berlin, 1973.
[30] G. STRANG, *Accurate partial difference methods* II. *Nonlinear problems*, Numer. Math., 6 (1964), pp. 37–46.
[31] F. STUMMEL, *Diskrete Konvergenz linearer Operatoren*, I, Math. Ann., 190 (1970), pp. 45–92.
[32] G. VAINIKKO, *Funktionalanalysis der Diskretisierungsmethoden*, Teubner, Leipzig, 1976.
[33] R. VICHNEVETSKY AND J. B. BOWLES, *Fourier Analysis of Numerical Approximations of Hyperbolic Equations*, SIAM Studies in Applied Mathematics, 5, Society for Industrial and Applied Mathematics, Philadelphia, PA, 1982.
[34] R. F. WARMING AND B. J. HYETT, *The modified equation approach to the stability and accuracy analysis of finite difference methods*, J. Comp. Phys., 14 (1974), pp. 159–179.
[35] P. WILDERS, *Minimization of dispersion in difference methods for hyperbolic conservation laws*, Ph.D. thesis, Mathematisch Centrum, Amsterdam, 1983.
[36] A. LERAT AND R. PEYRET, *Propriétés dispersives et dissipatives d'une classe de schémas aux différénces pour les systèmes hyperboliques non linéaires*, Rech. Aérosp., 2 (1975), pp. 61–79.
[37] J. M. SANZ-SERNA, *Stability and convergence in numerical analysis* I: *linear problems, a simple comprehensive account*, preprint (available from the author).

SIAM J. SCI. STAT. COMPUT.
Vol. 7, No. 3, July 1986

AN EXTERIOR POISSON SOLVER USING FAST DIRECT METHODS AND BOUNDARY INTEGRAL EQUATIONS WITH APPLICATIONS TO NONLINEAR POTENTIAL FLOW*

DAVID P. YOUNG†‡, ALEX C. WOO‡¶, JOHN E. BUSSOLETTI‡§ AND FORRESTER T. JOHNSON‡§

Abstract. A general method is developed combining fast direct methods and boundary integral equation methods to solve Poisson's equation on irregular exterior regions. The method requires $O(N \log N)$ operations where N is the number of grid points. Error estimates are given that hold for regions with corners and other boundary irregularities. Computational results are given in the context of computational aerodynamics for a two-dimensional lifting airfoil. Solutions of boundary integral equations for lifting and nonlifting aerodynamic configurations using preconditioned conjugate gradient are examined for varying degrees of thinness.

Key words. partial differential equations, fast direct methods, boundary integral equations, fast Poisson solvers, preconditioned conjugate gradient, transonic potential flow

1. Introduction. Fast direct methods have been used extensively in recent years to solve Poisson's equation on rectangular and other separable domains [1], [2]. Much work has been devoted to extending these methods to other elliptic partial differential equations and/or nonseparable domains. In particular, for irregular geometries the analogy of capacitance matrices with potential theory has been exploited by Proskurowski and Widlund [3], [4]. In this paper, we show how a consistent, second-order boundary integral discretization can be implemented using fast direct methods. The starting point is the classical theory of double- and single-layer potentials. If N is the number of grid points, our discretization enables a solution of Poisson's equation in $O(N \log N)$ opeations which retains the spectral properties of the boundary integral formulation. This discretization has certain advantages with regard to conditioning of the matrices, flexibility in boundary discretization, and computation of quantities such as surface pressures.

In § 2, we outline the hybrid method in the context of boundary integral (panel) methods. Section 3 explains how the boundary integral problem is approximated using an exterior fast solver and an error estimate is given. Section 4 presents some two-dimensional computational results. Section 5 explains some iterative techniques for solving the linear system resulting from the approximations given in § 3 (such techniques are necessary in three dimensions). It also contains the results of some numerical experiments with preconditioned conjugate gradient. In § 6 we put our work in the context of previous work in this area.

2. The hybrid method. Because of the extreme sensitivity of airfoil problems to small perturbations in geometry [5], [6], [7], panel (boundary integral) methods with their accurate representation of surfaces have long been standard for linear potential flow calculations. But implementations of these methods have not been optimal computationally. Fast direct methods and boundary integral methods can be combined for the Poisson problem with advantages over either method alone. Consider the boundary

* Received by the editors August 2, 1983, and in revised form February 1, 1985.

† Boeing Computer Services Company, Tubwila, Washington 98188. The work of this author was partially supported by the National Science Foundation under grant MCS 80-12220.

‡ The work of this author was partially supported by NASA Ames Contract NAS2-9830 (PAN AIR Development).

§ Boeing Military Airplane Co., Seattle, Washington 98124.

¶ NASA/Ames Research Center, Moffett Field, California 94035.

value problem

$$\Delta\phi = f \quad \text{in } \Omega, \qquad \alpha\phi + \beta\frac{\partial\phi}{\partial n} = g \quad \text{on } \partial\Omega, \tag{1}$$

for an exterior domain $\Omega \subseteq \mathbb{R}^2$. Extend f by zero outside Ω and write $\phi = \phi_1 + \phi_2$ with ϕ_1 and ϕ_2 satisfying

(i) $\Delta\phi_1 = f$ over $\mathbb{R}^2$ with arbitrary boundary conditions on $\partial\Omega$

$\phi_1 = O(\log R)$ as $R \to \infty$,

and

(ii) $\Delta\phi_2 = 0$ on Ω,

$$\alpha\phi_2 + \beta\frac{\partial\phi_2}{\partial n} = g - \alpha\phi_1 - \beta\frac{\partial\phi_1}{\partial n} \quad \text{on } \partial\Omega,$$

$\phi_2 = O(\log R)$ as $R \to \infty$, where $R = \sqrt{x^2 + y^2}$.

Solving problems (i) and (ii) can be made less expensive than solving the original Problem (1). Problem (i) can be solved using any exterior fast direct method. We have chosen Hockney's convolution algorithm [8] because of its simplicity and either Hockney's discrete Green's function or that given by James [9]. Problem (ii) is solved using a modified second-order boundary integral (panel) method. More details on boundary integral methods can be found in [5], [6], [7], [10]. The method we modified is described in detail in [5], [6], [11].

The panel method can be viewed as collocation on an integral equation derived from Green's third identity. The formulation we use solves both the exterior problem and a related interior problem using jumps in ϕ_2 and $\partial\phi_2/\partial n$ called doublet and source distributions.

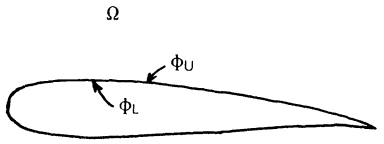


FIG. 1. *Domain description.*

If ϕ_L denotes the limiting value of ϕ_2 from the inside and ϕ_U the limiting value from the outside as in Fig. 1, we let

$$\mu = [[\phi]] \equiv \phi_U - \phi_L$$

called the doublet, and

$$\sigma = \left[\left[\frac{\partial\phi}{\partial n}\right]\right] \equiv \left.\frac{\partial\phi}{\partial n}\right|_U - \left.\frac{\partial\phi}{\partial n}\right|_L$$

called the source. Now, ϕ_2 can be described in terms of its jumps on $\partial\Omega$ by

$$\phi_2(P) = \int_{\partial\Omega} \sigma(Q)K(P, Q)\, dl_Q - \int_{\partial\Omega} \mu(Q)\frac{\partial K}{\partial n}(P, Q)\, dl_Q \tag{2}$$

where $K(P, Q) = -(1/2\pi)\log R(P, Q)$ for the two-dimensional problem and $R(P, Q)$ is the Euclidean distance between P and Q.

The surface is discretized using piecewise linear elements (also called panels). The unknown sources are piecewise linear and the doublets piecewise quadratic. Collocation points are chosen at the geometric centers of the boundary elements. The method is second order and is described in detail in [5], [6], [11].

The conventional panel methods generate an aerodynamic influence coefficient matrix giving the influence of each singularity basis function on each collocation point. Each element in the matrix is computed explicitly using the geometric relationship between the panel and the collocation point. If p is the number of panels and N an equivalent number of grid points in space (i.e., $p^2 = O(N)$ in two dimensions), generating this influence matrix requires $O(p^2) = O(N)$ operations although the coefficient is large. Solving the linear system directly involves $O(p^3) = O(N^{3/2})$ operations. In three dimensions, $p = O(N^{2/3})$ and the matrix generation requires $O(N^{4/3})$ operations while solving the linear system requires $O(N^2)$. This motivates the search for methods that avoid explicit generation of the matrix and the direct solution of the linear system. A panel or boundary integral code only gives solution values at the collocation points on the surface without special postprocessing. In conventional panel codes this is done with explicit integral evaluations similar to those used to generate the influence coefficient matrix. In two dimensions, for N grid points, the cost of evaluating ϕ_2 at all the grid points as required for the Poisson problem is $O(Np) = O(N^{3/2})$ operations. Because of the large constant, this is the greatest cost in two dimensions and for moderate N would dominate the cost in a three-dimensional code.

Our solution to the problem of finding ϕ_2 at all grid points is described below. It enables us to avoid generating and directly solving the boundary correction matrix. The same methods and error estimates are applicable to both the fast computation of ϕ_2 at all grid points and the fast computation of a matrix-vector multiple as required for a method such as conjugate gradient. This follows since multiplying the boundary correction matrix by a vector gives boundary values of the unknown potential ϕ_2 and its normal derivative $\partial\phi_2/\partial n$ at the collocation points. These values can instead be obtained directly by extrapolation once these quantities are known away from the boundary. The methods described below also enable the use of simple relaxation schemes and matrix splitting techniques where products of certain parts of the matrix with vectors can be computed without explicit generation of the matrix

3. Description of the method. The method outlined above for solving (1) has five steps:

1. ϕ_1 is computed using Hockney's exterior convolution algorithm. ϕ_1 satisfies $\Delta_h\phi_1 = f$ where Δ_h is the standard second-order, five-point finite difference approximation to Δ.

2. ϕ_1 and $\nabla\phi_1$ are interpolated to panel centers for use in the boundary correction algorithm. These functions are assumed smooth and the interpolation should be second order.

3. Boundary conditions for ϕ_2 are determined from $\alpha\phi_2 + \beta\,\partial\phi_2/\partial n = g - \alpha\phi_1 - \beta\,\partial\phi_1/\partial n$ at panel centers.

4. The boundary integral equation (2) is solved for the source and doublet strengths σ and μ on the boundary used collocation. In two dimensions, this can be done without excessive cost by directly computing the influence matrix and solving the linear system by Gaussian elimination. In three dimensions, an approach using iterative methods without explicit generation of the matrix is required. Such a scheme is described in § 5.

5. ϕ_2 is computed at the grid points using discrete charges to compute accurate far-field influences of the known surface singularities. This also requires one call to the exterior solver.

Our main emphasis in the rest of this paper will be on step 5. We will show how step 4 can be similarly accomplished for three-dimensional problems at a cost of $O(N \log N)$ operations (see § 5).

Assume that a source distribution σ on a flat panel p embedded in a regular grid is known (the dipole distribution will be discussed presently). This assumption is correct for step 5 and also for step 4 if a method such as conjugate gradient is used (see § 5 below). Two methods will be examined for representing the influence of the source distribution σ on a point P by charges on the grid points (see Fig. 2).

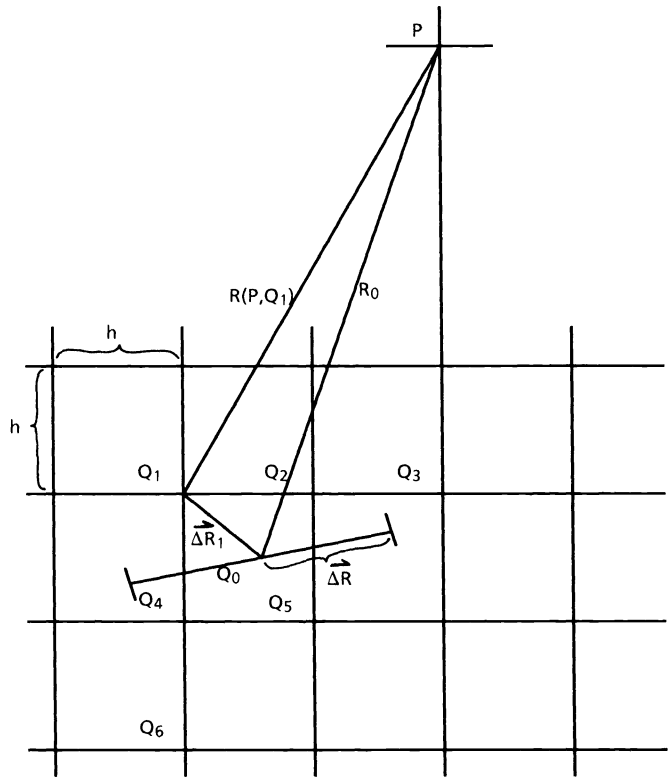


FIG. 2. *Arrangement of charges for far field influences.*

Method I. An approximation of the form

$$\phi_p(P) = \frac{1}{2\pi} \int_p \sigma(Q) \log R(P, Q)\, dl_Q \simeq \sum_i q_i \log R(P, Q_i) = q_{p^*} G = \tilde{\phi}_p(P)$$

is desired of as high an order as possible where G is Hockney's discrete Green's function given by

$$G(Q_i, Q_j) = \begin{cases} (1/2\pi) \log R(Q_i, Q_j) & \text{if } i \neq j, \\ 1 & \text{if } i = j. \end{cases}$$

Expanding $\log R(P, Q)$ and $\log R(P, Q_i)$ in Taylor series about the panel center Q_0 for large R, multiplying by σ, and integrating over the panel p yields

$$\int_p \sigma \log R\, dl = \log R_0 \int_p \sigma\, dl + \frac{\vec{R}_0}{R_0^2} \cdot \int_p \sigma \vec{\Delta} R\, dl$$
$$+ \frac{1}{2}\frac{(Y_0^2 - X_0^2)}{R_0^4} \int_p \sigma a^2\, dl - \frac{2X_0 Y_0}{R_0^4} \int_p \sigma ab\, dl$$
$$+ \frac{1}{2}\frac{(X_0^2 - Y_0^2)}{R_0^4} \int_p \sigma b^2\, dl + \cdots,$$

where $Q_0 = (X_0, Y_0)$, $\overrightarrow{\Delta R} = (a, b)$, and $\overrightarrow{\Delta R}_i = (a_i, b_i)$. Similarly, expanding each discrete charge about Q_0,

$$q_i \log R_i = q_i \log R_0 + \frac{\vec{R}_0}{R_0^2} \cdot q_i \overrightarrow{\Delta R}_i + \frac{1}{2}\frac{(Y_0^2 - X_0^2)}{R_0^4} q_i a_i^2$$
$$- \frac{2X_0 Y_0}{R_0^4} q_i a_i b_i + \frac{1}{2}\frac{(X_0^2 - Y_0^2)}{R_0^4} q_i b_i^2 + \cdots.$$

Matching terms in the expansions requires

$$\sum q_i = \int_p \sigma\, dl,$$

$$\sum q_i \overrightarrow{\Delta R}_i = \int_p \sigma \overrightarrow{\Delta R}\, dl,$$

$$\sum q_i a_i^2 = \int_p \sigma a^2\, dl,$$

$$\sum q_i a_i b_i = \int_p \sigma ab\, dl,$$

$$\sum q_i b_i^2 = \int_p \sigma b^2\, dl.$$

This gives six conditions to determine the six unknown charges $q_1, q_2, \cdots, q_6$. As is shown by Isaacson and Keller [12], if the points (a_i, b_i) form a triangular array, the two-dimensional interpolation problem equivalent to the above system has a unique solution. The error term is $O(h^3/R_0^3)$.

If the panels are assumed to be of length $O(h)$ and are flat, the error for the dipole distribution can also be made $O(h^3/R_0^3)$. The expansions are more complicated but the terms to be matched are

$$\sum q_i = 0,$$

$$\sum q_l \overrightarrow{\Delta R_i} = \int_p \mu \vec{n}\, dl,$$

$$\tfrac{1}{2} \sum q_i a_i^2 = n_x \int_p \mu a\, dl,$$

$$\sum q_i a_i b_i = n_x \int_p \mu b \, dl + n_y \int_p \mu a \, dl,$$

$$\tfrac{1}{2} \sum q_i b_i^2 = n_y \int_p \mu b \, dl,$$

where $\vec{n} = (n_x, n_y)$ is the unit normal to the panel.

The result is a charge distribution q_p such that $\phi_p(P) \cong q_p * G$. By linearity, we can superimpose charge distributions to obtain the total potential

$$\phi_T(P) = \frac{1}{2\pi} \int_{\partial\Omega} \sigma \log R \, dl = \sum_{p \in \partial\Omega} \frac{1}{2\pi} \int_p \sigma \log R \, dl$$

$$= \left(\sum_p q_p \right) * G + e \equiv \tilde{\phi}_T(P) + e \quad \text{where } e \text{ is the error term.}$$

In two dimensions, there are $O(1/h) = O(N)$ panels, so the error for ϕ_T is $\sum O(h^3/R_0^3) = O(h^2/\bar{R}^3)$, where the summation is over all p, $\bar{R}$ is the minimum of the distances from the point P to all panel ceters and R_0 is the distance from P to the geometric center of p.

The error corresponding to points P near some panel may thus be large. Hence, the total potential $\tilde{\phi}_T$ obtained from the exterior convolution algorithm using the sum of all the discrete charges must be corrected. This correction is accomplished point by point as follows.

Cycle through the points P in the grid. If P is sufficiently close to any panel p, i.e., closer than some empirically determined constant distance k, $\tilde{\phi}_T(P)$ must be corrected for this panel's influence. Because there are $O(1/h)$ panels we want to keep $h^3/R_{p,P}^3 \leqq Kh^3$ for each grid point panel pair for which no correction is needed. Thus, $k\sqrt[3]{1/K} \leqq R_{p,P}$ is required of the point P for $\tilde{\phi}_p$ to be accurate enough there. The correction is done by subtracting the discrete contribution and adding the surface integral,

$$\bar{\phi}_T(P) = \tilde{\phi}_T(P) - \sum_{i=1}^{6} q_i G_i(P, Q_i) + \frac{1}{2\pi} \int_p \sigma(Q) \log R(P, Q) \, dl_Q.$$

The total error in $\bar{\phi}_T$ after all corrections is

$$\bar{e} \leqq \sum_p C \frac{h^3}{R^3} \leqq C\left(\frac{1}{h}\right)\left(\frac{h^3}{k}\right) = O(h^2),$$

where the sum is over all panels p such that $R_0 \geqq k$.

The number of points at which corrections are required is the number of panels times the area of the circle of radius $\sqrt[3]{K}$ divided by h^2 and is thus $O((1/h) \cdot (C_1/h^2)) = O(N^{3/2})$.

By taking three more terms in the Taylor series, the error term for a single panel becomes $O(h^6/R^6)$. Hence, for a second order method we require $h^6/R^6 \leqq Kh^3$ or $\sqrt{h}/\sqrt[6]{K} \leqq R$. In this case the total number of corrections required is $O((1/h)) \cdot (h/h^2)) = O(N)$. A similar analysis can be made for dipole distributions assuming flat panels. Thus, the entire sequence of steps requires $O(N \log N)$ operations.

An implementation of a first order variant of this scheme as outlined in [13] can be found in the doctoral thesis of K. Halasi [14].

Method II. This method is more closely related to the convolution algorithm and is the one used in our code. It consists of four steps:

1. Compute $\tilde{\phi} = (1/2\pi) \int_p \sigma(Q) \log R(P, Q)\, dl_Q$ using exact near field integral computations for each grid point P within a circle of radius $\bar{R}$ (determined empirically) about the panel center.

2. Compute finite differences of $\tilde{\phi}$ to obtain charges $\bar{q} = \Delta_h \tilde{\phi}$ on the grid. $\bar{q}$ will be zero outside the circle.

3. In two dimensions in order to get the correct far field behavior, it is necessary to adjust the charge values so that $\sum \bar{q} = \int_p \sigma\, dl_Q$ where the sum is taken over the whole grid. This correction is evenly distributed over the nonzero $\bar{q}$'s.

4. Compute ϕ satisfying $\Delta_h \phi = \bar{q}$ using Hockney's convolution algorithm with the discrete Green's function of James [9].

This method is similar to that of Mayo [15], [16], who used, however, fewer changes and required a sixth order boundary integral method. We implemented Method II because certain data structures turned out to be easier to code. However, Method I provides a more intuitive formulation of this type of procedure.

The error in this process can be estimated by integrating the neglected charges outside the circle. The error analysis yields the same results as for Method I, i.e., the total error in ϕ is $O(h^2/\bar{R}^3)$. As explained in § 5, this approximation is also used as part of an iterative method to find ϕ_2. The process amounts to using the above approximation for off-diagonal elements of the discrete boundary correction operator. Since the unmodified panel method is assumed to be second order, application of simple perturbation analysis such as that contained in [12, Thm. 3, p. 37] shows the presented method to be second order in h.

Computation of velocities from the potential is a somewhat complicated matter. First order accurate velocities can be obtained at half grid points by centered finite differencing of the potential. However, when the differencing stencil crosses a panel, a large error is introduced. We must, therefore, correct these velocities by subtracting the finite differences of the individual panel potential influences for those panels that are close to the point in question and then adding the exact panel influence velocities computed as in [5], [6] for those panels. More details can be found in [17]. This gives a sum of first and second order terms, the accuracy of the result depending on the near-field influence cutoff $\bar{R}$.

4. Computational results. Our code implements Method II and was run as a component of a nonlinear potential flow code for two-dimensional problems described in [17]. This code uses a preconditioned steepest descent algorithm with a scaled fast Poisson solver using the algorithm described above as a preconditioner. The accuracy of the Poisson solver was consistently checked against a standard panel code that evaluated ϕ_2 at all grid points using exact integrals of the singularity strengths. Method II was about 20 times faster than exact integral evaluations. Table 1 shows the errors

TABLE 1
Code vs. exact potential.

Grid	h	R	h/R	Absolute errors			Relative errors		
				ϕ	V_x	V_y	ϕ	V_x	V_y
33×33	.03	.06	.5	.18	.038	.019	.02	.06	.043
65×65	.015	.06	.25	.029	.010	.0065	.0032	.016	.015
33×33	.03	.12	.25	.030	.004	.003	.0033	.0062	.0066
65×65	.015	.09	.167	.010	.0027	.0018	.0011	.0043	.0044
65×65	.015	.12	.125	.0060	.0011	.0009	.0007	.0018	.0022

in the maximum norm obtained by such a comparison. The number of grid points in each direction is either 33 or 65 as noted. Also, h is the grid spacing, R the radius of the circle within which discrete charges were computed in step 2, h/R is the convergence error variable, V_x and V_y are the x and y components of velocity and ϕ is the potential. The region is the exterior of the NACA 0012 airfoil at zero angle of attack. The dipole and source distributions are those needed to impose homogeneous Neumann (impermeable) boundary conditions after volume sources due to nonlinear terms have been evaluated at a freestream Mach number of 0.7. Specifically, $\|\mu\|_\infty \sim 100$ where μ is the doublet strength.

Figure 3 shows a log plot of the absolute errors in ϕ times R versus h/R. The empirically observed convergence rate is very close to the theoretical prediction.

The whole nonlinear iteration sequence was also run for this problem. The result is the same as that obtained with the exact integral formulation and the same as that obtained from Jameson's code FLO42 to three digits.

5. Solution of the boundary integral problem. The boundary integral problem for the correction potential ϕ_2 can be solved by direct methods such as Gaussian elimination. Since the system is dense and in general nonsymmetric and since the number of boundary elements is relatively small in two dimensions, we have used Gaussian elimination in our two-dimensional code.

However, in three dimensions, iterative methods have the potential to be significantly cheaper ($O(N \log N)$ operations per iteration) than direct methods. This is true, in particular, when each iteration involves only matrix-vector multiplies or matrix-vector multiplies with minor adjustments in which case the boundary integral matrix does not need to be explicitly generated and stored. Such is the case for the conjugate gradient algorithm and the paired Jacobi relaxation used by Schippers [18]. Multigrid methods as developed by Schippers [18], Hempker and Schippers [19], and Sloof et al. [20] can also be used. We describe an iterative method below based on the method of conjugate gradients applied to the normal equations. An important advantage of using a consistent discretization of a boundary integral equation of the second kind is that the condition number of the matrix is asymptotically constant. This observation follows from the classical spectral properties of compact operators and the consistency of the discretization. For unions of rectangles, finite difference capacitance matrices are first-order discretizations of boundary integral equations and hence have asymptotically constant condition numbers. For finite element formulations, this result was proved by Dryja [21] and has recently been extended to the general case of smooth regions where the sides of the triangular elements are all $O(h)$ as $h \to 0$ [22]. For the finite difference formulations, Shieh [23], [24] has shown that at most a fixed number of singular values of the capacitance matrix are outside a fixed interval as $h \to 0$. An $N \log N$ operation count can be maintained by using a "sparse" fast solver when solving the capacitance matrix equation iteratively. However, the number of iterations required may grow slowly as $h \to 0$ even though in practice this does not seem to occur. This is due to the fact that the diagonal elements of the capacitance matrix need not be asymptotically the same size as the self-influences for the boundary integral equation.

Below we give results of some experiments using a two-dimensional panel aerodynamics code that solves the Prandtl-Glauert equation by discretizing the classical boundary integral equation [5], [6]. We have used preconditioned conjugate gradient to solve both lifting and nonlifting problems. A boundary integral equation of the second kind results from using impermeable boundary conditions, i.e, $\partial\phi/\partial n = 0$ and

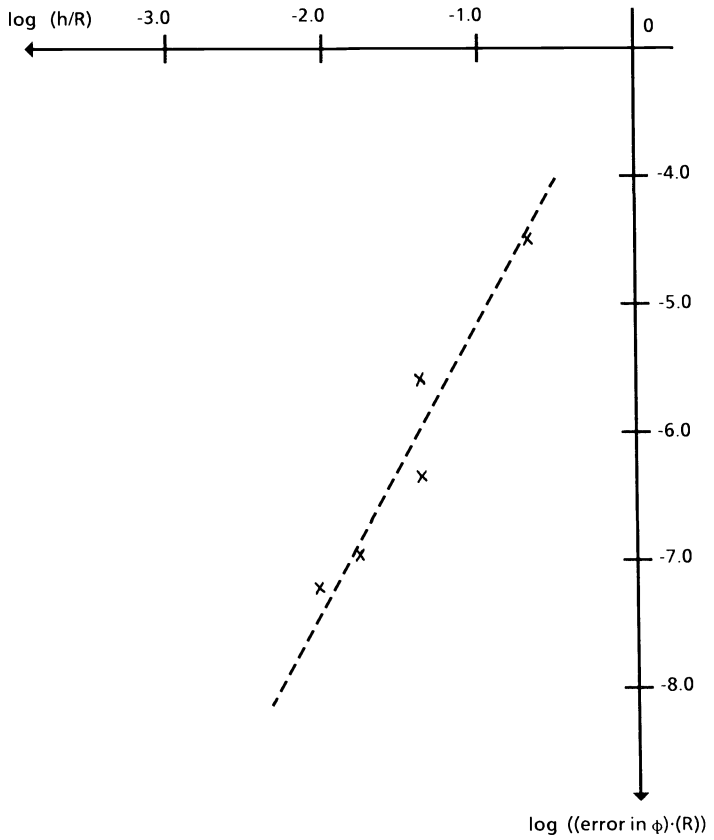


FIG. 3. *Plot of log of maximum errors in ϕ times R against* $\log(h/R)$ *for Poisson solver.*

sources alone on a nonlifting configuration. Table 2 shows how the condition number is related to thinness and Table 3 shows that it remains constant in this example as the number of panels is increased. All eigenvalues were computed using the EISPACK subroutine RGG.

TABLE 2
Condition numbers of boundary integral equation matrices.

Shape	Condition no.
Sphere	2.04
10% thick flattened sphere	11.0
1% thick flattened sphere with tip	52.1

To accelerate convergence for the thin configurations, a preconditioner consisting of the largest few elements in every row was used. Even with explicit computation of the boundary correction matrix and the matrix-vector products rather than implicit evaluation discussed below, we were able to obtain eight digits reduction in the residuals with significantly less work than with Gaussian elimination, while using relatively few surface elements. The results are shown in Table 4. NELRW is the number of elements

TABLE 3
Influence of panel density on condition number.

10% thick flattened sphere		1% thick flattened sphere with tip	
No. panels	Condition no.	No. panels	Condition no.
21	11.03	84	52.1
42	11.0	168	53.2
84	11.0		

TABLE 4
Preconditioned conjugate gradient applied to boundary integral matrices (Nonlifting configuration).

Problem	No. panels	NELRW	No. iterations to obtain 10^{-6} residual	Preconditioning flops / Total flops	Total C.G. flops / $1/3N^3$
10% thick	42	4	9	.16	1.53
10% thick	84	8	9	.21	.80
1% thick	84	8	9	.22	.78
1% thick	168	8	17	.11	.70
1% thick	168	16	10	.26	.49

selected from each row for the preconditioner. The largest elements in the row are always chosen so that the preconditioner consists of the near-field influences. Also shown is the number of operations associated with preconditioning (SPARSPAK [25] was used to solve the required linear system) divided by the total number of operations for the preconditioned conjugate gradient method and the total number of operations for the preconditioned CG algorithm divided by the number of operations for Gaussian elimination. The residual is measured in the L^1 norm.

For the lifting problem, the Kutta condition introduces a small eigenvalue resulting in a condition number of $10^3 - 10^4$. Preconditioning can ameliorate the influence of this eigenvalue on the convergence rate. Our preconditioner was chosen to include both the row and column associated with the Kutta condition as well as the largest elements in each row. Preliminary results of using this method on the NACA 0012 lifting airfoil at 2° angle of attack at mach .61 using our nonlinear potential flow code [17] are shown in Fig. 4. There were 110 panels in the configuration. Out of 12,761 matrix elements, 2593 were included in the preconditioner. After 20 iterations, the maximum residual was 4.03×10^{-7} and the Kutta condition was satisfied. Similar error reduction factors were observed throughout the nonlinear iteration sequence. The residuals were measured in the L^1 norm.

In an iterative solution of the boundary integral problem the product of the boundary integral matrix with a vector must be computed. Given an approximation to the surface singularities, the product of the boundary integral matrix with this vector is computed simply by determining resulting values of ϕ and $\partial\phi/\partial n$ at the centers of the boundary elements. This is exactly what step 5 in the algorithm of § 3 allows if the resulting ϕ and $\partial\phi/\partial n$ values can be accurately interpolated to points not on the grid. With near field corrections, this can be accomplished by interpolating only the smooth far field part of the solution, the near field influence integrals being computed explicitly. The adjoint of the influence matrix can be dealt with similarly since the

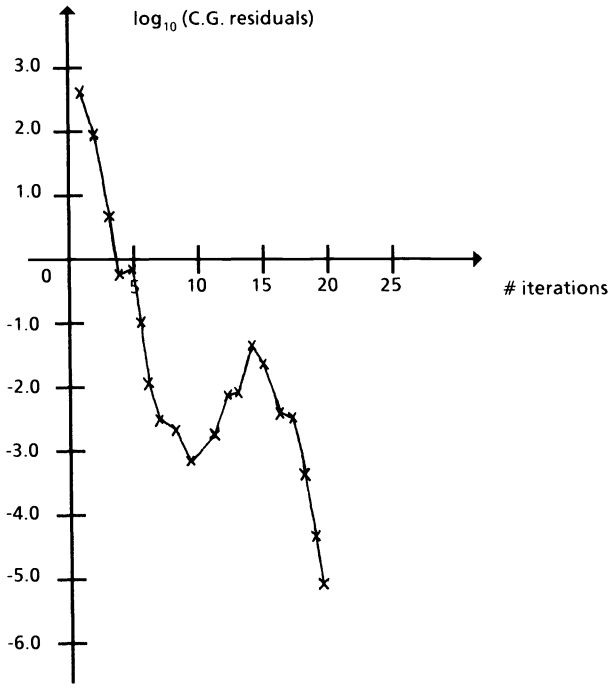


FIG. 4. *Preconditioned conjugate gradient applied to a lifting airfoil.*

adjoint of a single layer potential operator is another single layer operator and the adjoint of a double layer potential operator is a derivative of a single layer potential operator [26]. Incorporation of this implicit multiplication results in an algorithm that is asymptotically $O(N \log N)$ operations where N is the number of grid points for problems in which the boundary correction results in an integral equation of the second kind.

Schippers [18] has used a multigrid method on two-dimensional lifting airfoil problems using Jacobi, paired Gauss-Seidel, and paired Jacobi relaxation. These relaxation techniques can be carried out using the same near field-far field ideas without explicit generation of the matrix.

6. History and comparisons. The methods presented here can be viewed as generalizations of capacitance matrix techniques. Here we attempt to put our work in the context of this previous related work. The term capacitance matrix seems to have been coined by Hockney [8] and Buneman. The ideas were put in the context of the Woodbury formula by Buzbee, Dorr, George, and Golub [27]. Proskurowski and Widlund [3] generalized the formulation to second-order finite differences on arbitrary regions. They developed the analogy between this formulation and boundary integral equations and advocated the use of the conjugate gradient method to solve the capacitance matrix system. Shieh [23] analyzed the conditioning of capacitance marices for this formulation for the Neumann problem and showed that all but a finite number of singular values lie in some fixed interval bounded away from zero. He also showed an $N \log N$ operation count by using a "sparse" fast solver due to Banegas. The method was extended to three dimensions by O'Leary and Widlund [28] and to regular finite element discretizations by Proskurowski and Widlund [4]. The recent work of Dryja [21], [22] analyzes the finite element approach.

Mayo [15], [16] has applied fast direct methods to a boundary integral formulation for interior problems with smooth boundaries. A sixth-order boundary integral method was used to obtain a solution in $O(N \log N)$ operations.

Faber, White, and Sweet [29] have improved the conditioning of capacitance matrices by using large numbers of discrete charges while retaining the Woodbury formula. Far field approximations for boundary integral equations have been developed by Mike Epton and others at Boeing to improve the efficiency of conventional panel methods [11]. Another way to view the work in this paper (and the way it developed) is as a very general and efficient way of computing far field influences for panel methods. A preliminary report on these methods can be found in [30]. Application of these methods to incompressible Navier–Stokes equations was reported by Gustafson [31].

Recent surveys of fast direct methods include those of Hockney [32] and Stuben and Trottenburg [33]. The hybrid method described here has several advantages. One is that the condition number for integral equations of the second kind is bounded by a constant as the number of boundary elements grows. This is because it represents a consistent discretization of the boundary integral equation. Another advantage is that arbitrary geometries with local surface features can be handled without requiring the grid to go through such local features. Also, surface information is easily obtained from the singularities. More details of such calculations are given in [17].

Acknowledgments. We would like to acknowledge the support and assistance of Larry Erickson of the NASA-Ames Research Center and computing resources made available by NASA-Ames. The first author acknowledges the generous support of the National Science Foundation and the hospitality of the University of Colorado, the National Center for Atmospheric Research, and Professor Karl Gustafson in Boulder, Colorado. He also wishes to acknowledge the programming support of Kadosa Halasi of the University of Colorado at Boulder and computing resources made available at the National Center for Atmospheric Research and the University of Colorado. We also thank Jan Petersen of Boeing Computer Services who did the technical typing.

REFERENCES

[1] P. N. SWARZTRAUBER AND R. A. SWEET, *Efficient Fortran subprograms for the solution of separable elliptic partial differential equations*, ACM Trans. Math. Software, 5 (1979), pp. 352–364.

[2] R. W. HOCKNEY AND J. W. EASTWOOD, *Computer Simulation Using Particles*, McGraw-Hill, New York, 1981.

[3] W. PROSKUROWSKI AND O. WIDLUND, *On the numerical solution of Helmholtz's equation by the capacitance matrix method*, Math. Comp., 30 (1976), pp. 433–468.

[4] ———, *A finite element-capacitance matrix method for the Neumann problem for Laplace's equation*, this Journal, 1 (1980), pp. 410–424.

[5] F. T. JOHNSON, *A general panel method for the analysis and design of arbitrary configurations in subsonic flows*, NASA Contractor Report CR-3079, NASA-Ames Research Center, Moffett Field, CA, 1979.

[6] F. T. JOHNSON AND P. E. RUBBERT, *Advanced panel-type influence coefficient methods applied to subsonic flows*, AIAA Paper 75-50, 1975.

[7] J. L. HESS AND A. M. O. SMITH, *Calculation of nonlifting potential flow about arbitrary three-dimensional bodies*, ES40622, Douglas Aircraft Company, 1962.

[8] R. W. HOCKNEY, *The Potential Calculation and Some Applications*, in Methods in Computational Physics, Vol. 9, Academic Press, New York and London, 1969.

[9] R. A. JAMES, *The solution of Poisson's equation for isolated source distributions*, J. Comput. Phys., 25 (1977), pp. 71–93.

[10] G. FAIRWEATHER, F. J. RIZZO, D. J. SHIPPY AND J. S. WU, *On the numerical solution of two-dimensional potential problems by an improved boundary integral method*, J. Comput. Phys., 31 (1979), pp. 96–112.

[11] ALFRED E. MAGNUS AND MICHAEL A. EPTON, *Pan air—a computer program for predicting subsonic or supersonic linear potential flows about arbitrary configurations using a higher order panel method*, NASA Contractor Report 3251, NASA-Ames Research Center, Moffett Field, CA, 1979.

[12] E. ISAACSON AND H. B. KELLER, *Analysis of Numerical Methods*, John Wiley, New York, 1966.

[13] D. P. YOUNG, *A hybrid algorithm for Poisson's equation*, Boeing internal report, 1981.

[14] K. HALASI, *Numerical solution of two dimensional potential problems using Fourier analysis, boundary integral equations, and near-far field concepts*, Ph.D. Thesis, Univ. Colorado at Boulder, 1982.

[15] A. MAYO, *The fast solution of Poisson's and the biharmonic equations on irregular regions*, SIAM J. Numer. Anal., 21 (1984), pp. 285-299.

[16] ———, *The fast solution of Poisson's and the biharmonic equation on irregular regions*, in Advances in Computer Methods for Partial Differential Equations-IV, R. Vichnevetsky and R. S. Stepleman, eds., IMACS, New Brunswick, NJ, 1981, pp. 107-111.

[17] F. T. JOHNSON, R. M. JAMES, J. E. BUSSOLETTI, A. C. WOO AND D. P. YOUNG, *A transonic rectangular grid embedded panel method*, AIAA paper 82-0953.

[18] H. SCHIPPERS, *Application of multigrid methods for integral equations to two problems from fluid dynamics*, MC Preprint NW111/81, Mathematical Centre, Amsterdam, 1980.

[19] P. W. HEMKER AND H. SCHIPPERS, *Multiple grid methods for the solution of Fredholm integral equations of the second kind*, Math. Comp., 36 (1981), pp. 215-232.

[20] J. W. SLOOF, *Requirements and developments shaping a next generation of integral methods*, National Aerospace Laboratory NLR MP81007U, Amsterdam, 1981.

[21] M. DRYJA, *A capacitance matrix method for the Dirichlet problem on polygon regions*, Numer. Math., 39 (1982), pp. 51-64.

[22] ———, *A finite element-capacitance matrix method for the elliptic problem*, SIAM J. Numer. Anal., 20 (1983), pp. 671-680.

[23] A. S. L. SHIEH, *On the convergence of the conjugate gradient method for singular capacitance matrix equations from the Neumann problem of the Poisson equation*, Numer. Math., 29 (1978), pp. 307-327.

[24] ———, *Fast Poisson solvers on general two-dimensional regions for the Dirichlet problem*, Numer. Math., 31 (1979), pp. 405-429.

[25] ALAN GEORGE AND JOSEPH W-H. LIU, *Computer Solution of Large Sparse Positive Definite Systems*, Prentice-Hall, Inc., Englewood Cliffs, NJ, 1981.

[26] I. G. PETROVSKII, *Partial Differential Equations*, W. B. Saunders, Philadelphia, PA, 1967.

[27] B. L. BUZBEE, F. W. DORR, J. A. GEORGE AND G. H. GOLUB, *The direct solution of the discrete Poisson equation on irregular regions*, SIAM J. Numer. Anal., 8 (1971), pp. 722-736.

[28] D. P. O'LEARY AND O. WIDLUND, *Capacitance matrix methods for the Helmholtz equation on general three-dimensional regions*, Math. Comp., 33 (1979), pp. 849-879.

[29] V. FABER, A. WHITE AND R. SWEET, *An extension of the implicit capacitance matrix method algorithm for solving Poisson equations on irregular regions*, in Advances in Computer Methods for Partial Differential Equations-IV, R. Vichnevetsky and R. S. Stepleman, eds., IMACS, New Brunswick, NJ, 1981, pp. 339-342.

[30] D. P. YOUNG, *Software for Poisson problems on regions with complex boundaries using aerodynamics codes*, in Advances in Computer Methods for Partial Differential Equations-IV, R. Vichnevetsky and R. S. Stepleman, eds., IMACS, New Brunswick, NJ, 1981, p. 192.

[31] K. GUSTAFSON, *Hybrid fast Poisson solvers for fluid dynamics*, Proc. 10th IMACS World Congress, Montreal, 1982, Vol. 1, pp. 233-235.

[32] R. W. HOCKNEY, *Rapid elliptic solvers*, in Numerical Methods in Applied Fluid Dynamics, B. Hunt, ed., Academic Press, London, 1980, pp. 1-48.

[33] K. STUBEN AND U. TROTTENBURG, *On the construction of fast solvers for elliptic equations*, Gesellschaft fur Mathematik und Datenverarbeitung IMA-Report No. 82-0201, Bonn, 1982 (and also Von Karman Institute for Fluid Dynamics Lecture Series 1982-6).

SIAM J. SCI. STAT. COMPUT.
Vol. 7, No. 3, July 1986

022

COMPUTATION OF THE CHI-SQUARE AND POISSON DISTRIBUTION*

LEO KNÜSEL†

Abstract. The paper deals with the computation of upper and lower tail probabilities of the chi-square and Poisson distribution with a specified relative accuracy on both tails for virtually all possible parameter values. With some supplement the proposed algorithms will also work for the general gamma distribution. If the parameters are small, open forward and backward recursion is used for the summation with an adaptive number of steps depending on the specified accuracy. For large parameters asymptotic expansions related to the central limit theorem are applied for the approximation. The basic ideas of the proposed methods will also be applicable to other elementary statistical distributions such as the binomial, beta, and F-distribution as well as the hypergeometric distribution.

Key words. Poisson distribution, chi-square distribution, gamma distribution, open adaptive recursion, asymptotic expansions

AMS(MOS) subject classifications. 60-04, 60E99, 62-04, 65D20, 65U05

1. Introduction and summary. The paper deals with the computation of upper and lower tail probabilities of the chi-square and Poisson distribution with a specified relative accuracy on both tails for virtually all parameter values. With some supplement the proposed algorithms will also work for the general gamma distribution. The basic ideas of the proposed methods will also be applicable to other elementary statistical distributions such as the binomial, beta, and F-distribution as well as the hypergeometric distribution.

In § 2 the desired probabilities are reduced to three building blocks $I(a, x)$, $J(a, x)$, and $p(a, x)$. In §§ 3 and 4 the first two quantities are computed by simple and very stable algorithms where the adaptive number of steps depends on the arguments and the specified relative accuracy. In § 5 we propose a logarithmic procedure to compute the quantity $p(a, x)$ and in order to reduce cancellation effects we suggest using a scaled version of the gamma function and a shifted version of the log-function. Section 6 deals with asymptotic expansions of the desired tail probabilities as a whole and of the three building blocks individually. Proofs and derivations are not given in this paper and can be found with further references in Knüsel (1981).

2. Basic building blocks. We define the following quantities:

$$p(a, x) = \frac{e^{-x} x^{a-1}}{\Gamma(a)} \quad \text{where } \Gamma(a) = \int_0^\infty t^{a-1} e^{-t}\, dt,$$

$$I(a, x) = x e^x \int_0^1 t^{a-1} e^{-xt}\, dt, \tag{2.1}$$

$$J(a, x) = x e^x \int_1^\infty t^{a-1} e^{-xt}\, dt,$$

$(a > 0,\ x > 0)$. The following identity holds true:

$$[I(a, x) + J(a, x)]\, p(a, x) = 1. \tag{2.2}$$

* Received by the editors January 12, 1983, and in final revised form June 17, 1985.

† Institut für Statistik und Wissenschaftstheorie, Universität München, D-8000 München 22, Federal Republic of Germany.

Now, let X denote a random variable having a gamma distribution with parameter $a>0$:

$$\Pr\{X<x\}=\frac{1}{\Gamma(a)}\int_0^x t^{a-1}\,e^{-t}\,dt \quad \text{for } x>0. \tag{2.3}$$

In terms of p, I and J we have

$$\begin{aligned}\Pr\{X<x\}&=p(a,x)\cdot I(a,x),\\ \Pr\{X>x\}&=p(a,x)\cdot J(a,x).\end{aligned} \tag{2.4}$$

Next, let Y denote a random variable having a chi-square distribution with f degrees of freedom ($f=1,2,\cdots$):

$$\Pr\{Y<y\}=\frac{1}{2\Gamma(f/2)}\int_0^y (t/2)^{(f/2)-1}\,e^{-t/2}\,dt \quad \text{for } y>0. \tag{2.5}$$

Then the random variable $X=Y/2$ has a gamma distribution with parameter $a=f/2$. In terms of p, I and J we can write

$$\begin{aligned}\Pr\{Y<y\}&=p(a,x)\cdot I(a,x),\\ \Pr\{Y>y\}&=p(a,x)\cdot J(a,x),\end{aligned} \tag{2.6}$$

where $a=f/2$ and $x=y/2$.

Finally, let Z denote a random variable having a Poisson distribution with parameter $\lambda>0$. As opposed to the gamma and chi-square distribution, which are continuous distributions, the Poisson distribution is a discrete distribution:

$$\Pr\{Z=k\}=\frac{e^{-\lambda}\lambda^k}{k!} \quad \text{for } k=0,1,\cdots. \tag{2.7}$$

In terms of p, I, and J we can write

$$\begin{aligned}\Pr\{Z=k\}&=p(a,x),\\ \Pr\{Z\leqq k\}&=p(a,x)\cdot J(a,x),\\ \Pr\{Z>k\}&=p(a,x)\cdot I(a,x),\end{aligned} \tag{2.8}$$

where $a=k+1$ and $x=\lambda$.

As we have seen, the gamma, chi-square and Poisson distribution can be reduced to the same building blocks p, I, and J. The parameter a is a real number in case of the gamma distribution, half of an integer in case of the chi-square distribution and an integer in case of the Poisson distribution. Thus we look for reliable algorithms to compute p, I, and J, and in view of (2.2) only the smaller of the two quantities I and J needs to be computed directly.

3. Computation of $I(a,x)$. From definition (2.1) we derive the following properties of $I_a=I(a,x)$ for any given $x>0$:

$$\begin{aligned}&\text{a)} \quad I_a\downarrow 0 \quad \text{for } a\uparrow\infty,\\ &\text{b)} \quad I_a=\frac{x}{a}(1+I_{a+1}) \quad \text{for } a>0.\end{aligned} \tag{3.1}$$

Thus the following algorithm (open backward recursion) will yield an approximation

to I_a:

(3.2)
$$\text{Set } \tilde{I}_{a+n} = 0 \text{ for some positive integer } n.$$
$$\text{For } b = a+n-1, a+n-2, \cdots, a:$$
$$\tilde{I}_b = \frac{x}{b}(1+\tilde{I}_{b+1}).$$
$$\text{Then } \tilde{I}_a \text{ is an approximation to } I_a = I(a,x).$$

From (3.2) we derive

(3.3)
$$I_a - \tilde{I}_a = fac \cdot I_{a+n}$$

where

(3.4)
$$fac = \frac{x^n}{a(a+1)\cdots(a+n-1)}$$

and because of the monotonicity property in (3.1) we have the following inequality for the relative error of the approximation $\tilde{I}_a$:

(3.5)
$$\text{rel. err.} = |\tilde{I}_a - I_a| / I_a = fac \cdot (I_{a+n}/I_a) < fac.$$

Therefore we can guarantee a relative error of less than a given $\varepsilon > 0$ if we

(3.6) choose n as the smallest positive integer such that $fac < \varepsilon$.

(3.2) and (3.6) constitute the algorithm we propose to compute $I(a,x)$. It is an adaptive algorithm where the number of steps depends on the arguments a and x and on the given relative accuracy ε.

What can be said about the number n of steps required to achieve a given accuracy? We remember that only the smaller of the two quantities $I(a,x)$ and $J(a,x)$ needs to be computed directly, and in (4.14) we shall give the rule to use the algorithm (3.2) only when $a > x$. For given x and ε and on condition $a \geqq x$ the number n attains its maximum for $a = x$. For large x ($x > 100$ or so) and $a = x$ we have approximately

(3.7)
$$n \approx t\sqrt{x} \quad \text{where } e^{-t^2/2} = \varepsilon$$

(e.g. $t = 5.26$ for $\varepsilon = 10^{-6}$; $t = 7.43$ for $\varepsilon = 10^{-12}$).

The algorithm (3.2) actually computes the sum

(3.8)
$$\tilde{I}_a = \frac{x}{a} + \frac{x^2}{a(a+1)} + \cdots + \frac{x^n}{a(a+1)\cdots(a+n-1)}.$$

The terms of this sum are all positive, and for $a \geqq x$ they are monotonically decreasing. The algorithm performs the summation in the ideal way namely from small to large terms by nested multiplications. Therefore the algorithm is very stable, and on condition $a \geqq x$ the relative rounding error can be bounded by a term of order $\eta\sqrt{x}$ where η is the relative machine accuracy that is the smallest real machine number with $1+\eta > 1$ (cf. Knüsel (1981)).

We mention some further properties of $I(a,x)$ that can be useful when implementing the algorithm. We can write

(3.9)
$$I(a,x) = x \int_0^1 t^{a-1} e^{x(1-t)}\, dt$$

and this shows that $I(a,x)$ is an increasing function of x for any $a > 0$. From (3.1)

we derive for $a > x$

$$q < I(a, x) < q/(1-q) \quad \text{where } q = x/a \tag{3.10}$$

and for $x = a$, a large, we find from the asymptotic expansion (6.4) given in § 6

$$I(a, a) \approx \sqrt{a\pi/2} = 1.253\sqrt{a}. \tag{3.11}$$

On a machine with 10 decimal digits the adaptive open backward recursion given by (3.2) and (3.6) can guarantee a relative accuracy of less than $\varepsilon = 10^{-6}$ for values of a up to $a_{\max} = 10^6$ since the rounding error will be much smaller than ε though the maximum number of steps required would be about 5300. If computing time is a consideration, however, $I(a, x)$ can be computed more efficiently for large a and x by means of the asymptotic expansions given in § 6.

From (3.8) it is seen that the algorithm (3.2) is related to the method proposed by Bhattacharjee (1970) but there the summation is performed from large to small terms. See also Kennedy and Gentle (1980, p. 117).

4. Computation of $J(a, x)$. The quantity $J(a, x)$ in (2.1) is defined not only for positive a but for any real a. For $a = 0$ we have

$$J(0, x) = x\, e^x \int_x^\infty e^{-u}/u \, du \tag{4.1}$$

and the integral on the right-hand side is the so-called exponential integral.

The following properties can be derived for $J_a = J(a, x)$ for any given $x > 0$:

$$\text{a)} \quad J_a \downarrow 0 \quad \text{for } a \downarrow -\infty,$$

$$\text{b)} \quad J_{a+1} = 1 + \frac{a}{x} J_a \quad \text{for any real } a, \tag{4.2}$$

$$\text{c)} \quad J_1 = 1.$$

Comparing (4.2) with (3.1) we observe that J_a can be computed in much the same way as I_a, but this time the recursion has to work in the forward direction and as initial value we can use one instead of zero which saves us the first step in the iteration:

$$\begin{aligned} &\text{Set } \tilde{J}_{a-n+1} = 1 \text{ for some positive integer } n. \\ &\text{For } b = a-n+1, a-n+2, \cdots, a-1: \\ &\qquad \tilde{J}_{b+1} = 1 + \frac{b}{x} \tilde{J}_b. \\ &\text{Then } \tilde{J}_a \text{ is an approximation to } J_a = J(a, x). \end{aligned} \tag{4.3}$$

For the relative error of the approximation $\tilde{J}_a$ we obtain the inequality

$$\text{rel. err.} = |\tilde{J}_a - J_a|/J_a = |fac| \cdot (J_{a-n}/J_a) < |fac|, \tag{4.4}$$

where

$$fac = (a-1)(a-2) \cdots (a-n)/x^n. \tag{4.5}$$

We point out that n may be larger than a, but note that for $n < a + x$ and $a \leqq x$ each of the n terms $(a-j)/x$ in (4.5) is smaller than one in magnitude.

In view of (4.4) we can guarantee a relative error of less than a given $\varepsilon > 0$ if we

$$\text{choose } n \ (n \leqq a + x) \text{ as the smallest positive integer such that } |fac| < \varepsilon. \tag{4.6}$$

If such an integer n does not exist we compute J_a by complete forward recursion to be based upon the exact initial value J_b, $0<b\leqq 1$, and only rounding errors will be incurred in this case. Note that complete recursion is required only for small values of x and a. As an example, a desired relative accuracy of $\varepsilon=10^{-6}$ (or $\varepsilon=10^{-12}$) is already achieved by an incomplete recursion with the approximate initial value $\tilde{J}_{a-n+1}=1$ provided that $x\geqq 16.2$ ($x\geqq 30.3$ respectively) and $0\leqq a\leqq x$.

As to the exact initial value we note that

$$\begin{aligned} &J(0.5, x)=\sqrt{\pi x}\exp(x)\operatorname{erfc}(\sqrt{x})(<1) && \text{for } x>0,\\ &J(1, x)=1 && \text{for } x>0, \end{aligned} \tag{4.7}$$

where erfc is the complementary error function

$$\operatorname{erfc}(y)=(2/\sqrt{\pi})\int_y^{\infty}\exp(-t^2)\,dt \quad \text{for } -\infty<y<+\infty. \tag{4.8}$$

We need not bother about overflow and underflow problems in evaluating $J(0.5, x)$ as given by (4.7) since we can confine ourselves to $0.5\leqq x<20$ or so. This is because we have to compute $J(a, x)$ only for $a\leqq x$ and because complete recursion is required only for small values of x as we have seen in the last paragraph.

Thus there are no unsolved problems with the initial value in case of the Poisson and chi-square distribution where a and $2a$ respectively are integers. For the general gamma distribution, however, we would require the exact value of $J(b, x)$ for any real b with $0<b<1$ and $0<x<20$ or so in case complete recursion has to be used. A routine doing this job is not described in this paper.

The number n of steps required by the forward recursion (4.3) and (4.6) to achieve a given accuracy again depends on a and x. Since we will apply forward recursion only if $a\leqq x$ the maximum number of steps for given a will be attained when $a=x$, and for large a the same approximation formula as in case of the backward recursion, given in (3.7), holds true.

The following properties can prove useful when implementing the proposed algorithm. $J(a, x)$ can be written as

$$J(a, x)=\int_0^{\infty} e^{-u}\left(1+\frac{u}{x}\right)^{a-1} du \tag{4.9}$$

and this shows that $J(a, x)$ is not only a monotonous function in the argument a but also in x:

$$\begin{aligned} &J(a, x)\downarrow J(1, x)\equiv 1 && \text{for } x\uparrow\infty \text{ and } a>1,\\ &J(a, x)\uparrow J(1, x)\equiv 1 && \text{for } x\uparrow\infty \text{ and } a<1. \end{aligned} \tag{4.10}$$

Furthermore we derive from (4.2) for $1<a\leqq x$

$$1<J(a, x)<1/(1-q) \quad \text{where } q=(a-1)/x \tag{4.11}$$

and finally we obtain for $x=a$, a large, from the asymptotic expansion (6.5)

$$J(a, a)\approx\sqrt{a\pi/2}=1.253\sqrt{a} \tag{4.12}$$

which is the same formula as in case of $I(a, x)$ given in (3.11).

The algorithm (4.3) actually computes the sum

$$\begin{aligned} \tilde{J}_a=1&+(a-1)/x+(a-1)(a-2)/x^2+\cdots\\ &+(a-1)(a-2)\cdots(a-n+1)/x^{n-1}. \end{aligned} \tag{4.13}$$

For $a \leqq x$ and $n \leqq a+1$ the terms of this sum are all positive and monotonically decreasing. So the summation which is again done from the last to the first term by nested multiplications is performed in the ideal way as to the rounding error. The same is true if the summation is complete starting at the exact initial value J_b, $0 < b \leqq 1$. For $a+1 < n \leqq a+x$ the terms in (4.13) are still decreasing in magnitude but some of them now become negative. Nevertheless cancellation problems do not arise. First, the number of negative terms is bounded; for a relative error of $\varepsilon = 10^{-12}$, e.g., it is $\leqq 16$. And second, the sum $J(a, x)$ can only be slightly smaller than one when the open recursion provides a sufficient accuracy (a relative error of at least $\varepsilon = 10^{-3}$ or so); for $a \geqq 1$ we have $J(a, x) \geqq J(1, x) \equiv 1$, for $0.5 \leqq a \leqq 1$, $a \leqq x$ we have $J(a, x) \geqq J(0.5, 0.5) = 0.656$, and for $0 \leqq a < 0.5$, $a \leqq x$ the open recursion will not work unless x is at least larger than one and then we have $J(a, x) \geqq J(0, 1) = 0.596$.

We have already mentioned that we need not provide algorithms that compute both functions $I(a, x)$ and $J(a, x)$ for any pair of the arguments a, x (>0) since the two quantities are related by (2.2). Ideally, the smaller of the two quantities should be computed directly while the larger one could be determined by (2.2) without incurring cancellation. From (2.4) we obtain $I(a, x)/J(a, x) = \Pr\{X < x\}/\Pr\{X > x\}$ and this shows that the ratio $I(a, x)/J(a, x)$ is an increasing function of x for any $a > 0$. If we define x_m for given a such that $I(a, x_m) = J(a, x_m)$ then we obtain from the expansions (6.4) and (6.5) $x_m = a - \frac{1}{3} + 0(1)$ for $a \to \infty$. We suggest applying the following simple rule

$$\begin{aligned} &\text{compute } I(a, x) \quad \text{if } a > x, \\ &\text{compute } J(a, x) \quad \text{if } a \leqq x. \end{aligned} \tag{4.14}$$

Table 1 shows that this rule is adequate for the Poisson and chi-square distribution, where $a \geqq 0.5$.

TABLE 1

a	$I(a, a)$	$J(a, a)$	$I(a, a)/J(a, a)$
0.5	1.411	0.656	2.151
1.0	1.718	1.000	1.718
1.5	1.973	1.270	1.553
2.0	2.195	1.500	1.463
10	4.333	3.660	1.184
100	12.877	12.210	1.055
1 000	39.970	39.303	1.017
10 000	125.666	124.999	1.005

From (4.13) it is seen that the algorithm (4.3) is related to the asymptotic expansion 6.5.32 given by Abramowitz and Stegun (1964, p. 263). Another method of computing $J(a, x)$ for $a \leqq x$ can be obtained from the continued fraction 6.5.31 on the same page of the same book:

$$J(a, x) = \frac{1}{1+}\,\frac{1-a}{x+}\,\frac{1}{1+}\,\frac{2-a}{x+}\,\frac{2}{1+}\,\frac{3-a}{x+}\,\frac{3}{1+}\cdots. \tag{4.15}$$

This method can be even more efficient than the algorithm (4.3) in particular for large a and x, but as we have seen the algorithm (4.3) is very stable and we have a simple rule to determine the number of steps to guarantee a prescribed relative accuracy. I do not know whether the continued fraction (4.15) is also competitive in these two respects (see also Bhattacharjee (1970) and Moore (1982)).

5. Computation of $p(a, x)$. To reduce overflow and underflow problems in evaluating $p(a, x)$ as defined in (2.1) we compute as a first step the logarithm of $p(a, x)$. Note that an absolute error of ε in the logarithm $\ln(p(a, x))$ entails a relative error of ε in the antilogarithm $p(a, x)$, which means that the length of the mantissa of the logarithm determines the relative accuracy of the antilogarithm. Thus, if we wish to guarantee, e.g. six correct significant digits in the numeric value of $p(a, x)$, we have to compute $\ln(p(a, x))$ with six correct places after the decimal point.

The computation of $\ln(p(a, x))$ could be done simply by the formula

$$\ln(p(a, x)) = (a-1)\cdot\ln(x) - x - \ln(\Gamma(a)). \tag{5.1}$$

For large a and x, however, this method suffers under severe cancellation. As an example, for $x = a = 10^6$ we have $\ln(p(a, x)) \approx -7.8$ but $(a-1)\cdot\ln(x) \approx x + \ln(\Gamma(a)) \approx 1.38 \cdot 10^7$ and thus about seven decimal digits are lost when computing $p(a, x)$ by method (5.1).

The effect of cancellation can be attenuated by introducing a scaled version of the gamma function:

$$G(z) = \Gamma(z+1) \Big/ \left(\frac{z}{e}\right)^z \quad \text{for } z \geqq 0 \qquad (G(0) = 1). \tag{5.2}$$

According to Stirling's formula we have

$$\Gamma(z+1) \sim \left(\frac{z}{e}\right)^z \sqrt{2\pi z} \quad \text{for } z \to \infty \tag{5.3}$$

and thus $G(z) \sim \sqrt{2\pi z}$ for $z \to \infty$. The function G is increasing and its condition number $zG'(z)/G(z)$ is smaller than $1/2$ for all $z > 0$. The logarithmic scaled gamma function, $\ln(G(z))$, is well conditioned as well as the condition number being smaller than one in magnitude for all $z > 0$. To compute $\ln(G(z))$ for large z we can use the expansion 6.1.41 given by Abramowitz and Stegun (1964, p. 257) from which we obtain

$$\ln(G(z)) = \tfrac{1}{2}\ln(2\pi z) + 1/(12z) - 1/(360z^3) + \cdots. \tag{5.4}$$

When using seven terms of this expansion the absolute error will be less than 10^{-6} (or less than 10^{-12}) for $z \geqq 1.9$ ($z \geqq 5.7$ respectively).

As to the range of the function $p(a, x)$ we have

$$\begin{aligned} &\max_x p(a, x) = p(a, a-1) = 1/G(a-1) \leqq 1 \quad \text{if } a \geqq 1, \\ &p(a, x) \uparrow \infty \quad \text{for } x \downarrow 0 \quad \text{if } 0 < a < 1. \end{aligned} \tag{5.5}$$

Thus $p(a, x)$ is always smaller than 1 for $a > 1$ while it can be larger than 1 for $a < 1$. For large a and x the central limit theorem yields the approximation

$$p(a, x) \simeq \frac{1}{\sqrt{2\pi x}} \exp(-t^2/2) \quad \text{where } t = (a-1-x)/\sqrt{x} \tag{5.6}$$

and on a machine with $r_{\min} = 10^{-100}$ ($r_{\min}$ = smallest positive real machine number) the quantity $p(a, x)$ will underflow for $|t| \geqq 21$ or so. Note that both the expectation and the variance of the gamma distribution (2.3) are identical to the parameter a while for the Poisson distribution (2.7) both the expectation and the variance are identical to the parameter $\lambda = x$ so that $t = (a-1-x)/\sqrt{x} = (k-\lambda)/\sqrt{\lambda}$ is the standardized form of k in the sense of the Poisson distribution.

Using the logarithmic scaled gamma function we obtain from (5.1) for $a > 1$

$$\ln(p(a, x)) = (b-x) - b\cdot\ln(b/x) - \ln(G(b)) \quad \text{where } b = a-1\ (>0). \tag{5.7}$$

Both methods (5.1) and (5.7) are of the form $\ln(p) = A - B$ where $AB > 0$ and for large $b = a - 1$ and x we have

$$(5.8)\qquad \begin{aligned} A &= b \cdot \ln(x) && \text{in case (5.1),} \\ A &\simeq t \cdot \sqrt{x} \text{ with } t = (b-x)/\sqrt{x} && \text{in case (5.7).} \end{aligned}$$

Thus the effect of cancellation is much less severe with (5.7) particularly in the central part of the distribution where the standardized variable t is small. For $x = b$, where the quantity p attains its maximum as a function of x, there is no cancellation at all and $\ln(p)$ is given with full accuracy by (5.7).

One problem with method (5.7) should not be overlooked. Assuming that b and x are representable as machine numbers the quantities $b - x$ and $\ln(G(b))$ can be computed to machine accuracy as well, but this is not necessarily true for the term $b \cdot \ln(b/x)$. The condition number of the log-function $\ln(u)$ is $1/\ln(u)$ and so this function is ill-conditioned for $u \approx 1$, which is just the region where we are mostly interested in namely the central part of the distribution where $t = (b-x)/\sqrt{x}$ is small which means that $b/x \approx 1$. The problem can be overcome by introducing the shifted log-function

$$(5.9)\qquad \operatorname{lns}(v) = \ln(1+v) \quad \text{for } v \geqq 0$$

which has a condition number of one for $v = 0$, and we recommend computing the term $\ln(b/x)$ in (5.7) as follows:

$$(5.10)\qquad \ln(b/x) = \begin{cases} \operatorname{lns}\left(\dfrac{b-x}{x}\right) & \text{for } b \geqq x, \\ -\operatorname{lns}\left(\dfrac{x-b}{b}\right) & \text{for } b < x. \end{cases}$$

On a machine with $r_{\min} = 10^{-100}$ ($r_{\min}$ = smallest positive real machine number) and with 10 decimal digits we can achieve by (5.7) and (5.10) an accuracy of about five correct places after the decimal point so that $p(a, x)$ can be determined with about five correct significant digits for values of a and x as large as 10^8 even for the most extreme cases where $|t| = |a - 1 - x|/\sqrt{x} \simeq 20$ whereas method (5.1) cannot guarantee more than one correct significant digit.

6. Asymptotic expansions. With increasing a and x the iterative computation of $I(a, x)$ and $J(a, x)$ becomes more and more time consuming and the logarithmic computation of $p(a, x)$ less and less accurate due to cancellation. Both problems can be overcome by applying asymptotic expansions that refine the approximation by the normal distribution based on the central limit theorem.

Let X be a random variable having a gamma distribution with parameter a as defined in (2.3), and let us denote by $\varphi(t)$, $\Phi(t)$ and $\Phi^c(t)$ the density function, distribution function, and complementary distribution function, respectively, of the standard normal distribution

$$\varphi(t) = (1/\sqrt{2\pi}) \cdot \exp(-t^2/2),$$

$$\Phi(t) = \int_{-\infty}^{t} \varphi(u)\, du,$$

$$\Phi^c(t) = \int_{t}^{\infty} \varphi(u)\, du = 1 - \Phi(t),$$

$(-\infty < t < +\infty)$. From the results given by Esséen (1945) we obtain for $a \to \infty$:

$$\Pr\{X < x\} = \Phi(s) - \varphi(s)\left\{\frac{1}{3\sqrt{a}}(s^2-1) + \frac{1}{36a}(2s^5 - 11s^3 + 3s) + \frac{1}{1620a^{3/2}}(10s^8 - 145s^6 + 399s^4 - 69s^2 - 3) + O(a^{-2})\right\}, \tag{6.1}$$

$$p(a,x) = \frac{\varphi(s)}{\sqrt{a}}\left\{1 + \frac{1}{3\sqrt{a}}(s^3 - 3s) + \frac{1}{36a}(2s^6 - 21s^4 + 36s^2 - 3) + \frac{1}{1620a^{3/2}}(10s^9 - 225s^7 + 1269s^5 - 1665s^3 + 135s) + O(a^{-2})\right\}, \tag{6.2}$$

where $s = (x-a)/\sqrt{a}$ (refer to Knüsel (1981)). The expansion of $\Pr\{X > x\}$ is obtained in an obvious manner from (6.1) as $\Pr\{X > x\} = 1 - \Pr\{X < x\}$.

What accuracy can be achieved by means of the expansions given above? To ask a more precise question, let us take the first three terms of the expansion (6.1) as an approximation (= app) to the exact probability $\Pr\{X < x\}$ (= exa), and let us ask how large the parameter a has to be if we want the relative error of the approximation to be smaller than $\varepsilon = 10^{-6}$ for all values of x for which the exact probability can be represented as a positive machine number. We assume that $r_{\min} = 10^{-100}$ is the smallest positive machine number and we have $\Phi(-21.3) = r_{\min}$. From (6.1) we obtain for the relative error

$$\text{rel. err.} = \frac{\text{app} - \text{exa}}{\text{exa}} = \frac{\varphi(s)}{\Phi(s)}\left\{\frac{1}{1620a^{3/2}}(10s^8 - \cdots) + O(a^{-2})\right\} \tag{6.3}$$

and neglecting terms $O(a^{-2})$ this error is smaller than ε for all $|s| < 21.3$ if $a > 3 \cdot 10^{10}$. Note that $\varphi(s)/\Phi(s) \sim -s$ for $s \to -\infty$ as can be seen from (6.6) and (6.10) below. As to the accuracy of the expansion (6.2) the same considerations lead to pretty much the same results. Thus the expansions (6.1) and (6.2) will be useful only for rather large arguments a and x.

The quantity $p(a,x)$ can now be computed for virtually all values of a and x by means of the logarithmic procedure (5.10) in conjunction with the asymptotic expansion (6.2). But we are not yet content with the computation of the tail probabilities $\Pr\{X < x\}$ and $\Pr\{X > x\}$ as the iterative computation of $I(a,x)$ and $J(a,x)$ becomes too time consuming for values of a and x as large as 10^{10}; from (3.7) it is seen that the number of steps required to achieve a relative accuracy of $\varepsilon = 10^{-6}$ could become as large as $5.26 \cdot 10^5 \approx 500{,}000$. Now we can derive from (6.1) and (6.2) asymptotic expansions for $I(a,x)$ and $J(a,x)$, and it turns out that the convergence behaviour of these factors is much better than that of their parental expansions. We find for $x \to \infty$:

$$I(a,x) = \sqrt{x} \cdot a_1(t) - a_2(t) + a_3(t)/\sqrt{x} - a_4(t)/x + O(x^{-3/2}), \tag{6.4}$$

$$J(a,x) = \sqrt{x} \cdot a_1(\bar{t}) + a_2(\bar{t}) + a_3(\bar{t})/\sqrt{x} + a_4(\bar{t})/x + O(x^{-3/2}), \tag{6.5}$$

where

$$\begin{aligned} a_1(t) &= \Phi^c(t)/\varphi(t), \\ a_2(t) &= \frac{a_1(t)}{6}(t^3 - 3t) - \frac{1}{6}(t^2 - 4), \\ a_3(t) &= \frac{a_1(t)}{72}(t^6 - 9t^2 + 6) - \frac{1}{72}(t^5 - t^3 - 6t), \end{aligned} \tag{6.6}$$

$$a_4(t)=\frac{a_1(t)}{6480}(5t^9+45t^7-81t^5-315t^3+270t)$$
$$-\frac{1}{6480}(5t^8+40t^6-111t^4-174t^2+192),$$

with $t=(a-1-x)/\sqrt{x}$, $\bar{t}=(x-a+1)/\sqrt{x}$ $(=-t)$.

Remember that we have to compute only the smaller of the two quantities $I(a, x)$ and $J(a, x)$ directly and so the expansions above are of interest only for positive arguments t and $\bar{t}$.

If we take the first term of the expansion (6.4) as an approximation to $I(a, x)$,

$$I(a, x)\simeq\sqrt{x}\cdot a_1(t),$$

then the relative error is

$$\text{rel. err.}=\frac{\text{app}-\text{exa}}{\text{exa}}=\frac{a_2(t)}{\sqrt{x}\cdot a_1(t)}+O(1/x) \quad \text{for } x\to\infty. \tag{6.7}$$

Now a closer look at the coefficients $a_j(t)$ shows that

$$a_j(t)\sim 1/t^j \quad \text{for } t\to\infty \tag{6.8}$$

$(j=1,\cdots,4)$, and so $a_2(t)/a_1(t)\to 0$ for $t\to\infty$ (refer to (6.10) below). Thus we may expect that the relative error (6.7) will be uniformly small for all $t\geqq 0$ if x is sufficiently large! The same will be true when using two or more terms of the expansions (6.4) as an approximation to $I(a, x)$, and with the expansion for $J(a, x)$ things behave exactly the same way. Tables 2-5 confirm our considerations. For example it is seen that three terms of the expansion (6.4) give an approximation to $I(a, x)$ with a relative error of less than $0.253\cdot 10^{-4}$ for all $t\geqq 0$ if $x\geqq 100$.

So, the fact that $a_j(t)/a_1(t)\to 0$ for $t\to\infty$ $(j\geqq 2)$ is the significant improvement in the expansions of $I(a, x)$ and $J(a, x)$ as compared with the expansions (6.1) and (6.2) where the corresponding ratio tends to $\pm\infty$ for $|s|\to\infty$. And the reason why we have replaced the gamma variables a and $s=(x-a)/\sqrt{a}$ by the Poisson variables x $(=\lambda)$ and $t=(a-1-x)/\sqrt{x}=(k-\lambda)/\sqrt{\lambda}$ (refer to (2.8)) lies in the fact that the nice property (6.8) would not hold true with the gamma variables.

When computing the coefficients a_2, a_3, and a_4 by straightforward application of the formulas (6.6) we face again the problem of cancellation. If $t=20$, e.g., we have $a_4(t)=s_1-s_2$ where

$$s_1=\frac{a_1(t)}{6480}(5t^9+\cdots)\simeq 19.7\cdot 10^6,$$
$$s_2=\frac{1}{6480}(5t^8+\cdots)\simeq 19.7\cdot 10^6, \tag{6.9}$$

the correct difference being $a_4(t)=0.586\cdot 10^{-5}$, and so about thirteen decimal digits are cancelled. A more stable method of computing these coefficients for large t can be based upon the asymptotic expansions for $t\to\infty$

$$a_1(t)=1/t-1/t^3+3/t^5-3\cdot 5/t^7+3\cdot 5\cdot 7/t^9-\cdots,$$
$$a_2(t)=1/t^2-4/t^4+25/t^6-210/t^8+\cdots,$$
$$a_3(t)=1/t^3-11/t^5+130/t^7-1750/t^9+\cdots, \tag{6.10}$$
$$a_4(t)=1/t^4-26/t^6+546/t^8-11368/t^{10}+\cdots.$$

These expansions can be derived from 26.2.12 given by Abramowitz and Stegun (1964, p. 932).

TABLE 2

Asymptotic expansion of $I(a, x)$ *for* $x = 100$ (*refer to* (6.4)).

$$I(a,x) = \sqrt{x}\,a_1(t) - a_2(t) + a_3(t)/\sqrt{x} - a_4(t)/x + \cdots$$
$$= \Delta_0 + \Delta_1 + \Delta_2 + \Delta_3 + \cdots \quad \text{with } t = (a-1-x)/\sqrt{x}$$

Absolute errors: $d_1 = I(a,x) - \Delta_0$, $d_2 = I(a,x) - \Delta_0 - \Delta_1 = d_1 - \Delta_1$, etc.
Relative errors: $r_1 = d_1/I(a,x)$, $r_2 = d_2/I(a,x)$, etc.
The values in columns 5 to 11 are given in units of the sixth decimal place.

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
$a-1$	t	$I(a,x)$	Δ_0	d_1	Δ_1	d_2	Δ_2	d_3	Δ_3	d_4	r_1	r_2	r_3	r_4
100	0.00	11.877219	12.533141	−655,922	−666,667	10,745	10,444	301	296	5	−0.0552	0.0^3905	0.0^4253	0.0^6340
105	0.50	8.348448	8.763645	−415,197	−424,166	8,969	8,880	89	93	−4	−0.0497	0.00107	0.0^4108	-0.0^6359
110	1.00	6.281867	6.556795	−274,928	−281,440	6,512	6,512	0	3	−3	−0.0438	0.00104	0.0^7102	-0.0^6440
115	1.50	4.967770	5.158156	−190,386	−194,951	4,565	4,592	−27	−26	−1	−0.0383	0.0^3919	-0.0^5549	-0.0^6313
120	2.00	4.076437	4.213692	−137,255	−140,456	3,201	3,231	−30	−30	0	−0.0337	0.0^3785	-0.0^5748	-0.0^6177
125	2.50	3.440193	3.542651	−102,458	−104,734	2,276	2,302	−26	−26	0	−0.0298	0.0^3662	-0.0^5764	-0.0^7815
130	3.00	2.967114	3.045903	−78,789	−80,438	1,649	1,670	−21	−21	0	−0.0266	0.0^3556	-0.0^5701	-0.0^7253
135	3.50	2.603541	2.665678	−62,137	−63,355	1,218	1,235	−17	−16	−1	−0.0239	0.0^3468	-0.0^5612	0.0^8424
140	4.00	2.316455	2.366524	−50,069	−50,987	918	931	−13	−12	−1	−0.0216	0.0^3397	-0.0^5522	0.0^7180
145	4.50	2.084613	2.125706	−41,093	−41,799	706	715	−9	−9	0	−0.0197	0.0^3338	-0.0^5440	0.0^7231
150	5.00	1.893817	1.928081	−34,264	−34,815	551	558	−7	−7	0	−0.0181	0.0^3291	-0.0^5369	0.0^7237
155	5.50	1.734266	1.763230	−28,964	−29,401	437	442	−5	−5	0	−0.0167	0.0^3252	-0.0^5310	0.0^7222
160	6.00	1.598999	1.623777	−24,778	−25,129	351	356	−5	−4	−1	−0.0155	0.0^3220	-0.0^5261	0.0^7199
165	6.50	1.482950	1.504370	−21,420	−21,706	286	290	−4	−3	−1	−0.0144	0.0^3193	-0.0^5221	0.0^7174
170	7.00	1.382353	1.401042	−18,689	−18,925	236	239	−3	−3	0	−0.0135	0.0^3171	-0.0^5188	0.0^7151
175	7.50	1.294354	1.310794	−16,440	−16,636	196	199	−3	−2	−1	−0.0127	0.0^3152	-0.0^5161	0.0^7129
180	8.00	1.216752	1.231320	−14,568	−14,733	165	167	−2	−2	0	−0.0120	0.0^3136	-0.0^5138	0.0^7110
185	8.50	1.147827	1.160821	−12,994	−13,134	140	141	−1	−1	0	−0.0113	0.0^3122	-0.0^5119	0.0^8943
190	9.00	1.086214	1.097873	−11,659	−11,779	120	121	−1	−1	0	−0.0107	0.0^3110	-0.0^5104	0.0^8806
195	9.50	1.030819	1.041336	−10,517	−10,620	103	104	−1	−1	0	−0.0102	0.0^3100	-0.0^6906	0.0^8691
200	10.00	0.980752	0.990286	−9,534	−9,623	89	90	−1	−1	0	−0.00972	0.0^4911	-0.0^6795	0.0^8593
300	20.00	0.496296	0.498759	−2,463	−2,475	12	12	0	0	0	−0.00496	0.0^4244	-0.0^6118	0.0^9545
400	30.00	0.331862	0.332964	−1,102	−1,106	4	4	0	0	0	−0.00332	0.0^4110	-0.0^7360	0.0^9116
500	40.00	0.249222	0.249844	−622	−623	1	2	−1	0	−1	−0.00250	0.0^5621	-0.0^7154	$0.0^{10}377$

TABLE 3

Asymptotic expansion of $I(a, x)$ *for* $x=400$ (*refer to* (6.4)).

$$I(a, x)=\sqrt{x}a_1(t)-a_2(t)+a_3(t)/\sqrt{x}-a_4(t)/x+\cdots$$

$$=\Delta_0+\Delta_1+\Delta_2+\Delta_3+\cdots \quad \text{with } t=(a-1-x)/\sqrt{x}$$

Absolute errors: $d_1=I(a, x)-\Delta_0$, $d_2=I(a, x)-\Delta_0-\Delta_1=d_1-\Delta_1$, etc.
Relative errors: $r_1=d_1/I(a, x)$, $r_2=d_2/I(a, x)$, etc.
The values in columns 5 to 11 are given in units of the sixth decimal place.

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
$a-1$	t	$I(a, x)$	Δ_0	d_1	Δ_1	d_2	Δ_2	d_3	Δ_3	d_4	r_1	r_2	r_3	r_4
400	0.00	24.404913	25.066283	−661,370	−666,667	5,297	5,222	75	74	1	−0.0271	$0.0^{3}217$	$0.0^{5}306$	$0.0^{7}215$
410	0.50	17.107580	17.527289	−419,703	−424,166	4,463	4,440	23	23	0	−0.0245	$0.0^{3}261$	$0.0^{5}134$	$-0.0^{7}210$
420	1.00	12.835407	13.113591	−278,184	−281,440	3,256	3,256	0	1	−1	−0.0217	$0.0^{3}254$	$0.0^{7}285$	$-0.0^{7}266$
430	1.50	10.123651	10.316313	−192,662	−194,951	2,289	2,296	−7	−6	−1	−0.0190	$0.0^{3}226$	$-0.0^{6}654$	$-0.0^{7}192$
440	2.00	8.288536	8.427385	−138,849	−140,456	1,607	1,616	−9	−7	−2	−0.0168	$0.0^{3}194$	$-0.0^{6}909$	$-0.0^{7}110$
450	2.50	6.981713	7.085302	−103,589	−104,734	1,145	1,151	−6	−7	1	−0.0148	$0.0^{3}164$	$-0.0^{6}937$	$-0.0^{8}512$
460	3.00	6.012198	6.091806	−79,608	−80,438	830	835	−5	−5	0	−0.0132	$0.0^{3}138$	$-0.0^{6}864$	$-0.0^{8}164$
470	3.50	5.268613	5.331355	−62,742	−63,355	613	617	−4	−4	0	−0.0119	$0.0^{3}116$	$-0.0^{6}757$	$0.0^{9}212$
480	4.00	4.682523	4.733048	−50,525	−50,987	462	465	−3	−3	0	−0.0108	$0.0^{4}987$	$-0.0^{6}646$	$0.0^{8}108$
490	4.50	4.209968	4.251412	−41,444	−41,799	355	357	−2	−2	0	−0.00984	$0.0^{4}843$	$-0.0^{6}546$	$0.0^{8}141$
500	5.00	3.821624	3.856162	−34,538	−34,815	277	279	−2	−2	0	−0.00904	$0.0^{4}725$	$-0.0^{6}459$	$0.0^{8}146$
510	5.50	3.497278	3.526460	−29,182	−29,401	219	221	−2	−1	−1	−0.00834	$0.0^{4}628$	$-0.0^{6}386$	$0.0^{8}137$
520	6.00	3.222601	3.247553	−24,952	−25,129	177	178	−1	−1	0	−0.00774	$0.0^{4}549$	$-0.0^{6}326$	$0.0^{8}123$
530	6.50	2.987178	3.008740	−21,562	−21,706	144	145	−1	−1	0	−0.00722	$0.0^{4}482$	$-0.0^{6}276$	$0.0^{8}108$
540	7.00	2.783278	2.802084	−18,806	−18,925	119	119	0	−1	1	−0.00676	$0.0^{4}426$	$-0.0^{6}235$	$0.0^{9}935$
550	7.50	2.605050	2.621587	−16,537	−16,636	99	99	0	−1	1	−0.00635	$0.0^{4}379$	$-0.0^{6}201$	$0.0^{9}802$
560	8.00	2.447989	2.462639	−14,650	−14,733	83	83	0	0	0	−0.00598	$0.0^{4}339$	$-0.0^{6}172$	$0.0^{9}687$
570	8.50	2.308578	2.321641	−13,063	−13,134	71	71	0	0	0	−0.00566	$0.0^{4}305$	$-0.0^{6}149$	$0.0^{9}587$
580	9.00	2.184027	2.195746	−11,719	−11,779	60	60	0	0	0	−0.00537	$0.0^{4}275$	$-0.0^{6}130$	$0.0^{9}502$
590	9.50	2.072103	2.082672	−10,569	−10,620	51	52	−1	0	−1	−0.00510	$0.0^{4}250$	$-0.0^{6}113$	$0.0^{9}431$
600	10.00	1.970994	1.980572	−9,578	−9,623	45	45	0	0	0	−0.00486	$0.0^{4}228$	$-0.0^{7}992$	$0.0^{9}370$
800	20.00	0.995049	0.997519	−2,470	−2,475	5	6	−1	0	−1	−0.00248	$-0.0^{5}610$	$-0.0^{7}147$	$0.0^{10}340$
1,000	30.00	0.664824	0.665928	−1,104	−1,106	2	2	0	0	0	−0.00166	$0.0^{5}275$	$-0.0^{8}450$	$0.0^{11}725$
1,200	40.00	0.499065	0.499688	−623	−623	0	1	−1	0	−1	−0.00125	$0.0^{5}155$	$-0.0^{8}192$	$0.0^{11}236$

TABLE 4

Asymptotic expansion of $J(a, x)$ *for* $x=100$ (*refer to* (6.5)).

$$J(a,x)=\sqrt{x}a_1(\bar{t})+a_2(\bar{t})+a_3(\bar{t})/\sqrt{x}+a_4(\bar{t})/x+\cdots$$

$$=\Delta_0+\Delta_1+\Delta_2+\Delta_3+\cdots \quad \text{with } \bar{t}=(x-a+1)/\sqrt{x}$$

Absolute errors: $d_1=J(a,x)-\Delta_0$, $d_2=J(a,x)-\Delta_0-\Delta_1=d_1-\Delta_1$, etc.
Relative errors: $r_1=d_1/J(a,x)$, $r_2=d_2/J(a,x)$, etc.
The values in columns 5 to 11 are given in units of the sixth decimal place.

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
$a-1$	$\bar{t}$	$J(a,x)$	Δ_0	d_1	Δ_1	d_2	Δ_2	d_3	Δ_3	d_4	r_1	r_2	r_3	r_4
100	0.00	13.209961	12.533141	676,820	666,667	10,153	10,444	−291	−296	5	0.0512	0.0^3769	-0.0^4221	0.0^6348
95	0.50	9.196596	8.763645	432,951	424,166	8,785	8,880	−95	−93	−2	0.0471	0.0^3955	-0.0^4104	-0.0^6275
90	1.00	6.844742	6.556795	287,947	281,440	6,507	6,512	−5	−3	−2	0.0421	0.0^3951	-0.0^6795	-0.0^6382
85	1.50	5.357724	5.158156	199,568	194,951	4,617	4,592	25	26	−1	0.0372	0.0^3862	0.0^5451	-0.0^6288
80	2.00	4.357409	4.213692	143,717	140,456	3,261	3,231	30	30	0	0.0330	0.0^3748	0.0^5666	-0.0^6170
75	2.50	3.649713	3.542651	107,062	104,734	2,328	2,302	26	26	0	0.0293	0.0^3638	0.0^5705	-0.0^7826
70	3.00	3.128031	3.045903	82,128	80,438	1,690	1,670	20	21	−1	0.0263	0.0^3540	0.0^5660	-0.0^7286
65	3.50	2.730284	2.665678	64,606	63,355	1,251	1,235	16	16	0	0.0237	0.0^3458	0.0^5584	0.0^9907
60	4.00	2.418454	2.366524	51,930	50,987	943	931	12	12	0	0.0215	0.0^3390	0.0^5503	0.0^7152
55	4.50	2.168228	2.125706	42,522	41,799	723	715	8	9	−1	0.0196	0.0^3334	0.0^5427	0.0^7210
50	5.00	1.963461	1.928081	35,380	34,815	565	558	7	7	0	0.0180	0.0^3288	0.0^5361	0.0^7222
45	5.50	1.793079	1.763230	29,849	29,401	448	442	6	5	1	0.0166	0.0^3250	0.0^5304	0.0^7212
40	6.00	1.649266	1.623777	25,489	25,129	360	356	4	4	0	0.0155	0.0^3218	0.0^5257	0.0^7192
35	6.50	1.526369	1.504370	21,999	21,706	293	290	3	3	0	0.0144	0.0^3192	0.0^5218	0.0^7169
30	7.00	1.420208	1.401042	19,166	18,925	241	239	2	3	−1	0.0135	0.0^3170	0.0^5186	0.0^7147
25	7.50	1.327631	1.310794	16,837	16,636	201	199	2	2	0	0.0127	0.0^3151	0.0^5159	0.0^7127
20	8.00	1.246221	1.231320	14,901	14,733	168	167	1	2	−1	0.0120	0.0^3135	0.0^5137	0.0^7108
15	8.50	1.174098	1.160821	13,277	13,134	143	141	2	1	1	0.0113	0.0^3122	0.0^5119	0.0^8929
10	9.00	1.109774	1.097873	11,901	11,779	122	121	1	1	0	0.0107	0.0^3110	0.0^5103	0.0^8796
5	9.50	1.052061	1.041336	10,725	10,620	105	104	1	1	0	0.0102	0.0^4998	0.0^6901	0.0^8683
4	9.60	1.041224	1.030713	10,511	10,409	102	101	1	1	0	0.0101	0.0^4979	0.0^6878	0.0^8663
3	9.70	1.030606	1.020303	10,303	10,204	99	98	1	1	0	0.0100	0.0^4961	0.0^6855	0.0^8643
2	9.80	1.020200	1.010099	10,101	10,005	96	95	1	1	0	0.00990	0.0^4943	0.0^6833	0.0^8624
1	9.90	1.010000	1.000095	9,905	9,811	94	93	1	1	0	0.00981	0.0^4926	0.0^6812	0.0^8605
0	10.00	1.000000	0.990286	9,714	9,623	91	90	1	1	0	0.00971	0.0^4909	0.0^6791	0.0^8588

TABLE 5

Asymptotic expansion of $J(a, x)$ *for* $x=400$ (*refer to* (6.5)).

$$J(a,x)=\sqrt{x}a_1(\bar{t})+a_2(\bar{t})+a_3(\bar{t})/\sqrt{x}+a_4(\bar{t})/x+\cdots$$
$$=\Delta_0+\Delta_1+\Delta_2+\Delta_3+\cdots \quad \text{with } \bar{t}=(x-a+1)/\sqrt{x}$$

Absolute errors: $d_1=J(a,x)-\Delta_0$, $d_2=J(a,x)-\Delta_0-\Delta_1=d_1-\Delta_1$, etc.
Relative errors: $r_1=d_1/J(a,x)$, $r_2=d_2/J(a,x)$, etc.
The values in columns 5 to 11 are given in units of the sixth decimal place.

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
$a-1$	$\bar{t}$	$J(a,x)$	Δ_0	d_1	Δ_1	d_2	Δ_2	d_3	Δ_3	d_4	r_1	r_2	r_3	r_4
400	0.00	25.738098	25.066283	671,815	666,667	5,148	5,222	−74	−74	0	0.0261	0.0^3200	-0.0^5286	0.0^7218
390	0.50	17.955872	17.527289	428,583	424,166	4,417	4,440	−23	−23	0	0.0239	0.0^3246	-0.0^5131	-0.0^7184
380	1.00	13.398286	13.113591	284,695	281,440	3,255	3,256	−1	−1	0	0.0212	0.0^3243	-0.0^7775	-0.0^7247
370	1.50	10.513566	10.316313	197,253	194,951	2,302	2,296	6	6	0	0.0188	0.0^3219	0.0^6593	-0.0^7184
360	2.00	8.569464	8.427385	142,079	140,456	1,623	1,616	7	7	0	0.0166	0.0^3189	0.0^6858	-0.0^7108
350	2.50	7.191194	7.085302	105,892	104,734	1,158	1,151	7	7	0	0.0147	0.0^3161	0.0^6899	-0.0^8515
340	3.00	6.173083	6.091806	81,277	80,438	839	835	4	5	−1	0.0132	0.0^3136	0.0^6838	-0.0^8174
330	3.50	5.395332	5.331355	63,977	63,355	622	617	5	4	1	0.0119	0.0^3115	0.0^6739	0.0^9107
320	4.00	4.784503	4.733048	51,455	50,987	468	465	3	3	0	0.0108	0.0^4979	0.0^6635	0.0^9995
310	4.50	4.293570	4.251412	42,158	41,799	359	357	2	2	0	0.00982	0.0^4837	0.0^6538	0.0^8134
300	5.00	3.891258	3.856162	35,096	34,815	281	279	2	2	0	0.00902	0.0^4721	0.0^6454	0.0^8141
290	5.50	3.556083	3.526460	29,623	29,401	222	221	1	1	0	0.00833	0.0^4626	0.0^6382	0.0^8134
280	6.00	3.272862	3.246553	25,309	25,129	180	178	2	1	1	0.00773	0.0^4547	0.0^6323	0.0^8121
270	6.50	3.030592	3.008740	21,852	21,706	146	145	1	1	0	0.00721	0.0^4481	0.0^6274	0.0^8107
260	7.00	2.821128	2.802084	19,044	18,925	119	119	0	1	−1	0.00675	0.0^4425	0.0^6233	0.0^9925
250	7.50	2.638323	2.621587	16,736	16,636	100	99	1	1	0	0.00634	0.0^4378	0.0^6200	0.0^9798
240	8.00	2.477456	2.462639	14,817	14,733	84	83	1	0	1	0.00598	0.0^4339	0.0^6172	0.0^9681
230	8.50	2.334846	2.321641	13,205	13,134	71	71	0	0	0	0.00566	0.0^4304	0.0^6149	0.0^9583
220	9.00	2.207585	2.195746	11,839	11,779	60	60	0	0	0	0.00536	0.0^4275	0.0^6129	0.0^9499
210	9.50	2.093344	2.082672	10,672	10,620	52	52	0	0	0	0.00510	0.0^4250	0.0^6113	0.0^9428
200	10.00	1.990240	1.980572	9,668	9,623	45	45	0	0	0	0.00486	0.0^4227	0.0^7990	0.0^9368
100	15.00	1.331866	1.327485	4,381	4,368	13	14	−1	0	−1	0.00329	0.0^4106	0.0^7333	$0.0^{10}969$
50	17.50	1.142394	1.139161	3,233	3,224	9	9	0	0	0	0.00283	0.0^5791	0.0^7215	$0.0^{10}557$
10	19.50	1.025574	1.022965	2,609	2,603	6	7	−1	0	−1	0.00254	0.0^5641	0.0^7158	$0.0^{10}374$
4	19.80	1.010075	1.007544	2,531	2,525	6	6	0	0	0	0.00251	0.0^5622	0.0^7151	$0.0^{10}353$
3	19.85	1.007538	1.005019	2,519	2,513	6	6	0	0	0	0.00250	0.0^5619	0.0^7150	$0.0^{10}350$
2	19.90	1.005013	1.002506	2,507	2,500	7	6	1	0	1	0.00249	0.0^5616	0.0^7149	$0.0^{10}347$
1	19.95	1.002500	1.000006	2,494	2,488	6	6	0	0	0	0.00249	0.0^5613	0.0^7148	$0.0^{10}343$
0	20.00	1.000000	0.997519	2,481	2,475	6	6	0	0	0	0.00248	0.0^5610	0.0^7147	$0.0^{10}340$

REFERENCES

M. ABRAMOWITZ AND I. A. STEGUN, eds. (1964), *Handbook of Mathematical Functions*, Dover, New York.

G. P. BHATTACHARJEE (1970), *Algorithm* AS 32: *The incomplete gamma integral*, Appl. Stat. (JRSS, Series C), 19, pp. 285–287.

M. EL LOZY (1976), *Remark on algorithm* 299: *chi-squared integral*, ACM Trans. Math. Software, 2, pp. 393–395.

C. G. ESSÉEN (1945), *Fourier analysis of distribution functions, a mathematical study of the Laplace–Gaussian law*, Acta Math. 77, pp. 1–125.

W. GANDER (1977), *A machine independent algorithm for computing percentage points of the chi-square distribution*, J. Appl. Math. Phys. (ZAMP), 28, pp. 1133–1136.

I. D. HILL AND M. C. PIKE (1967), *Algorithm* 299: *chi-squared integral*, Comm. ACM, 10, pp. 243–244.

W. J. KENNEDY AND J. E. GENTLE (1980), *Statistical Computing*, Marcel Dekker, New York.

L. KNÜSEL (1981), *Computation of the Poisson and Chi-Square Distribution*, Forschungsbericht aus dem Institut für Statistik, Universität München.

J. R. MOORE (1982), *Derivatives of the incomplete gamma integral*, Appl. Stat. (JRSS, Series C), 31, pp. 330–335.

SIAM J. SCI. STAT. COMPUT.
Vol. 7, No. 3, July 1986

A GENERALIZATION OF THE METHOD OF CORRELATED SAMPLING FOR NUMERICAL INTEGRATION*

THOMAS E. FLICK† AND LEE K. JONES‡

Abstract. In correlated sampling an integrand f is approximated by a function g whose integral is known. However if the integral of g is unknown then ordinary correlated sampling cannot be used. Our generalization of correlated sampling addresses this situation: If the approximating function g is computationally simpler than f but both have unknown integrals, it is possible to estimate the integral of f with considerable time savings by optimal selective sampling of both f and g. One application, where the integral of f represents a probability of error, resulted in time savings by a factor of nearly 20.

Key words. correlated sampling, numerical integration, series expansion, error estimation, classification error, hypothesis testing

AMS (MOS) subject classifications. 65D30, 65U05, 62H12

1. Introduction. Statistical estimation is usually employed to integrate a function of many variables whenever the function is too complex to allow direct analytical methods. The simplest approach to such estimation is the ordinary Monte Carlo sampling scheme. But often this approach is too computationally burdensome to produce an estimate with a desired accuracy in a reasonable amount of time. This difficulty occurs especially when a large amount of computer time is necessary to evaluate the function for each sample generated. For example one may want to determine the expected value of a function expressed as a summation of thousands of terms. Another example occurs in multi-hypothesis testing where it is necessary to estimate classification error among hundreds of hypotheses. For these situations we propose a generalization of the method of correlated sampling. Correlated sampling (to be reviewed in § 2) can be much faster than the ordinary Monte Carlo approach. In correlated sampling an integrand f is approximated by a function g whose integral is known. However if the integral of g is unknown then ordinary correlated sampling cannot be used. Our generalization of correlated sampling addresses this situation: If the approximating function g is computationally simpler than f but both have unknown integrals, it is still possible to estimate the integral of f with considerable time savings by optimal selective sampling of both f and g. In fact experiments demonstrate time savings by an average factor of nearly 20 when compared to an ordinary Monte Carlo approach.

2. Correlated sampling. Let f be a "difficult" function to integrate on $\mathbb{R}^d$ with respect to some Borel probability measure μ. By generating N independent and identically distributed μ points $X_1, X_2, \cdots, X_N$ it is possible to estimate $\int f\,d\mu$ as

$$\hat{I} = \frac{1}{N}\sum_{i=1}^{N} f(X_i). \tag{1}$$

This is the ordinary Monte Carlo method. Now suppose there is a good approximation to f, namely g, whose integral $G = \int g\,d\mu$ is known. The principle of correlated sampling

* Received by the editors June 6, 1984, and in revised form May 15, 1985. This work was produced with the facilities of the Naval Research Laboratory and supported, in part, by the Office of Naval Research under Contract N00014-84-C2245.

† Naval Research Laboratory, Code 6520, Washington, DC 20375.

‡ Catholic University, Mathematics Department, Washington, DC 20064, and Unified Industries, Inc., Springfield, Virginia 22150.

(see [1], [2]) states that it is possible to obtain a variance reduced estimate of $\int f d\mu$ as

$$\hat{I}_g = \frac{1}{N}\sum_{i=1}^{N}[f(X_i) - g(X_i)] + G. \tag{2}$$

The variance of the methods is given by

$$\operatorname{var}\hat{I} = \frac{1}{N}\operatorname{var}(f), \tag{3}$$

$$\operatorname{var}\hat{I}_g = \frac{1}{N}\operatorname{var}(f-g). \tag{4}$$

Clearly the variance is reduced by a factor of $\operatorname{var}(f-g)/\operatorname{var}(f)$ by using $\hat{I}_g$ instead of $\hat{I}$. This may be appreciable if g closely approximates f.

3. Generalized correlated sampling. We generalize the above as follows: Suppose f is a difficult function (to evaluate as well as integrate) with an expected time complexity of $\bar{M}$ units required to evaluate at (including generating) a random (w.r.t. μ) X. Let g be a function which approximates f but requires only $\bar{R}$ units to evaluate in the expected case. Suppose evaluation of $f-g$ requires $\bar{P}$ expected units. Finally, assume S units are required in each case to generate a random X.

Now sample (independently μ) $f-g$, N_f times, and g, N_g times, and use the (unbiased) estimate of $\int f d\mu$

$$\hat{J}_g = \frac{1}{N_f}\sum_{i=1}^{N_f}[f(X_i) - g(X_i)] + \frac{1}{N_g}\sum_{i=N_f+1}^{N_f+N_g} g(X_i). \tag{5}$$

Then

$$\operatorname{var}(\hat{J}_g) = \frac{1}{N_f}\operatorname{var}(f-g) + \frac{1}{N_g}\operatorname{var}(g) \tag{6}$$

and the expected total time complexity is

$$\bar{T} = \bar{P}N_f + \bar{R}N_g. \tag{7}$$

For the moment assume N_f, N_g take positive real values. If we wish to minimize $\bar{T}$ for a fixed variance or minimize variance for fixed $\bar{T}$, elementary methods of Lagrange multipliers yield in either case

$$(N_g/N_f)^2 = \bar{P}\operatorname{var}(g)/(\bar{R}\operatorname{var}(f-g)). \tag{8}$$

This implies the relations

$$\operatorname{var}(\hat{J}_g) = (1/\bar{T})[(\bar{P}\operatorname{var}(f-g))^{1/2} + (\bar{R}\operatorname{var}(g))^{1/2}]^2, \tag{9}$$

$$\bar{T} = (1/\operatorname{var}(\hat{J}_g))[(\bar{P}\operatorname{var}(f-g))^{1/2} + (\bar{R}\operatorname{var}(g))^{1/2}]^2, \tag{10}$$

$$N_f = \bar{T}/[\bar{P} + (\bar{R}\bar{P}\operatorname{var}(g)/\operatorname{var}(f-g))^{1/2}], \tag{11}$$

$$N_g = \bar{T}/[\bar{R} + (\bar{R}\bar{P}\operatorname{var}(f-g)/\operatorname{var}(g))^{1/2}]. \tag{12}$$

Taking appropriate integer parts of N_f and N_g gives optimum solutions to the actual discrete sampling problems.

Note that the variance reduction could be considerable compared to straightforward Monte Carlo where, in particular,

$$\operatorname{var}(\hat{I}) = \bar{M}\operatorname{var}(f)/\bar{T}. \tag{13}$$

(Fix expected time complexity $\bar{T}$ for both procedures and compare (13) with (9).)

Of course, in many cases, the "good" approximation g is not known in advance. Hence a moderate amount of "preliminary" sampling is performed to "find" a g which minimizes

$$\sigma_g = (\bar{P} \operatorname{var}(f-g))^{1/2} + (\bar{R} \operatorname{var}(g))^{1/2} \tag{14}$$

and then more extensive sampling using g is done to achieve high accuracy rapidly.

Naturally the preliminary procedure introduces some inaccuracies since estimates of $\operatorname{var}(f-g)$ and $\operatorname{var}(g)$ are used instead of the true values. But the empirical results with our application to hypothesis testing appear very promising. Also, the estimates of σ_g at $g=g_1$ and those at $g=g_2$ are often highly correlated for g_1, g_2 close to the optimal g with respect to some suitable distance measure. So variance in estimating optimal g, N_f and N_g may be small for moderate numbers of samples. Finally, only an accurate estimate of $\operatorname{var}(f-g)/\operatorname{var}(g)$ is necessary to determine the optimal sampling ratio N_f/N_g.

4. Applications.

Example 1. *Integrating a series expansion.* Let

$$f = \sum_{i=1}^{M} a_i h_i(X), \tag{15}$$

$$g = \sum_{i=1}^{R} a_i h_i(X), \tag{16}$$

where the h_i's are computable in equal time. Let one unit be an evaluation of h_i plus one multiplication and one addition. Then we have

$$\bar{M} = M+S, \quad \bar{R} = R+S \quad \text{and} \quad \bar{P} = M-R+S. \tag{17}$$

Here our proposed procedure may be stated: Approximate the integral of the tail of the series (from $R+1$ to M) by using proportionately fewer samples than used for integrating the sum of the first R terms. R could be estimated by first minimizing (14) using preliminary samples.

Example 2. *Error estimation for multihypothesis tests.* When classifying an observation in $\mathbb{R}^d$ as belonging to one of Q classes by a maximum likelihood test (see [3], [5], [6]), one is primarily interested in the quantities of the form

$$\varepsilon_i^c = \int \Psi_{\gamma(\omega_i)} p(X|\omega_i)\, dX \tag{18}$$

where ε_i^c, called the complement of the type i error, is one minus the probability of misclassifying X given that class ω_i is present. Here $p(X|\omega_i)\, dX$ is the probability measure for class ω_i and the integrand, $f=\Psi_{\gamma(\omega_i)}$, is the indicator of the region of $\mathbb{R}^d$, $\gamma(\omega_i)$, where observations are assigned to class ω_i. If the $p(X|\omega_i)$ are all Gaussian with equal covariances then $\gamma(\omega_i)$ is the interior of a polytope with possibly as many as $Q-1$ sides. Because we must always check for the possibility of $Q-1$ sides, $\bar{M}$ for (18) is linear in Q. (Details are available upon request.) The integrand may be approximated by

$$g = \Psi_{\gamma_R(\omega_i)} \tag{19}$$

where $\gamma_R(\omega_i)$ is the region of assignment to ω_i if only R classes are considered. By choosing the R classes with the R closest mean vectors to that of class ω_i, substantial variance savings may be achieved. (Note $\bar{R}$ for g is linear in R.) In a classification

model for optical ship recognition with 255 classes, the authors were able to achieve a time savings of a factor of nearly 20. Time savings due to generalized correlated sampling (including the preliminary sampling) was 6.2. An additional factor of 3 was achieved by using polar coordinates (see [4]). This has allowed the authors to save weeks of computer time.

REFERENCES

[1] J. H. HALTON, *A retrospective and prospective survey of the Monte Carlo method*, SIAM Rev., 12 (1970), pp. 1-63.

[2] S. HABER, *Numerical evaluation of multiple integrals*, SIAM Rev., 12 (1970), pp. 481-526.

[3] L. K. JONES, *On finding non-randomized minimax tests for statistical decision problems with applications to signal detection*, Z. Warsch. Verw. Geb., 67 (1984), pp. 109-120.

[4] T. E. FLICK AND L. K. JONES, *Efficient error estimation for Gaussian classifiers*, Proc. IEEE 7th International Conference on Pattern Recognition, Montreal, Canada, July 1984.

[5] K. FUKUNAGA AND T. E. FLICK, *Density identification and risk estimation*, Purdue University Technical Report, TR-EE 82-83, Chapt. 4, Nov. 1982.

[6] K. FUKUNAGA, *Introduction to Statistical Pattern Recognition*, Academic Press, New York, 1972.

SIAM J. SCI. STAT. COMPUT.
Vol. 7, No. 3, July 1986

024

SOME COMPUTATIONAL ASPECTS OF A METHOD FOR RATIONAL APPROXIMATION*

LOTHAR REICHEL†

Abstract. Numerical aspects of a method for rational approximation of analytic functions on regions in the complex plane are considered. The approximation method divides the problem of computing a rational approximant into three subproblems. First, one chooses a space of rational functions, then one selects a basis for this space, and finally one determines an element of the space by interpolation. For approximation on regions with bounded simply connected complement, we discuss the choice of space and basis from a numerical point of view. We illustrate with computed examples.

Key words. rational approximation, interpolation, numerical conditioning

1. Introduction. We describe a numerical method for rational approximation of analytic functions on regions in the complex plane with bounded simply connected complement. In this method one first chooses a space Q_n of rational functions and a basis for this space. Then, a rational approximant is selected from Q_n by interpolation at points on the boundary of the region. We discuss numerical aspects of the choices of space and basis and show that these choices should depend on the shape of the region. The distribution of interpolation points should in turn depend on the rational space, and we further consider the computation of such points.

The questions of how to choose basis and interpolation points also arise in polynomial approximation and have, in this context, been discussed in Gautschi [2], Mason [4] and [12].

In this first section, we present some computed examples which illustrate the questions to be discussed in § 2 (choice of space) and § 3 (choice of basis). The examples do not show how to organize the computations most efficiently. We will return to that in § 4, which contains rational approximations of conformal mappings.

Let Ω be the closed exterior of the ellipse $E(a, b) := \{a \cos(t) + ib \sin(t), 0 \leqq t \leqq 2\pi\}$, $a \geqq b$. Denote by $\partial\Omega$ the boundary of Ω and let Ω_c be the complement of Ω w.r.t. the extended complex plane $\mathbb{C}^*$. Let $w = \psi(z)$ denote the conformal mapping from Ω to $|w| \geqq 1$ such that $\psi(\infty) = \infty$ and $\psi(a) = 1$. Then

$$\psi(z) = \frac{1}{a+b}\left(z + \sqrt{z^2 - (a^2 - b^2)}\right), \tag{1.1}$$

where the branch of the square root is chosen so that $\left|(a+b)^{-1}\left(z + \sqrt{z^2 - (a^2 - b^2)}\right)\right| \geqq 1$ for $z \in \Omega$. In a neighborhood of infinity $\psi(z)$ has the representation

$$\psi(z) = c^{-1}z + \sum_{k=0}^{\infty} d_k z^{-k} \tag{1.2}$$

with $c = \frac{1}{2}(a+b)$. We determine approximants of $\psi(z)$ by applying the approximation scheme to compute rational approximants of

$$f(z) := \psi(z) - c^{-1}z, \tag{1.3}$$

which is regular in Ω. We seek to approximate $f(z)$ by functions of the form

$$r_n(z) := \sum_{k=1}^{n} \alpha_k q_k(z) \tag{1.4}$$

* Received by the editors May 22, 1984, and in revised form March 1, 1985.
† Department of Mathematics, University of Kentucky, Lexington, Kentucky 40506.

where

$$q_k(z) := \begin{cases} 1, & k = 1, \\ s_k \prod_{j=1}^{k-1} (z - \zeta_j)^{-1}, & k = 2(1)n, \end{cases} \tag{1.5}$$

i.e. we choose the rational space $Q_n := \text{span}\{q_1, q_2, \cdots, q_n\}$. The points ζ_j are defined by choosing l not necessarily distinct points $\zeta_1^*, \zeta_2^*, \cdots, \zeta_l^*$ in Ω_c and letting

$$\zeta_{kl+m} := \zeta_m^*, \qquad m = 1(1)l, \quad k = 0, 1, 2, \cdots. \tag{1.6}$$

The s_k are scaling factors chosen so that $\|q_k\|_{\partial\Omega} = 1$, where $\| \ \|_{\partial\Omega}$ denotes the maximum norm on $\partial\Omega$,

$$\|q_k\|_{\partial\Omega} := \max_{z \in \partial\Omega} |q_k(z)|. \tag{1.7}$$

We determine the complex coefficients α_k of $r_n(z)$ by interpolating $f(z)$ at n nodes $z_j \in \partial\Omega$. This yields a linear system of equations for the α_j. For future reference, we write this linear system

$$A_n \boldsymbol{\alpha} = \mathbf{f}, \tag{1.8}$$

where $A_n := [a_{jk}]$, $a_{jk} := q_k(z_j)$, $k, j = 1(1)n$, and $\mathbf{f} := (f(z_1), f(z_2), \cdots, f(z_n))^T$, $\boldsymbol{\alpha} := (\alpha_1, \alpha_2, \cdots, \alpha_n)^T$. We obtain an approximation $\psi_n(z)$ of $\psi(z)$ from

$$\psi_n(z) := c^{-1}z + r_n(z). \tag{1.9}$$

Example 1.1. Let $a := \sqrt{2}$ and $b := 1/\sqrt{2}$. Choose the rational space

$$Q_n := \text{span}\{1, z^{-1}, z^{-2}, \cdots, z^{-n-1}\}, \tag{1.10}$$

and the basis $q_k(z) := s_k z^{-k+1}$, $s_k := 2^{(k-1)/2}$, $k = 1(1)n$. This corresponds to $\zeta_j = 0 \ \forall_j$ in (1.5). We will determine several approximations $r_n(z)$ of $f(z)$ by using different sets of interpolation points and different values of n.

First, choose the simply computable interpolation points

$$z_{j+1} := a \cos\left(2\pi \frac{j}{n}\right) + ib \sin\left(2\pi \frac{j}{n}\right), \quad j = 0(1)n - 1.$$

Let $n = 8$. Fig. 1.1 shows the unit circle and $\psi_8(\partial\Omega)$, the image of $\partial\Omega := E(\sqrt{2}, 1/\sqrt{2})$ under $\psi_8(z)$. $\psi_8(\partial\Omega)$ is not close to the unit circle, and increasing n increases the approximation error. The numerical behavior is analogous to the Runge phenomenon for polynomial approximation. As we will see below, the difficulties can be removed by using a different distribution of nodes. The dashes on Fig. 1.1 mark the images of the interpolation points under $\psi(z)$.

We next choose differently distributed interpolation nodes. To describe the distribution, we introduce the conformal mapping $w = \varphi(z)$ which maps $\Omega_c \cup \partial\Omega$ onto $|w| \geq 1$ so that $\varphi(0) = \infty$ and $\varphi(a) = 1$. The nodes z_j are defined by

$$\varphi(z_{j+1}) = \exp(2\pi ij/n), \qquad j = 0(1)n - 1. \tag{1.11}$$

This set of interpolation points is analogous to the Fejér points for polynomial interpolation. In Smirnov and Lebedev [13, Chap. 1], it is shown that if $f(z)$ is analytic on Ω, then the $r_n(z)$, computed by interpolating $f(z)$ at the n nodes defined by (1.11), converge to $f(z)$ in the norm (1.7) as $n \to \infty$.

Figure 1.2 shows $\partial\Omega$ and the nodes defined by (1.11) for $n = 24$, marked by dashes. In Fig. 1.3 we show the unit circle, $\psi_{24}(\partial\Omega)$, and the images under $\psi(z)$ of the nodes

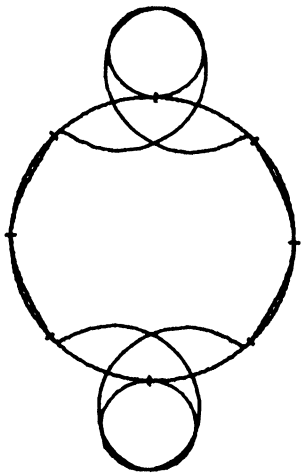


FIG. 1.1

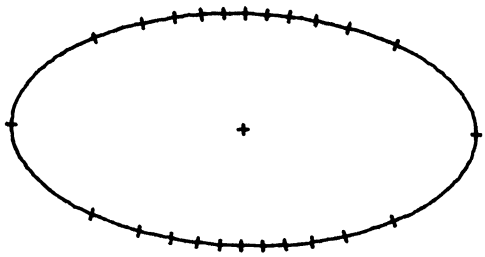


FIG. 1.2

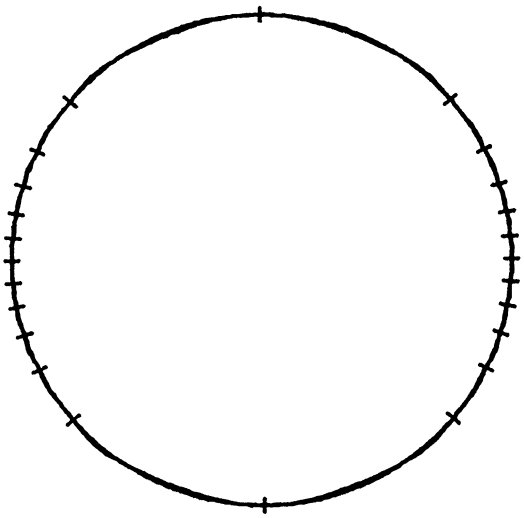


FIG. 1.3

in Fig. 1.2. The curve $\psi_{24}(\partial\Omega)$ is barely distinguishable from the unit circle. Figure 1.4 shows the approximation error for different n.

All computations in this paper were done on a DEC-10 computer in single precision arithmetic, i.e. with 8 significant digits. Figure 1.5 shows, for various n, the condition number $\text{cond } A_n := \|A_n\| \|A_n^{-1}\|$ of the $n \times n$ complex matrix A_n defined by (1.8). $\| \ \|$ denotes the matrix maximum norm for real matrices, and A_n is regarded as a $2n \times 2n$ real matrix. Figures 1.4–1.5 show that when computing with 8 significant digits, it is impossible to get a substantially smaller approximation error than for $n = 48$. This results from the poor numerical condition of A_n for large n. Already for $n = 48$, the ill-conditioning affects the accuracy of the computed approximation $\psi_{48}(z)$, as is seen by the rounded lower part of the graph of Fig. 1.4. In exact arithmetic, the convergence would be geometric and the graph would approximate a straight line.

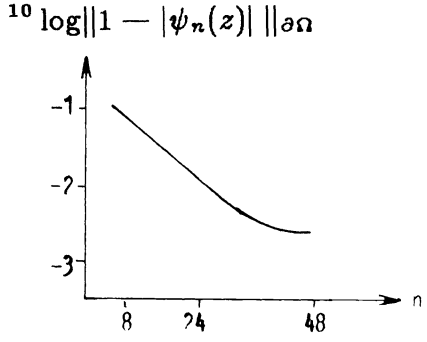


FIG. 1.4

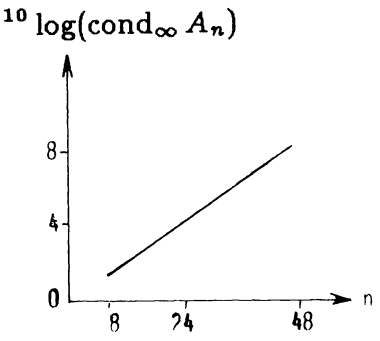


FIG. 1.5

The difficulty in computing high-accuracy approximations stems from the rapid increase of cond A_n with n combined with a fairly slow rate of convergence of $r_n(z)$ to $f(z)$ as n grows. The next example shows that the difficulties increase with the ratio between the lengths of the axes of the ellipse.

Example 1.2. Let $\partial\Omega := E(2, 1/2)$, $\zeta_j := 0 \ \forall_j$, and define the nodes z_j by (1.11). Figure 1.6 shows $\partial\Omega$ with 40 nodes. Figure 1.7 shows the unit circle, $\psi_{40}(\partial\Omega)$, and the images of the 40 nodes under $\psi(z)$. The lack of accuracy is obvious, and since cond $A_n = 1.1 \cdot 10^8$, the approximation cannot be improved significantly by increasing the degree n. On the other hand, an approximant of lower degree does not give higher accuracy either.

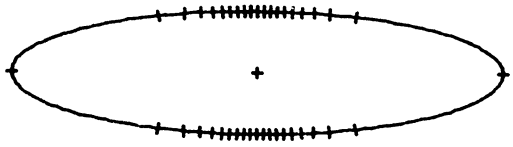


FIG. 1.6

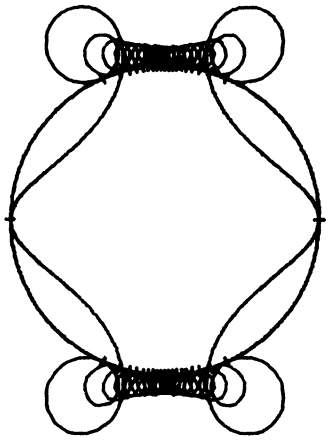


FIG. 1.7

The difficulties encountered in Examples 1.1–1.2 can to a large extent be overcome by choosing a rational space different from (1.10) to obtain a higher rate of convergence, or by selecting a different basis of the space (1.10) so that the linear system (1.8) is less ill-conditioned. In many cases it is possible to determine spaces Q_n which give a higher rate of convergence, and which have simple basis functions that give a fairly well-conditioned matrix A_n. This is illustrated in the following example.

Example 1.3. Let $\partial\Omega := E(\sqrt{2}, 1/\sqrt{2})$ and let $\zeta_1^*, \zeta_2^*, \zeta_3^*$ be the zeros of the degree-3 Chebyshev polynomial of the first kind scaled to the interval $[-\sqrt{3}/2, \sqrt{3}/2]$ between the foci of the ellipse. Define the ζ_j by (1.6) with $l = 3$. We generalize the distribution method (1.11) for interpolation points as follows. Let $w = \varphi_k(z)$ be the conformal mapping from $\Omega_c \cup \partial\Omega$ to $|w| \geqq 1$ such that $\varphi_k(\zeta_k^*) = \infty$ and $\varphi_k(\sqrt{2}) = 1$, $k = 1, 2, 3$. The nodes z_j are required to satisfy

$$\prod_{k=1}^{3} (\varphi_k(z_{j+1})^{1/3}) = \exp(2\pi ij/n), \qquad j = 0(1)n-1. \tag{1.12}$$

We let $0 \leqq \arg \varphi_k < 2\pi$ and determine the root of each $\varphi_k(z)$ before forming the product. This gives the correct branch of the root. For $n = 24$ we obtain Figs. 1.8–1.9. The former

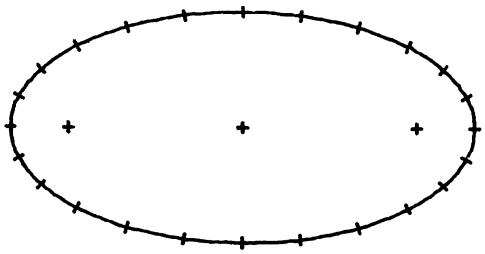


FIG. 1.8

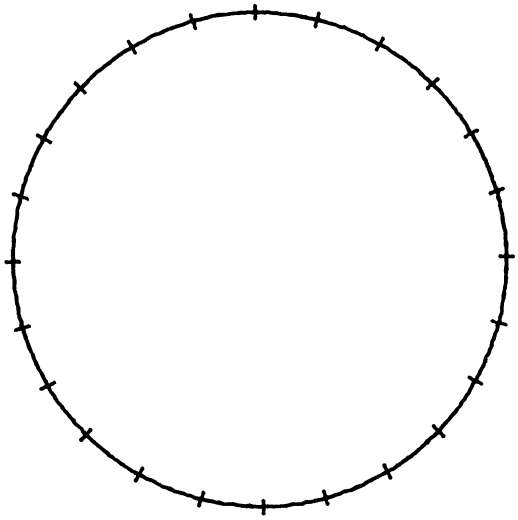


FIG. 1.9

shows $\partial\Omega$, the points ζ_k^* marked with crosses and the nodes z_j. Figure 1.9 shows the unit circle, $\psi_{24}(\partial\Omega)$ and the images of the z_j under $\psi(z)$. $\psi_{24}(\partial\Omega)$ is not distinguishable from the unit circle.

Figures 1.10–1.11 correspond to Figs. 1.4–1.5 and reveal that the rational space and basis of the present example yield a higher rate of convergence and a better conditioned matrix A_n than in Examples 1.1–1.2.

The ideas of this last example will now be developed in § 2.

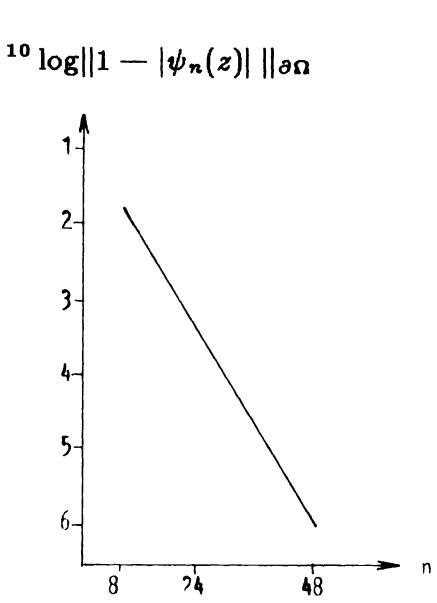


FIG. 1.10

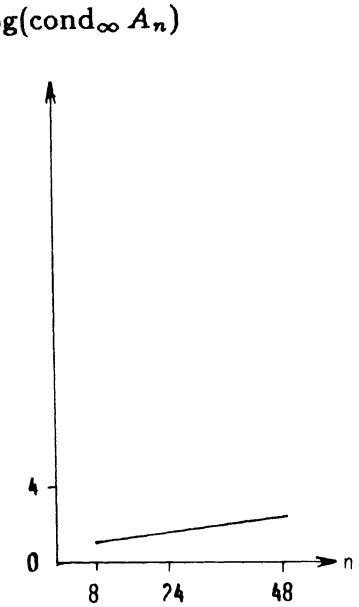


FIG. 1.11

2. Choice of rational space. Let Ω be a closed set in $\mathbb{C}$ with a bounded simply connected complement Ω_c and boundary $\partial\Omega$. We assume that $\partial\Omega$ is smooth since this simplifies the description of the approximation method, but this restriction can be relaxed. We consider approximation of functions $f(z)$ which are analytic in an open set $\hat{\Omega}$ which contains Ω. In this section we discuss the choice of rational space $Q_n := \operatorname{span}\{q_1, q_2, \cdots, q_n\}$, i.e. the choice of l and of the point set $\{\zeta_j^*\}_{j=1}^l$ in Ω_c. The ζ_j defining the $q_k(z)$ are obtained from (1.6). We also show how appropriate interpolation points can be computed.

Let $r_n(z)$ be the rational approximant to $f(z)$ obtained by interpolating $f(z)$ at the nodes $z_j \in \partial\Omega$, $j = 1(1)n$, according to the scheme (1.4)-(1.8). The approximation error can be expressed as,

$$f(z) - r_n(z) = \frac{1}{2\pi i} \int_\gamma \frac{W(z)}{W(\zeta)} \cdot \frac{f(\zeta)}{\zeta - z}\, d\zeta, \qquad z \in \Omega,$$

see Smirnov and Lebedev [13, Chap. 1], where

$$W(z) = \frac{\prod_{j=1}^n (z - z_j)}{\prod_{j=1}^{n-1} (z - \zeta_j)}.$$

γ is a contour in $\hat{\Omega}\backslash\Omega$ containing all points ζ_j^* in its interior. Fix the node z_1 on $\partial\Omega$, and define conformal mappings $w = \varphi_j(z)$ from Ω_c to $|w| > 1$ such that $\varphi_j(\zeta_j^*) = \infty$, $j = 1(1)l$. $\varphi_j(z)$ can be continued continuously to a bijective function on $\Omega_c \cup \partial\Omega$. We denote this extension also by $\varphi_j(z)$. $\varphi_j(z)$ is uniquely determined by the additional requirement $\varphi_j(z_1) = 1$. Introduce for $\rho \geqq 1$ the level curves

$$L(\rho) := \left\{ z : \prod_{j=1}^l |\varphi_j(z)| = \rho^l \right\}. \tag{2.1}$$

In particular, $L(1) = \partial\Omega$.

THEOREM 2.1. *For $n = 1, 2, 3, \cdots$ define the interpolation nodes z_j, $j = 2(1)n$ by*

$$\prod_{k=1}^l (\varphi_k(z_{j+1}))^{1/l} = \exp(2\pi i j/n), \qquad j = 1(1)n-1, \tag{2.2}$$

where we let $0 \leqq \arg \varphi_k < 2\pi$ and compute the root before multiplication. We then obtain the correct branch of the root. If $f(z)$ is analytic on and exterior to $L(\rho)$ for $\rho > 1$, then

$$\|f - r_n\|_{\partial\Omega} \leqq M\rho^{-n}, \qquad n = 1, 2, 3, \cdots$$

where M is a constant independent of n.

Proof. In Smirnov and Lebedev [13, Thm. 1, p. 61, and Thm. 2, p. 70], the theorem is proved for $l = 1$. The proof for $l > 1$ is analogous. □

The computation of nodes z_j from (2.2) is not very attractive. The next lemma shows a different way to determine the nodes.

LEMMA 2.1. *The system of integral equations*

$$\begin{aligned} d + \int_{\partial\Omega} \ln|z - \zeta| \sigma(\zeta) |d\zeta| &= \frac{1}{l} \sum_{j=1}^l \ln|z - \zeta_j^*|, \qquad z \in \partial\Omega, \\ \int_{\partial\Omega} \sigma(\zeta) |d\zeta| &= 1 \end{aligned} \tag{2.3}$$

for $\{\sigma, d\} \in L^2(\partial\Omega) \times \mathbb{R}$ has a unique solution $\{\sigma^*, d^*\}$. The interpolation nodes z_j,

$2 \leqq j \leqq n$, *defined by* (2.2) *satisfy*

$$\int_{z_1}^{z_j} \sigma^*(\zeta)|d\zeta| = \frac{j-1}{n}, \qquad j = 2(1)n, \tag{2.4}$$

where integration is performed along $\partial\Omega$ *in the direction which corresponds to increasing argument in* (2.2).

Proof. The lemma is a special case of [8, Thm. 3.1]. □

We turn next to the choice of space Q_n. We are interested in the shape of the level curves $L(\rho)$ as a function of ρ and of the location of the points ζ_j^*. The following examples are related to the approximation problems discussed in the introduction.

Example 2.1. Figure 2.1 shows the ellipse $E(\sqrt{2}, 1/\sqrt{2})$ and the level curves $L(1.13)$ and $L(1.37)$ defined by (2.1) with $l = 1$ and $\zeta^* = 0$. The curve $L(1.13)$ passes through the foci of the ellipse. The foci are marked with crosses and ζ_1^* with a dot. The curves $L(\rho)$ contract rapidly and become nearly circular as ρ increases. This property becomes more pronounced for flatter ellipses, see Fig. 2.2, which shows $\partial\Omega := E(2, \frac{1}{2})$, $L(1.13)$ and $L(1.37)$. The foci are marked with crosses. The contraction of the level curves gives slow convergence when approximating functions $f(z)$ with not all singular points near the origin. This was demonstrated in Examples 1.1–1.2. □

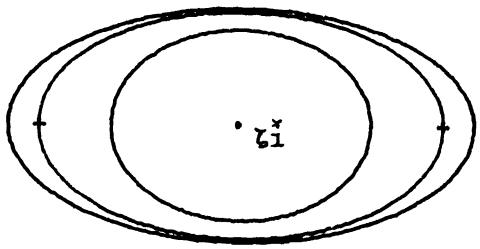


FIG. 2.1

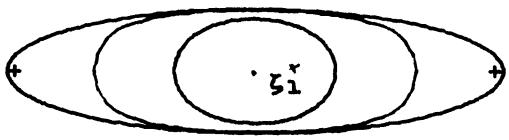


FIG. 2.2

If the locations of the singular points of $f(z)$ in Ω_c are known, we try to distribute points ζ_j^* so that a level curve $L(\rho)$ for some large ρ contains all the singular points of $f(z)$ in its interior. Sometimes the method of Papamichael et al. [6] can be used to determine the location of the singular points closest to $\partial\Omega$. This information could also be used to introduce basis functions which are singular where $f(z)$ is, see, e.g., Papamichael and Kokkinos [5]. We turn to the case when the locations of the singular points of $f(z)$ are not known. We then seek to determine a point set $\{\zeta_j^*\}_{j=1}^l$ in Ω_c such that

(1) the distance from a point $z \in \partial\Omega$ to $L(\rho)$, $\rho > 1$, is approximately the same for all points z on $\partial\Omega$;

(2) the points ζ_j^* are not very near $\partial\Omega$;

(3) there exists a simple fairly well-conditional basis of Q_n;

(4) the distribution of the ζ_j^* is simple to carry out.

The purpose of the first requirement is to make the rate of convergence $r_n(z) \to f(z)$, $n \to \infty$, $z \in \partial\Omega$, dependent on the distance δ between $\partial\Omega$ and the singular points of $f(z)$ closest to $\partial\Omega$, but fairly independent of at which points in Ω_c of distance δ from $\partial\Omega$ $f(z)$ is singular. This is reasonable when no a priori information on the location of the singular points of $f(z)$ is available. The second condition is connected to the wish that any level curve $L(\rho)$ containing all singular points of $f(z)$ in its interior also has all ζ_j^* in its interior. The reason for this is clear from Theorem 2.1. Instead of trying to find points ζ_j^* that satisfy conditions (1)-(3) directly, we solve a numerically simpler problem: we distribute points ζ_j^* in Ω_c so that for some constant c

$$h(z) := c \prod_{j=1}^{l} |z - \zeta_j^*|^{-1/l} \approx 1, \qquad z \in \partial\Omega. \tag{2.5}$$

We next discuss how this distribution is related to conditions (1)-(2), and then present a numerical method for determining points ζ_j^* that satisfy (2.5) and condition (2). In § 3 we show that condition (2.5) ensures the existence of a simple fairly well-conditioned basis of Q_n. Requirement (2.5) states that we wish to approximate $\partial\Omega$ by a lemniscate. This can always be done, see Hille [3, Chap. 16]. Equation (2.6) below shows that when $h(z)$ is nearly constant on $\partial\Omega$, then the level curves of $h(z)$ in Ω_c are of similar shape to the curves $L(\rho)$. It therefore suffices to study the level curves of $h(z)$.

LEMMA 2.2. *Let* $\partial\Omega := \{z: h(z) = 1\}$. *Then*

$$L(\rho) = \{z: h(z) = \rho\}. \tag{2.6}$$

Let $\psi(z)$ *be a conformal mapping from* Ω *to* $|w| \geqq 1$ *such that* $\psi(\infty) = \infty$. *Then for some constants* θ, $0 \leqq \theta \leqq 2\pi$, *and a suitable branch of the root,*

$$\psi(z) := e^{i\theta} c \prod_{j=1}^{l} (z - \zeta_j^*)^{1/l}. \tag{2.7}$$

Let $\psi^{-1}(w)$ *denote the inverse of* $\psi(z)$. *Then*

$$\frac{\partial h}{\partial n}(z) = \left| \frac{d\psi^{-1}(e^{it})}{dt} \right|^{-1}, \quad z \in \partial\Omega, \quad e^{it} = \psi(z), \tag{2.8}$$

where $\partial/\partial n$ *denotes the normal derivative directed into* Ω_c.

Proof. $\ln |\Pi_{j=1}^{l} \varphi_j(z)(h(z))^{-l}|$ is harmonic in Ω_c, vanishes on $\partial\Omega$, and therefore vanishes in Ω_c. This establishes (2.6). $\ln |h(z)\psi(z)|$ is harmonic in Ω and vanishes on $\partial\Omega$. Therefore $h(z) = |\psi(z)|^{-1}$. This establishes (2.7). (2.8) follows from (2.7) and the Cauchy-Riemann equations. □

For condition (1) to be satisfied, it is necessary that $\partial h/\partial n(z)$ does not vary too much on $\partial\Omega$. This is satisfied if $\partial\Omega$ is smooth and Ω_c is not pronouncedly nonconvex. When Ω_c is strongly nonconvex $|(d/dt)\psi^{-1}(e^{it})|^{-1}$ can vary considerably for $0 \leqq t < 2\pi$, even if $\partial\Omega$ is smooth, see [7] for an example. In this situation the approximation method of this paper is unsuitable.

Example 2.2. Let $\partial\Omega := E(\sqrt{2}, 1/\sqrt{2})$ and let $T_3(z)$ be the degree-3 Chebyshev polynomial of the first kind scaled to the interval $[-\sqrt{3/2}, \sqrt{3/2}]$ between the foci of the ellipse $E(\sqrt{2}, 1/\sqrt{2})$, and normalize $T_3(z)$ to have the highest order coefficient 0.210. Choose $h(z) := |T_3(z)|^{-1/3}$. Then $0.987 \leqq h(z) \leqq 1.012$ for $z \in \partial\Omega$. Figure 2.3 shows the level curves $\{z: h(z) = 1.13\}$ and $\{z: h(z) = 1.37\}$, and the roots ζ_j^*, $j = 1(1)3$ of $T_3(z)$ marked with dots. The foci of $E(\sqrt{2}, 1/\sqrt{2})$ are marked with crosses. The figure shows that the level curves of $h(z)$ are in better agreement with condition (1) than the level curves of Example 2.1.

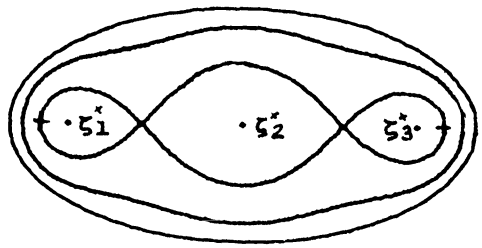


FIG. 2.3

We next turn to the question of how the proposed allocation of the ζ_j^* relates to condition (2). The corollary to the following lemma suggests that difficulties in finding points ζ_j^* such that both condition (2) and (2.5) are satisfied could arise if $\partial\Omega$ has a very small radius of curvature somewhere.

LEMMA 2.3 (Walsh [14, p. 252]). *Let S be an open set with a bounded simply-connected complement and boundary ∂S. Let $G(z)$ be the Green's function for the Laplace operator for S, i.e. $G(z)$ satisfies $\Delta G(z)=0$ for $z\in S\setminus\{\infty\}$, $G(z)=0$ on ∂S, $G(z)=\ln|z|+g(z)$ where $\Delta g(z)=0$ in S. Let K denote the smallest convex set which contains ∂S. If $\hat{z}$ is an arbitrary point in S, then the normal at $\hat{z}$ to the curve $\gamma:=\{z: G(z)=G(\hat{z})\}$ in the sense of decreasing $G(z)$ must intersect K. If $\tilde{z}$ is the point nearest to $\hat{z}$ of intersection of this normal with K, the radius of curvature at $\hat{z}$ of the curve γ is not less than $|\hat{z}-\tilde{z}|$.*

COROLLARY. *Assume that $\partial\Omega$ is analytic, and let $\hat{z}$ be an arbitrary point of $\partial\Omega$. Let v be a vector beginning at z, parallel to the normal of $\partial\Omega$ at z, directed into Ω_c, whose length equals the radius of curvature of $\partial\Omega$ at z. By analyticity of $\partial\Omega$, ψ can be continued analytically across $\partial\Omega$. Let $r_0<1$ be the smallest $r\geqq 0$ such that $|\psi(z)|$ has no singular points in the region $S:=\{z: |\psi(z)|>r_0\}$. Let ∂S be the boundary of S, and let K denote the convex hull of ∂S. Then the vector v touches (or crosses) K.*

Proof. The corollary follows from Lemma 2.3 by the identity $\psi(z)=ce^{G(z)+iH(z)}$, where c is a constant, $G(z)$ is the Green's function of the lemma, and $H(z)$ is its harmonic conjugate. We continue $\psi(z)$ analytically into Ω_c to the boundary ∂S, on which $|\psi(z)|$ has at least one singular point. For every $\varepsilon>0$, a Green's function $G_\varepsilon(z)$ with a pole at infinity can be defined on the region $S_\varepsilon:=\{z: |\psi(z)|\geqq r_0+\varepsilon\}$. We let K_ε be the convex hull of S_ε and note that $\partial\Omega$ is a level curve of $G_\varepsilon(z)$. By Lemma 2.3 the corollary follows for S replaced by S_ε for any $\varepsilon>0$. But v is independent of ε, which completes the proof. □

Example 2.3. Let $\partial\Omega$ be the ellipse $E(a, b)$ with $a\geqq b\geqq 0$. The radius of curvature of $\partial\Omega$ at $z=a$ is $\rho:=b^2/a$. The foci of $E(a, b)$ are $\pm\sqrt{a^2-b^2}$. Region S of the corollary is $\mathbb{C}\setminus[-\sqrt{a^2-b^2},\sqrt{a^2-b^2}]$. Let v be the normal vector to $\partial\Omega$ at $z=a$, directed into Ω_c and of length $|v|=\rho$. Let d denote the distance between $z=a$ and the closest focus. Introduce

$$\chi:=\frac{|v|}{d}=1+\sqrt{1-b^2/a^2}$$

giving $1\leqq\chi\leqq 2$, consistent with the corollary. For $a/b=2$, $\chi=1.9$.

We next describe a numerical method for allocating the ζ_j^*. The method consists of four steps:

(a) If $\partial\Omega$ is analytic and with nowhere a very small radius of curvature, let $\Gamma:=\partial\Omega$; otherwise let Γ be a curve near $\partial\Omega$ with these properties.

(b) Solve the system of integral equations

$$\alpha + \int_\Gamma \ln|z-\zeta|\sigma(\zeta)|d\zeta| = 0, \qquad z \in \Gamma,$$
$$\int_\Gamma \sigma(\zeta)|d\zeta| = 1 \tag{2.9}$$

for $\{\sigma, \alpha\} \in L^2(\Gamma) \times \mathbb{R}$. (2.9) has a unique solution $\{\sigma^*, \alpha^*\}$; see [9], where also a numerical solution method is described. Denote by $\psi_\Gamma(z)$ the conformal mapping from the exterior of Γ to $|w| > 1$ such that $\psi_\Gamma(\infty) = \infty$ and $\psi_\Gamma(\hat{z}_1) = 1$, where $\hat{z}_1$ is a point on $\partial\Omega$. By [9]

$$\psi_\Gamma(z) = \exp 2\pi i \int_{\hat{z}_1}^{z} \sigma^*(\zeta)|d\zeta|, \qquad z \in \Gamma, \tag{2.10}$$

where integration is carried out along Γ in the positive sense. $e^{|\alpha^*|}$ is the capacity of Γ.

(c) Continue $\psi_\Gamma(z)$ analytically into Ω_e. We do this by solving an initial value problem for the Cauchy-Riemann equations by a method described in [10]. The method generates approximate level curves of $|\psi_\Gamma(z)|$ for equidistant and decreasing levels, as well as approximate stream lines starting at the images of the roots of unity under $\psi_\Gamma^{-1}(z)$. Each step of the method generates a new approximate level curve, and we continue the generation of level curves until, from the computed level curves, it appears likely that we have reached a singularity of $\psi_\Gamma(z)$, see Figs. 2.4 and 2.6. The generation of level curves is easily monitored at a graphics terminal.

(d) Allocate l points ζ_j^*, $j = 1(1)l$, on the innermost generated level curve, so that they are images under $\psi_\Gamma^{-1}(w)$ of equidistant points on a circle $|w| = d < 1$.

The next lemma shows that for l sufficiently large, the set $\{\zeta_j^*\}_{j=1}^l$ determined as described above will make the product $\Pi_{i=1}^l |z-\zeta_j^*|^{1/l}$ nearly constant on Γ.

LEMMA 2.4. *Let the set $\{\zeta_j^*\}_{i=1}^l$ be computed by steps* (a)-(d). *Then for some constant r, $0 \leqq r < 1$, decreasing when the distance between ζ_j^* and $\partial\Omega$ increases, we have uniformly for $z \in \Gamma$*

$$\prod_{j=1}^{l} (z-\zeta_j^*) = (c\psi_\Gamma(z))^l(1+0(r^l)), \qquad l \to \infty,$$

where c is a complex constant whose magnitude is the capacity of Γ.

Proof. The lemma can, for example, be shown by the methods in [11]. □

To measure how much the product $\prod_1^l |z-\zeta_j^*|^{1/l}$ differs from being constant on $\partial\Omega$, we introduce the level curves

$$\mathscr{L}(c) := \left\{ z: \prod_{j=1}^{l} |z-\zeta_j^*|^{1/l} = c \right\}. \tag{2.11}$$

All computed examples allowed $\Gamma := \partial\Omega$.

Example 2.4. Let $\partial\Omega$ be the superellipse $\{x+iy: x^4+y^4 = 1, x, y \in \mathbb{R}\}$. Figure 2.4 shows $\partial\Omega$ and four level curves as well as stream lines of $\psi(z)$ in the interior of $\partial\Omega$ computed by steps (a)-(d). The figure indicates that $\psi(z)$ is likely to be singular at the four points $\pm 0.71 \pm 0.71i$. It is easy to show that $\psi^{-1}(1/\psi(z))$ is singular at the four points $\pm 1/\sqrt{2} \pm i/\sqrt{2}$, see [10]. We allocate the points ζ_j^* on the innermost level curve drawn. Denote this level curve by $|\psi(z)| = r_0$. Figure 2.5 shows the points $\zeta_j^* := \psi^{-1}(r_0 \exp(2\pi i(j-1)/n))$, $j = 1(1)8$ for $n = 8$. For this choice of ζ_j^* the largest c such that the curve $\mathscr{L}(c)$, defined by (2.11), is in Ω is $c = 1.9$. The smallest value of c such

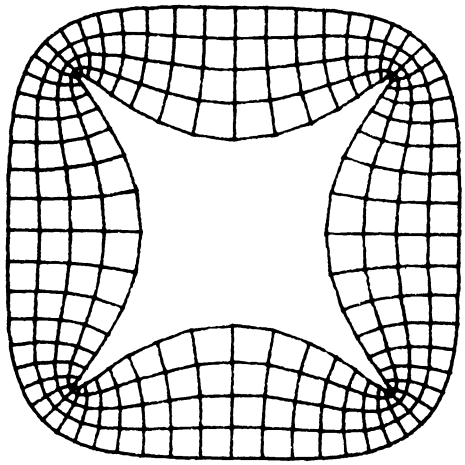


FIG. 2.4

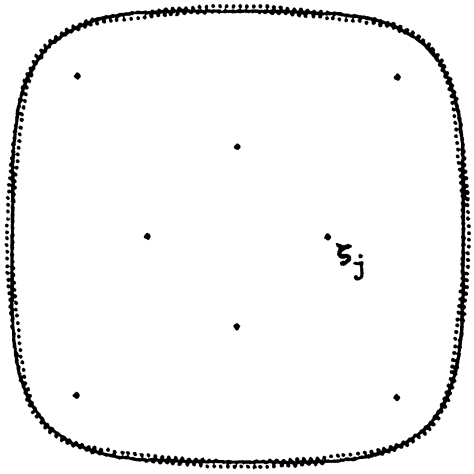


FIG. 2.5

that $\mathscr{L}(c)$ is in $\Omega_c \cup \partial\Omega$ is $c = 1.11$. The closeness of these c-values shows that $h(z)$ is nearly constant on $\partial\Omega$. This will also be of interest in § 3. The curves $\mathscr{L}(1.09)$ and $\mathscr{L}(1.11)$ are drawn with dots and are seen to be close to $\partial\Omega$.

Example 2.5. Let $\partial\Omega := E(\sqrt{2}, 1/\sqrt{2})$. Figure 2.6 shows $\partial\Omega$ and five computed level curves as well as stream lines of $\psi(z)$ in Ω_c. The level curves are confocal ellipses.

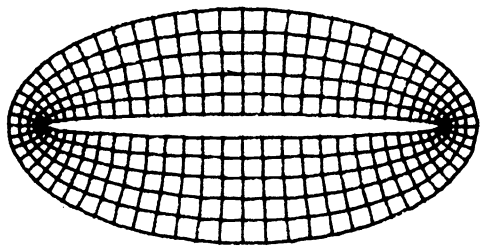


FIG. 2.6

The level curves and stream lines suggest that $\psi(z)$ is singular at the foci of $\partial\Omega$, in agreement with (1.1). Let $|\psi(z)| = r_0$ be the innermost level curve drawn, and allocate six points $\zeta_j := \psi^{-1}(r_0\, e^{2\pi i(j-1)/6}\, e^{i\theta})$, $j = 1(1)6$, where θ is chosen such that ζ_1 and ζ_4 lie on the y-axis. Forming the mean values of close points, $\zeta_1^* := \frac{1}{2}(\zeta_1 + \zeta_4)$, $\zeta_2^* := \frac{1}{2}(\zeta_2 + \zeta_3)$, $\zeta_3^* := \frac{1}{2}(\zeta_5 + \zeta_6)$ yields approximations of the zeros of the third degree Chebyshev polynomial of the first kind for the interval between the foci. With ζ_j^*, $j = 1(1)3$, being the roots of this Chebyshev polynomial, we obtain Fig. 2.7. The dotted level curve $\mathscr{L}(1.105)$ is the smallest curve enclosing $\partial\Omega$, and $\mathscr{L}(1.102)$, also dotted, is the largest curve interior to $\Omega_c \cup \partial\Omega$.

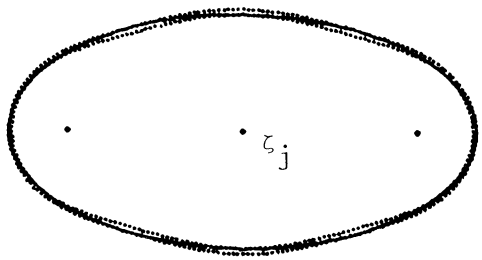


FIG. 2.7

Next we turn to ellipses $\partial\Omega := E(2, \frac{1}{2})$, and allocate the ζ_j^* analogously as above: ζ_j^*, $j = 1(1)3$, are the roots of a Chebyshev polynomial $T_3(z)$ for the interval between the foci of $\partial\Omega$. Figure 2.8 shows $\partial\Omega$, the ζ_j^* and level curves $\mathscr{L}(c)$. Notice that the largest locus $\mathscr{L}(c)$ interior to $\Omega_c \cup \partial\Omega$ is not connected. In particular it does not contain the interval between the foci in its interior. By the previous discussion, the subspace defined by these ζ_j^* is not suitable for approximating functions which are nonanalytic on that interval. If we instead use the roots of the Chebyshev polynomial of the fifth degree, we are in a better position, as seen in Fig. 2.9. Then $\mathscr{L}(1.23)$, the largest level curve interior to or on $\Omega_c \cup \partial\Omega$, is connected and contains the interval between the foci in its interior. The smallest level curve enclosing $\partial\Omega$ is $\mathscr{L}(1.27)$.

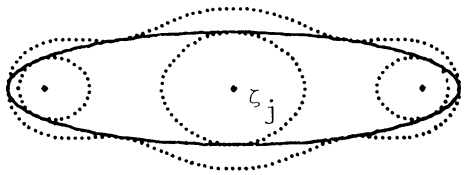


FIG. 2.8

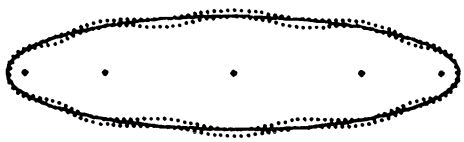


FIG. 2.9

3. Choice of basis. We will show that when the ζ_j^* are allocated as suggested in § 2, the basis (1.5) of Q_n is fairly well-conditioned. We will also consider the numerical condition of a basis which is slightly faster to evaluate. The reason for using a well-conditioned basis in computation is the wish to obtain a linear system (1.8) with

a fairly well-conditioned matrix A_n. However, cond A_n also depends on the allocation of the nodes z_j. Even if we use a basis that is well-conditioned in a sense defined below, cond $A_n \to \infty$ as $z_1 \to z_2$. Conversely, for some ill-conditioned bases one may at least, for fixed n, be able to find nodes z_j such that cond A_n is small. Nevertheless, numerical experiments suggest that with the relation (2.3)-(2.4) between nodes z_j and poles ζ_j, there is a strong connection between the condition number for the basis and cond A_n, in that both condition numbers are large or small simultaneously.

The following definitions are analogous to those introduced by Gautschi [2] in his investigation of polynomial bases on intervals of the real axis. Let $\mathbf{a} := (a_1, a_2, \cdots, a_n) \in \mathbb{R}^n$ and define the mapping $F_n : \mathbb{R}^n \to Q_n$ and its inverse by

$$(F_n\mathbf{a})(z) := r_n(z) := \sum_{k=1}^{n} a_k q_k(z)$$

$$F_n^{-1} r_n := \mathbf{a}.$$

Equip $\mathbb{R}^n$ with the norm $\|a\|_\infty := \max_{1 \leqq k \leqq n} |a_k|$. In Q_n we use norm (1.7). We obtain the induced operator norms

$$\|F_n\| := \max_{\|a\|_\infty = 1} \left\| \sum_{k=1}^{n} a_k q_k(z) \right\|_{\partial\Omega},$$

$$\|F_n^{-1}\| := \max_{\|r_n\|_{\partial\Omega} = 1} \|\mathbf{a}\|_\infty.$$

The condition number of the basis is defined as cond $F_n := \|F_n\| \, \|F_n^{-1}\|$.

Example 3.1. Let $\partial\Omega := \{z : |z| = r\}$ and consider the basis $q_n(z) := (r/z)^{k-1}$, $k = 1(1)n$. Then $\|q_k\|_{\partial\Omega} = 1 \; \forall k$ and $\|F_n\| = n$. Since $(z/r)^k$ are Chebyshev polynomials for $\partial\Omega$, see Davis [1, p. 146], we obtain

$$\|F_n^{-1}\|^{-1} = \min_{\|\mathbf{a}\|_\infty = 1} \left\| \sum_{k=1}^{n} a_k q_k(z) \right\|_{\partial\Omega} = \min_{\|\mathbf{a}\|_\infty = 1} \left\| \sum_{k=1}^{n} a_k (z/r)^{n-k} \right\|_{\partial\Omega} = 1.$$

Hence cond $F_n = n$, which shows that this basis is fairly well-conditioned on $\partial\Omega$.

Example 3.2. Let $\partial\Omega$ be the ellipse $E(c, d)$, $c > d$, reflected in the circle $|z| = c$, i.e. $\partial\Omega := \{z := c^2(c \cos(t) + id \sin(t))^{-1}, 0 \leqq t < 2\pi\}$. We let Q_n be the same as in Example 3.1, and choose the basis $q_k(z) := (c/z)^{k-1}$, $k = 1(1)n$. The substitution $w := c^2/z$ transforms F_n into $(F_n\mathbf{a})(w) = \sum_{k=0}^{n-1} a_k (w/c)^k$, $w \in E(c, d)$. The determination of cond F_n is now a problem of determining the condition of a scaled power basis on $E(c, d)$. This problem is treated in [12], where we showed that for $c > d$, cond F_n grows exponentially with n and the exponential growth rate is given by cond $F_n \sim ((1+\sqrt{2-\alpha^2})/(1+\alpha))^n$, $\alpha := d/c$.

These examples demonstrate that the choice of a well-conditioned basis must depend on the shape of $\partial\Omega$. We turn now to the investigation of basis (1.5), and to begin with, we assume that $\partial\Omega = \mathscr{L}(c)$ for some $c > 0$. Then $|q_{kl+j}(z)| = |q_j(z)|$, $z \in \partial\Omega$, $1 \leqq j \leqq l$, $k \geqq 0$. Therefore there is a constant $M > 0$ such that

$$|q_k(z)| \geqq M \quad \text{on } \partial\Omega, \; k \geqq 0. \tag{3.1}$$

We can bound $\|F_n^{-1}\|$ by observing that $a_{n-k} s_{n-k}$, $k = 0(1)n-1$, can be expressed as divided differences of

$$r_n(z) \prod_{j=1}^{n-1} (z - \zeta_j) = \sum_{k=1}^{n-1} a_k s_k \prod_{j=k}^{n-1} (z - \zeta_j) + a_n s_n,$$

where $s_1 := 1$. The divided differences can be written as complex integrals over $\partial\Omega$, see Davis [1, pp. 67-69],

$$a_n s_n = \frac{1}{2\pi i}\int_{\partial\Omega} \frac{r_n(z)\prod_{j=1}^{n-1}(z-\zeta_j)}{(z-\zeta_{n-1})}\,dz, \tag{3.2a}$$

$$a_{n-k}s_{n-k} = \frac{1}{2\pi i}\int_{\partial\Omega} \frac{r_n(z)\prod_{j=1}^{n-1}(z-\zeta_j)}{\prod_{j=n-k}^{n-1}(z-\zeta_j)(z-\zeta_{n-k-1})}\,dz, \qquad k=1(1)n-1, \tag{3.2b}$$

where we define $\zeta_0 := \zeta_n$. From (3.2), we obtain

$$|a_{n-k}| \leqq \left\| \frac{1}{q_{n-k}(z)(z-\zeta_{n-k-1})} \right\|_{\partial\Omega} \frac{1}{2\pi}\int_{\partial\Omega} |dz|, \qquad k=0(1)n-1, \tag{3.3}$$

and by combining (3.3) and (3.1), we find that the a_j are uniformly bounded for all j. Hence for some constant $\hat{M}$, $\|F_n^{-1}\| \leqq \hat{M}\ \forall n$. Since $\|F_n\| \leqq n$, we obtain

$$\operatorname{cond} F_n \leqq \hat{M}n.$$

We next remove the requirement $\partial\Omega = \mathscr{L}(c)$. A new bound corresponding to (3.1) is needed. Let $k = m_0 l + m_1$, $1 \leqq m_1 \leqq l$. Then

$$\begin{aligned} \left\| \frac{1}{q_{k+1}(z)} \right\|_{\partial\Omega} &= \max_{z\in\partial\Omega} \prod_{j=1}^{k} |z-\zeta_j| \Big/ \min_{z\in\partial\Omega} \prod_{j=1}^{k} |z-\zeta_j| \\ &\leqq M \max_{z\in\partial\Omega} \prod_{j=1}^{l} |z-\zeta_j|^{m_0} \Big/ \min_{z\in\partial\Omega} \prod_{j=1}^{l} |z-\zeta_j|^{m_0}, \end{aligned} \tag{3.4}$$

where M is a constant independent of k. In order to bound the quotient on the right-hand side of (3.4), we introduce constants c_1 and c_2 defined by

$$c_1 := \text{smallest } c \text{ such that } \mathscr{L}(c) \subset \Omega,$$

$$c_2 := \text{greatest } c \text{ such that } \mathscr{L}(c) \subset \Omega_c \cup \partial\Omega.$$

Then $c_1 \geqq c_2$, and by (3.4),

$$\left\| \frac{1}{q_{k+1}(z)} \right\|_{\partial\Omega} \leqq M(c_1/c_2)^{lm_0}. \tag{3.5}$$

Substituting (3.5) into (3.3) gives

$$\operatorname{cond} F_n \leqq \tilde{M}n(c_1/c_2)^{ln_0},$$

where $\tilde{M}$ is a constant independent of n, and n_0 is the integer part of n/l. In § 2, we described a method for allocating the ζ_j^* so that c_1/c_2 is close to 1, and the basis (1.5) is then quite well-conditioned for moderate n.

We conclude this section with a remark on a basis simpler than (1.5). For notational convenience, assume that the ζ_j^*, $j=1(1)l$, are distinct and that $n = \hat{n}l+1$ for some integer $\hat{n} > 0$. Then

$$\begin{aligned} &\hat{q}_1(z) := 1, \\ &\hat{q}_{kl+j+1}(z) := s_{kl+j}(z-\zeta_j^*)^{-k-1}, \quad j=1(1)l, \quad k=0(1)\hat{n}-1, \end{aligned} \tag{3.6}$$

is a basis for Q_n. Calculations with $l := 2$, $\zeta_1^* := \delta > 0$, $\zeta_2^* := -\delta$ and $\partial\Omega := \{z: |z^2-\delta^2| = c^2\}$ show that this basis becomes severely ill-conditioned as $\delta \to 0$ or $c \to \infty$. This is reflected in larger condition numbers of the linear system (1.8) for basis (3.6) than for basis (1.5). Basis (3.6) is therefore not recommended.

4. Applications to conformal mapping. We can apply the approximation method to compute rational approximants of some exterior conformal mappings $\psi: \Omega \to |w| \geqq 1$ such that $\psi(\infty) = \infty$ and $\psi(z_1) = 1$, where $z_1 \in \partial\Omega$. This gives approximants of a simple form which can be evaluated rapidly by nested multiplication. In the examples of this section we carried out the following computations in order.

(1) Solve (2.9) with $\Gamma := \partial\Omega$ for $\{\sigma^*, \alpha^*\}$ to obtain the restriction of $\psi(z)$ to $\partial\Omega$ and the constant c in (1.9). If $\psi(z)$ is symmetric w.r.t. the x-axis, i.e. $\psi(z) = \overline{\psi(\bar{z})}$, and $r_n(z)$ is symmetric also, then (1.9) yields $c \in \mathbb{R}$ and $c := \exp(\alpha^*)$. For nonsymmetric $\psi(z)$ we compute c from $c = ((1/2\pi)\int_0^{2\pi} \psi^{-1}(e^{i\theta})\, e^{-i\theta}\, d\theta)$. Methods for solving (2.9) are discussed in [9].

(2) Allocate the ζ_j^* by carrying out steps (a)-(d) of § 2.

(3) Solve (2.3) for the density function for the nodes z_j, $j = 1(1)n$. A low-accuracy solution suffices. Note that the system (2.3) and that solved in step (1) only differ in their right-hand sides. Compute the nodes from (2.4).

(4) Solve the linear system (1.8) for the coefficients of $r_n(z)$. By (1.9) we obtain $\psi_n(z)$.

In the computed examples we used $n := 2^p$ nodes, where p was the smallest integer such that the graph of the curve $\{\psi_n(z): z \in \partial\Omega\}$ was not distinguishable from the unit circle on the plotter.

Example 4.1. This is a continuation of Example 1.2. Let $\partial\Omega := E(2, \frac{1}{2})$ and $l := 5$. Let ζ_j^*, $j = 1(1)5$, be the zeros of the Chebyshev polynomial of degree 5 for the interval between the foci of $\partial\Omega$. The ζ_j^* are shown in Fig. 2.9. Let $z_1 = 2$, and compute $\{z_j\}_{j=2}^{64}$ according to step (3). Then $\| |\psi_{64}(z)| - 1 \|_{\partial\Omega} = 2 \cdot 10^{-4}$. Figure 4.1 shows the unit circle, $\psi_{64}(\partial\Omega)$ and $\psi(z_j)$, $j = 1(1)64$.

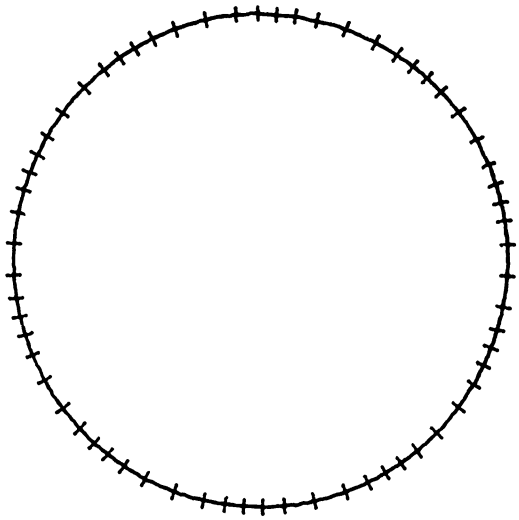


FIG. 4.1

The basis functions were scaled so that $|q_k(i/z)| = 1$ and yielded $\operatorname{cond} A_{64} = 1.2 \cdot 10^3$. The condition number decreases if we increase l and let the ζ_j^* be the zeros of an lth degree Chebyshev polynomial. □

Example 4.2. Let $\partial\Omega := \{x + iy: x^4 + y^4 = 1, x, y \in \mathbb{R}\}$, choose $l = 8$, and let the ζ_j^* be as shown in Fig. 2.5. Let $z_1 := 1$. With $n := 16$, we obtain $\| |\psi_n(z)| - 1 \|_{\partial\Omega} = 3.4 \cdot 10^{-3}$ and $\operatorname{cond} A_n = 7 \cdot 10^1$. □

Example 4.3. Let $\partial\Omega$ be the reflected ellipse obtained by reflecting $E(1.2)$ in the unit circle. Figure 4.2 shows $\partial\Omega$ and two approximate level curves of $\psi(z)$ generated according to step (2) with $\Gamma := \partial\Omega$. We choose $l = 3$ and $\zeta_1^* := 0$, $\zeta_{2,3}^* := \pm 0.37$. The ζ_j^* are marked with crosses in Fig. 4.2. Let $z_1 := 1$. Already for $n = 4$ the error in $\psi_n(z)$ is below the resolution of the plotter, $\| |\psi_n(z)| - 1 \|_{\partial\Omega} = 4.7 \cdot 10^{-3}$.

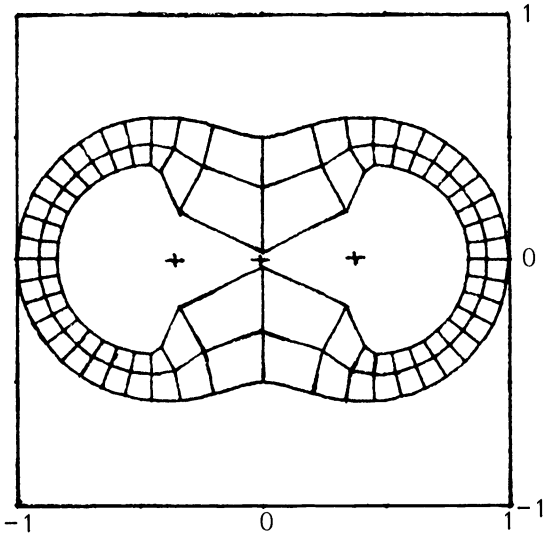


FIG. 4.2

Acknowledgment. I wish to thank Germund Dahlquist for many valuable discussions.

REFERENCES

[1] P. J. DAVIS, *Interpolation and Approximation*, Dover, New York, 1975.
[2] W. GAUTSCHI, *Conditions of polynomials in power form*, Math. Comp., 33 (1979), pp. 343-352.
[3] E. HILLE, *Analytic Function Theory*, Vol. 2, Ginn, Boston, 1959.
[4] J. C. MASON, *Recent advances in near-best approximation*, in Approximation Theory III, E. W. Cheney, ed., Academic Press, 1980, pp. 629-636.
[5] N. PAPAMICHAEL AND C. A. KOKKINOS, *Numerical conformal mapping of exterior domains*, Comp. Meth. Appl. Mech. Engng., 31 (1982), pp. 189-203.
[6] N. PAPAMICHAEL, M. K. WARBY AND D. M. HOUGH, *The determination of poles of the mapping function and their use in numerical conformal mapping*, J. Comp. Appl. Math., 9 (1983), pp. 155-166.
[7] L. REICHEL, *On complex rational approximation by interpolation at preselected nodes*, Complex Variables, 4 (1984), pp. 63-87.
[8] ———, *Solving an interface problem for the Laplace operator by the boundary collocation method*, Report TRITA-NA-8116, Dept. Computer Science, Royal Inst. of Technology, Stockholm, 1981.
[9] ———, *A fast method for solving certain integral equations of the first kind with applications to conformal mapping*, J. Comp. Appl. Math., to appear.
[10] ———, *Numerical methods for analytic continuation and mesh generation*, Constr. Approx., to appear.
[11] ———, *An asymptotically orthonormal polynomial family*, BIT, 24 (1984), pp. 647-655.
[12] ———, *On polynomial approximation in the complex plane with application to conformal mapping*, Math. Comp., 44 (1985), pp. 425-433.
[13] V. I. SMIRNOV AND N. A. LEBEDEV, *Functions of a Complex Variable*, Iliffe, London, 1968.
[14] J. L. WALSH, *The Location of Critical Points of Analytic and Harmonic Functions*, American Mathematical Society, New York, 1950.

SIAM J. SCI. STAT. COMPUT.
Vol. 7, No. 3, July 1986

025

ERRATUM: AN EXHAUSTIVE ANALYSIS OF MULTIPLICATIVE CONGRUENTIAL RANDOM NUMBER GENERATORS WITH MODULUS $2^{31}-1$*

GEORGE S. FISHMAN† AND LOUIS R. MOORE III‡

In expression (17) on p. 30, $N'_k(q_0, \cdots, q_{k-1}; A, M)$ should replace $N_k(q_0, \cdots, q_{k-1}; A, M)$.

* This Journal, 7 (1986), pp. 24–45.

† Curriculum in Operations Research and Systems Analysis, University of North Carolina, Chapel Hill, North Carolina 27514.

‡ Curriculum in Operations Research and Systems Analysis, and School of Business Administration, University of North Carolina, Chapel Hill, North Carolina 27514.

SIAM J. SCI. STAT. COMPUT.
Vol. 7, No. 4, October 1986

THEORETICAL AND NUMERICAL STRUCTURE FOR REACTING SHOCK WAVES*

PHILLIP COLELLA†, ANDREW MAJDA‡ AND VICTOR ROYTBURD§

Abstract. Several remarkable theoretical and computational properties of reacting shock waves are both documented and analyzed. In particular, for sufficiently small heat release or large reaction rate, we demonstrate that the reacting compressible Navier-Stokes equations have dynamically stable weak detonations which occur in bifurcating wave patterns from strong detonation initial data. In the reported calculations, an increase in reaction rate by a factor of 5 is sufficient to create the bifurcation from a spiked nearly Z-N-D detonation to the wave pattern with a precursor weak detonation. The numerical schemes used in the calculations are fractional step methods based on the use of a second order Godunov method in the inviscid hydrodynamic sweep; on sufficiently coarse meshes in inviscid calculations, these fractional step schemes exhibit qualitatively similar but purely numerical bifurcating wave patterns with numerical weak detonations. We explain this computational phenomenon theoretically through a new class of nonphysical discrete travelling waves for the difference scheme which are numerical weak detonations. The use of simplified model equations both to predict and analyze the theoretical and numerical phenomena is emphasized.

Key words. reacting shock waves, strong and weak detonations, Godunov's method

AMS(MOS) subject classifications. 76L05, 80A32, 65P05

1. Introduction. Through numerical experiments, several peculiar theoretical and practical computational properties regarding the structure and stability of reacting shock waves are both documented and analyzed. The waves which we study are defined by solutions of the compressible Navier-Stokes or compressible Euler equations for a mixture composed of chemically reacting species in a single space dimension.

The compressible Navier-Stokes equations for a reacting gas are extremely complex, and it is not surprising that simpler qualitative-quantitative model equations for the high Mach number regime have been developed [5], [7], [11]. These simpler model equations are a coupled 2×2 system given by a Burgers equation coupled to a chemical kinetics equation (see § 2 for a detailed description of the model equations). This model system has transparent analogues of the Chapman-Jouguet (C-J) theory, the Z-N-D theory, and also the structure of reacting shock profiles with finite diffusion and reaction rates, and these are developed in detail in [7]. One of the objectives of this paper is to use the predictions of this simplified model system both for theoretical purposes and as a diagnostic for numerical modelling of the more complex equations of reacting gas flow in the shock wave regime. The authors advocate the use of these simpler model equations for numerical code development for shock phenomena in reacting gases in much the same fashion as the Burgers equation has provided both a wide class of simple test problems and the anlysis of difference schemes for the Burgers equation has influenced code development for nonreactive compressible gas flow.

In § 2, we begin by listing the equations of compressible reacting gas flow and describing in detail the simplified model equations mentioned above; then, we describe

* Received by the editors January 22, 1985, and in final form October 4, 1985.

† Department of Mathematics, Lawrence Berkeley Laboratory, Berkeley, California 94720. The work of this author was supported by the U.S. Department of Energy under contract AC03-76SF00098.

‡ Department of Mathematics, Princeton University, Princeton, New Jersey 08544. The work of this author was partially supported by the Army Research Office under grant DAAL03-86-K-003, by the National Science Foundation under grant DMS84-0223, and by the Office of Naval Research under grant N00014-85-K-0507.

§ Department of Mathematical Sciences, Rensselaer Polytechnic Institute, Troy, New York 12181. The work of this author was partially supported by the National Science Foundation under grant DMS84-08260.

the numerical methods used in this paper. We use very natural fractional step schemes with three ingredients per time step: 1) the inviscid hydrodynamics is solved by the Godunov, second order Godunov [3], or random choice [1] methods; 2) the chemistry equation is advanced by explicit solution of the ODE for mass fraction given the temperature; 3) the diffusion equation is solved via the Crank–Nicolson or backward Euler methods. Such a class of numerical schemes is one of the obvious candidates for use in modelling reacting gases given the current development of methods for solving the compressible Euler equations. Also, with the simplified one-step kinetics schemes which we study, the chemistry equation for the mass fraction is linear given the temperature at each mesh point so that even when the reaction rate is high, this equation can be solved exactly—thus, no additional errors from solving the stiff ODE are introduced.

For the calculations in § 3, the shock layer is fully resolved, typical length scales are on the order of 10^{-6} or 10^{-5} meters, and the diffusion coefficients on such a length scale are roughly order one in magnitude. Our objectives are to document the structure and dynamic stability of reacting shock layers on such length scales where diffusive mechanisms are important. The wave structure is remarkably complex with varying heat release and reaction rate, and to our knowledge no time-dependent computations analyzing this structure have appeared previously. First, we report on detailed numerical experiments with the model equations which corroborate the rather complex behavior (see [7]) of the reacting shock profiles as the heat release varies. We use numerical experiments to predict a bifurcating wave pattern instead of the expected strong detonation for sufficiently small heat release. This bifurcating wave pattern has a precursor stable weak detonation moving at a faster speed followed by a slower moving purely fluid dynamic shock. The above experiments in the model suggest analogous behavior for the reacting compressible Navier–Stokes equations. Through numerical experiments for a detonation with fairly small heat release (modelled on an ozone decomposition detonation), we document the existence of dynamically stable weak detonations and the existence of bifurcating wave patterns as described above for the model equations. In fact, with all other parameters held fixed for this detonation wave, an increase in the reaction rate by a factor of 5 changes the wave profile from a spiked Z-N-D detonation structure to such a bifurcating wave pattern with a stable precursor weak detonation. We mention here that weak detonation waves are observed experimentally when initiated through external means [4] and that a variety of theoretical scenarios for the existence of weak detonations are given in [4, Chap. 3].

Resolving detonation waves on viscous length scales is not a practical option for a large scale reacting gas computation with many wave interactions such as the problem of transition to detonation. In § 4, we set all diffusion coefficients to be zero and investigate the problem of computing the spiked Z-N-D detonations of the inviscid reacting Euler equations on coarser meshes. This problem has practical interest because the spike in a Z-N-D profile has significantly higher values for the pressure. Any algorithm which is based on using the Chapman–Jouguet theory alone (such as [2]) automatically will ignore this local pressure spike in the travelling wave structure no matter how fine a mesh is used. The numerical experiments with the inviscid fractional step schemes with either the Godunov or second order Godunov methods exhibit the following surprising behavior:

(1.1)

A) For very fine meshes, the Z-N-D wave is completely resolved by these numerical methods.

B) For moderately fine meshes (i.e. meshes yielding very high resolution for

the second order Godunov method in the nonreactive case) and either of the fractional step methods, a numerical bifurcating wave pattern emerges with a structure qualitatively similar to those documented theoretically in § 3. This numerical wave strucure has a discrete weak detonation profile moving at the mesh speed—one grid point per time step—with all chemical energy released in this numerical precursor wave followed by a slower moving numerical shock wave.

The property in (1.1B) is an unexpected and serious defect in the use of fractional step schemes based on (higher order) Godunov methods for inviscid reacting gas calculations in the shock wave regime. On the other hand, for the simplified model equation the inviscid fractional step scheme for the random choice method yields a correct pressure spike in the Z-N-D profile with as few as three mesh points resolving the reaction zone while the split Godunov scheme has the nonphysical monotone numerical bifurcating wave pattern with as many as twenty mesh points resolving the reaction zone in the same problem (see § 4). However, in this paper, we have not pursued the use of the inviscid fractional step random choice scheme for the reacting compressible Euler equations and plan to do this in the future.

Finally, in § 5, we give a theoretical explanation for the computational phenomena on coarse meshes reported in the previous section for the Godunov methods. We work within the context of the simplified model and derive a new class of nonphysical discrete travelling waves for the difference equation for a simplified variant of the basic fractional step methods which uses the upwind scheme rather than Godunov's method. As predicted by the numerical experiments from § 4, these exact discrete travelling waves are numerical weak detonations which move at the speed $\bar{s} = \Delta x/\Delta t$, i.e. one grid point/time step and the numerical experiments from § 4 verify the stability of these purely numerical discrete weak detonations on sufficiently coarse meshes. The structure of these nonphysical discrete travelling waves is quite different from that of the well-known discrete entropy violating travelling waves [6], [8] which can occur for difference schemes in the nonreactive case. Furthermore, in the context of the simplified model, such discrete travelling waves always exist on a given mesh if either

(1.2) A) $K\Delta x$ is large enough with K the reaction rate or
B) the heat release q_0 is large enough for a fixed mesh.

The explicit conditions for the existence of numerical weak detonations provide a quantitative guideline for the validity of the basic fractional step schemes in coarser mesh calculations.

2. Preliminaries.

The compressible Navier-Stokes equations for a reacting mixture. We assume a standard simplified form for the reacting mixture throughout this paper. Thus, there are only two species present, unburnt gas and burnt gas, and we postulate that the unburnt gas is converted to burnt gas by a one-step irreversible chemical reaction. Under the above hypothesis the compressible Navier-Stokes equations for the reacting mixture [12] are the system of four equations,

$$
\begin{aligned}
&\rho_t + (\rho u)_x = 0,\\
&(\rho u)_t + (\rho u^2 + p)_x = \mu u_{xx},\\
&(\rho E)_t + (\rho u E + u p)_x = \left(\mu\left(\frac{u^2}{2}\right)_x\right)_x + c_\rho(\lambda T_x)_x,\\
&(\rho Z)_t + (\rho u Z)_x = -\rho K(T)Z + (DZ_x)_x,
\end{aligned}
\tag{2.1}
$$

where ρ is the density, u is the fluid velocity, E is the total specific energy, and Z is the mass fraction of unburnt gas. The total specific energy, E, has the form

$$E = e + q_0 Z + \frac{u^2}{2} \tag{2.2}$$

with e the specific internal energy and q_0 the amount of heat released by the given chemical reaction. For the assumed ideal gas mixture (with the same γ-gas laws), the pressure and temperature are defined respectively by the formulae $p = (\gamma - 1)\rho e$ and $T = p/\rho R \times M$ with R, Boltzmann's gas constant, M the molecular weight, c_ρ the specific heat, and γ defined by $c_\rho(\gamma - 1) = R$. The factor $K(T)$ in (2.1) is strongly dependent on temperature and has the form

$$K(T) = K_0 \phi(T) \tag{2.3}$$

with K_0 the reaction rate. The function $\phi(T)$ typically has the Arrhenius form,

$$\phi(T) = T^\alpha e^{-A/T}$$

or for computational purposes, the approximation for large A given by ignition temperature kinetics,

$$\phi(T) = \begin{cases} 1, & T \geqq \tilde{T}_0, \\ 0, & T < \tilde{T}_0 \end{cases}$$

with $\tilde{T}_0$ the ignition temperature.

The coefficients μ, λ, and D in (2.1) are coefficients of viscosity, heat condition, and species diffusion, respectively. The compressible Euler equations for the reacting mixture are the special case of (2.1) with $\mu = \lambda = D = 0$.

The simplified model equations. Obviously, even in a single space variable, the above system is extremely complex so it is not surprising that simpler qualitative-quantitative models for the equations in (2.1) have been developed [5], [7], [11]. The simplified model equations for the shock wave regime derived through asymptotic limits from the system in (2.1) (see [11]) have the form

$$\begin{aligned} &u_t + (\tfrac{1}{2}u^2 - q_0 Z)_x = \beta u_{xx}, \\ &Z_x = K\phi(u)Z \end{aligned} \tag{2.4}$$

where u is an asymptotic lumped variable with some features of pressure or temperature, Z is the mass fraction of burnt gas, $q_0 > 0$ is the heat release, $\beta \geqq 0$ is a lumped diffusion coefficient, K is the reaction rate, and $\phi(u)$ has a typical form as described below (2.3). The reader should not be confused by the appearance of Z_x on the left-hand side of (2.4) rather than Z_t. The coordinate x in (2.4) is not the space coordinate but is determined through the asymptotics as a scaled space-time coordinate representing distance to the reaction zone; the x-differentiation occurs because Z in (2.4) is convected at the much slower fluid velocity rather than the much faster reacting shock speed (see [11] for details). With these interpretations the equations in (2.4) become a well posed problem by prescribing initial data $u_0(x)$ for $u(x, t)$ at time $t = 0$ and prescribing the value of $Z(x, t)$ as $x \to \infty$ (corresponding to finite values ahead of the reaction zone with the rescaling in [11]), i.e. $Z_0(t)$ should be specified with the boundary condition,

$$Z_0(t) = \lim_{x \to \infty} Z(x, t). \tag{2.5}$$

In this paper, we always set $Z_0(t) \equiv 1$ for simplicity. The analogues of the Chapman–Jouguet theory, the Z-N-D theory, and the structure of travelling waves with nonzero

diffusion and finite reaction rates for the equations in (2.4) have all been discussed in detail in [7] and we refer the reader to that paper when we discuss properties of solutions in the model.

The numerical methods. First, we describe the basic fractional step numerical method used in solving the model equation from (2.4). We set $w=(u, Z)$. Given mesh values $w_j^N=(u_j^N, Z_j^N)$, in the first fractional step we determine $u_j^{N+1/2}$ from u_j^N by using a finite difference approximation to the inviscid Burgers equation

$$u_t+(\tfrac{1}{2}u^2)_x=0.$$

In the computations reported below, we use Godunov's method, a second order Godunov method [3], or the random choice method [1] as the finite difference approximation. In the next fractional step, we determine Z_j^{N+1} as the solution of the ODE

$$Z_x=K\phi(u)Z$$

with u given approximately by $u_j^{N+1/2}$. We march from positive values of x to negative values of x and use the boundary conditions from (2.5) with $Z_0(t)\equiv 1$ on the right-hand side of the large interval where the calculations are carried out. Given the values of $u_j^{N+1/2}$, the above ODE is linear in Z and we solve it by the trapezoidal approximations of the integral in the exact solution formula to derive

$$Z_{j-1}^{N+1}=Z_j^{N+1}\exp\left(\frac{-K\Delta x}{2}(\phi(u_{j-1}^{N+1/2})+\phi(u_j^{N+1/2}))\right) \tag{2.6}$$

with $Z_j^{N+1}\equiv 1$ for j large enough. Finally, in the third sweep of the fractional step method we solve the diffusion equation

$$u_t-\beta u_{xx}=q_0Z_x\equiv q_0K\phi(u)Z. \tag{2.7}$$

The linear diffusion equation on the left-hand side of (2.7) is discretized by using either the backward Euler or Crank-Nicolson methods with initial data $u_j^{N+1/2}$. The value of u_j^{N+1} is then determined by solving this inhomogeneous difference equation where the values for $(u_j^{N+1/2}, Z_j^{N+1/2})$ are used in the approximation of the forcing function on the extreme right-hand side of (2.7) at time level $(N+1)\Delta t$. This completes the description of the basic fractional step method for the simplified model equation. Obviously, the only stability condition needed in the method is the C-F-L condition

$$\frac{\Delta t}{\Delta x}|u_j^N|<1$$

required in the first sweep.

Next we describe the basic fractional step algorithms which we use for the reacting compressible Navier-Stokes equations in (2.1). We use three fractional steps analogous to those in the model system. In the first sweep, the inviscid nonreactive compressible Euler equations are solved, i.e. $L_E^{\Delta t}$ denotes a finite difference approximation to the equations,

$$\rho_t+(\rho u)_x=0,$$
$$(\rho u)_t+(\rho u^2+p)_x=0,$$
$$(\rho E)_t+(\rho uE+up)_x=0,$$
$$(\rho Z)_t+(\rho uZ)_x=0.$$

For this difference approximation, we use either the Godunov or a second order Godunov method [3] for an ideal γ-gas law with the mass fraction Z advected as a passive scalar. In the second fractional step all diffusion mechanisms are solved, i.e. $L_D^{\Delta t}$ is a finite difference approximation to

$$\rho_t = 0,$$

$$u_t = \frac{1}{\rho}(\mu u_x)_x,$$

$$\left(\frac{u^2}{2}\right)_t = \frac{1}{\rho}\left(\mu\left(\frac{u^2}{2}\right)_x\right)_x,$$

$$T_t = \frac{1}{\rho}(\lambda T_x)_x,$$

$$Z_t = \frac{1}{\rho}(DZ_x)_x.$$

In this difference approximation, we use the Crank-Nicolson scheme implemented in such a way that ρu, $\rho u^2/2$, ρT, and ρZ are conserved (this is why we need to discretize the trivial equation, $\rho_t = 0$). The total energy at the end of this fractional step is recovered from the formula in (2.2) with $(u^2/2)$ obtained from the kinetic energy diffusion equation. In the final sweep, we solve the chemistry equation, i.e. $L_c^{\Delta t}$ denotes the discrete solution operator for

$$\rho_t = 0,$$

$$Z_t = -K_0\phi(T)Z.$$

At each grid point, we exactly integrate the linear ODE for Z using the fixed value of temperature, $T_j^{N+2/3}$ at the grid point determined from the previous sweeps; thus,

$$Z_j^{N+1} = \exp\left(-K_0\phi(T_j^{N+2/3})\Delta t\right)Z_j^{N+2/3}.$$

This completes the description of the method used to advance the solution from level $n\Delta t$ to time level $(n+1)\Delta t$. Actually we implemented the approximation from time level $n\Delta t$ to $(n+2)\Delta t$ in the form,

$$L_{\text{total}}^{2\Delta t} = L_E^{\Delta t} L_D^{\Delta t} L_C^{\Delta t} L_C^{\Delta t} L_D^{\Delta t} L_E^{\Delta t}$$

so that we have second order accuracy in time for the algorithm. The only stability restriction on the above numerical method is the basic C-F-L condition for the inviscid hydrodynamic sweep, $L_E^{\Delta t}$.

3. The structure and stability of detonation waves with finite viscosity and reaction rate.

Wave structure for the simplified model system. Since we begin by studying the structure and dynamic stability of detonation waves for the model system, we begin with a brief summary of the surprisingly complex structure of the travelling waves for the model system in (2.4) (the quantitative details can be found in [7]). Given a preshock constant state $w_R = (u_R, 1)$ in chemical equilibrium so that $\phi(u_R) = 0$, we study travelling wave solutions of (2.4) with the given preshock state w_R and a fixed speed s. We seek special solutions of (2.4) with the form,

$$w = w\left(\frac{x - st}{\beta}\right)$$

so that with $\xi = (x - st)/\beta$

$$\lim_{\xi \to \infty} w(\xi) = (u_R, 1), \qquad \lim_{\xi \to -\infty} w(\xi) = (u_L, 0), \tag{3.1}$$

where u_L needs to be determined. With $\tilde{Z} = q_0 Z$ and $K_0 = \beta K$, substituting the above form of w into (2.4) leads to the autonomous system of two nonlinear ODE's,

$$u' = \tfrac{1}{2}u^2 - su - \tilde{Z} + C,$$
$$\tilde{Z}' = K_0 \phi(u)\tilde{Z}.$$

The integration constant C is determined by the formula,

$$C = -\tfrac{1}{2}u_R^2 + su_R + q_0,$$

and in general, there are two states u_{L^*}, u_L^* with $u_{L^*} < u_L^*$ and satisfying

$$-\tfrac{1}{2}u_R^2 + su_R + q_0 = -\tfrac{1}{2}(u_{L^*})^2 + su_{L^*} = -\tfrac{1}{2}(u_L^*)^2 + su_L^*. \tag{3.2}$$

The two states, $(u_{L^*}, 0)$ and $(u_L^*, 0)$, are the only conceivable limiting values for the second equation in (3.1) and define the end states for the corresponding weak and strong detonation waves propagating with speed s and determined by the Chapman-Jouguet theory (see [7]). When do such travelling waves exist with a finite reaction rate and nonzero diffusion for fixed s? According to the results in [7], for a fixed positive value of $K_0 = \beta K$ and fixed values u_{L^*}, u_L^*, as the heat release varies there is a critical heat release, q_{cr}, so that

(3.3)
- A) For $q_0 > q_{cr}$, a strong detonation travelling wave profile with speed s exists connecting $(u_R, 1)$ to $(u_L^*, 0)$.
- B) For $q_0 = q_{cr}$, a weak detonation travelling wave with speed s exists connecting $(u_R, 1)$ to $(u_{L^*}, 0)$.
- C) For $q_0 < q_{cr}$, no combustion wave moving with speed s is possible.

A similar behavior occurs if the heat release is fixed and K_0 is varied (see [7]); we make this remark because the reaction rate is the quantity actually varied in the calculations reported below. In fact, an even finer structure for the travelling waves in case A) of (3.3) occurs provided that the parameter $K_0 = \beta K$ satisfies either

$$u_L^* - s > K_0 \tag{3.4}$$

or

$$u_L^* - s < K_0. \tag{3.5}$$

In the case when the inequality in (3.4) is satisfied, all of the strong detonation profiles are nonmonotone and exhibit a combustion spike. However, when the case in (3.5) occurs, there is a second critical value of q_0, q_{sp}, with $q_{sp} > q_{cr}$ so that

(3.6)
- A) For $q_0 > q_{sp}$, the strong detonation profile always has a nonmonotone combustion spike.
- B) For q_0 with $q_{cr} < q_0 \leqq q_{sp}$, the strong detonation profile is monotone without a combustion spike.

See [7, Fig. 1] for graphs of the typical wave profiles described in (3.3) and (3.6) as the heat release is varied. Given the complex structure of the travelling wave profiles, it is not apparent when these profiles are dynamically stable and also what happens when q_0 satisfies $q_0 < q_{cr}$ so that no travelling wave profile moving at speed s occurs.

Next we report on a detailed numerical study using the fractional step scheme described in § 2 which addresses the above issues.

In these experiments, the viscous length scale is completely resolved and we set $\beta \equiv 1$. In all of our reported computations, we take as initial data the values defining an inviscid strong detonation wave moving with speed s, i.e.

$$u_0(x) = \begin{cases} u_R, & x > 0, \\ u_L^*, & x \leqq 0, \end{cases} \tag{3.7}$$

where given q_0 and s, the equation in (3.2) is satisfied with $u_L^* > u_{L^*}$. We use a fixed finite interval with Dirichlet boundary conditions for u at the ends determined by the respective limits, u_R and u_L^*. Also, given a wave speed s, we perform a preliminary Galilean transformation $x' = x - st$ and solve the transformed equations for zero speed waves. Besides the obvious advantage of keeping the waves from leaving the fixed computational region as time evolves, with this transformation we can also exploit the higher resolution of the Godunov scheme for nearly zero wave speeds.

In the initial experiments described below, we fixed $u_L^* = 1$, $u_{L^*} = .4$, $s = .7$ and varied the heat release q_0. We took $K = 1$, $\beta = 1$ and used ignition temperature kinetics with the ignition temperature at the value, $u = 0$. With these parameters, the value of q_{cr} from (3.3) is $q_{cr} = .568$ and that corresponds to $u_R = -.407$. Also, the inequality in (3.5) is satisfied for these parameter values and q_{sp} from (3.6) is given by $q_{sp} = .949$.

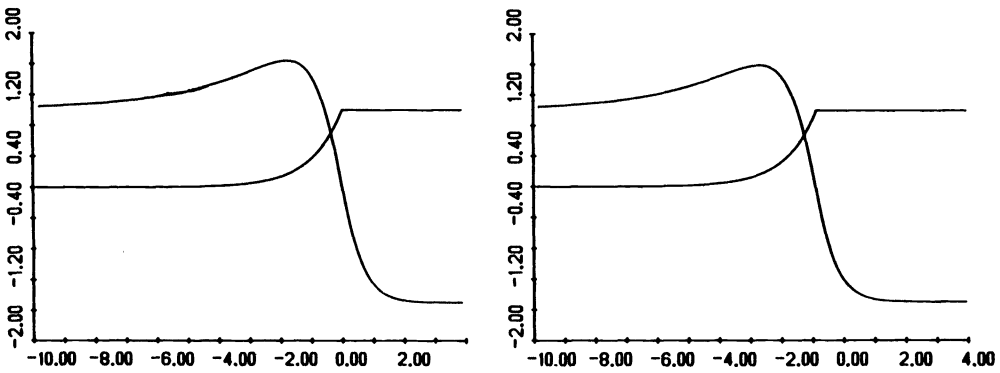


FIG. 1. *Spiked strong detonation profile for* $q_0 > q_{cr}$.

Case 1. *Spiked strong detonation profile.* We set $q_0 = 2.375 \gg q_{cr}$; this q_0 corresponds to $u_R = -1.5$. In Fig. 1(a) we present the exact spiked solution profile obtained by direct quadrature of the ODE below (3.1). In Fig. 1(b) we present the profile that emerged from dynamic stability calculations with the fractional step method described in § 2 with the initial data from (3.7). We used 560 zones on the interval $[-5, 2]$ and this dynamically computed steady profile differs from the exact solution by less than 1% in the maximum norm. This calculation both validates the method from § 2 and also demonstrates the expected stability of the spiked combustion profile.

Case 2. *Monotone strong detonation profiles,* $q_0 = q_{sp}$. We used $q_0 = q_{sp} = .949$ and with the shock tube initial data from (3.7) and only 140 zones on $[-5, 2]$, the time-dependent solution converged very rapidly (after only 50 time steps with CFL number of one-half) to the profile in Fig. 2(b)—this profile is practically identical to the exact steady solution in Fig. 2(a).

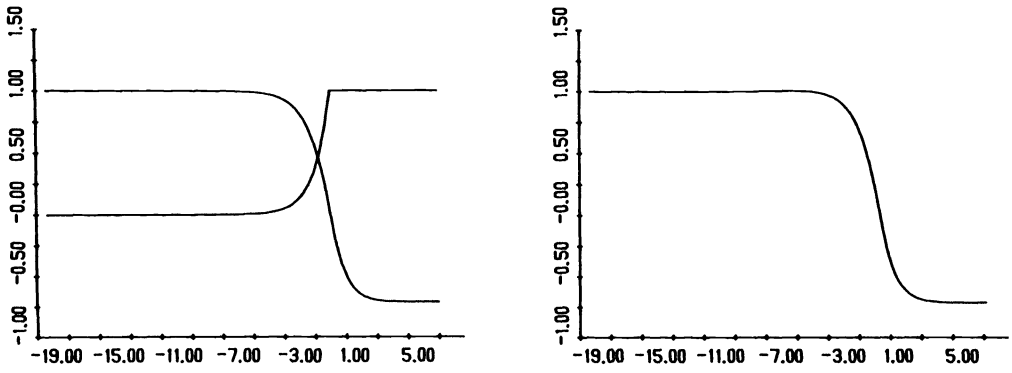


FIG. 2. *The exact steady profile* (a) *and the dynamically emerging monotone detonation profile* (b) *for* $q_0 = q_{sp}$.

Case 3. *Strong detonation profiles for* q_0 *near* q_{cr}. In the reported experiment, we set $q_0 = .571$, a value slightly larger than q_{cr}. The exact steady solution calculated by quadrature of the nonlinear ODE is given in Fig. 3(a). The profile is completely monotone with a very long characteristic flat segment with a value of u corresponding to $u \cong .4 = u_{L^*}$; we also observe that most of the chemical energy is released in this flat segment. Thus, this wave structure is almost that of the weak detonation observed for $q_0 = q_{cr}$. One might suspect that such a wave is dynamically unstable. As a numerical test, we took spiked perturbed initial data for this wave with the form depicted in Fig. 3(b) and with 560 mesh points on $[-5, 2]$. The numerical solution after 600 time steps is given in Fig. 3(c); this solution is identical to the profile in Fig. 3(a) and demonstrates the dynamic stability of this wave.

The profiles with a step shape like those in Fig. 3(a) are a difficult case for the numerical methods from § 2 on a finite interval due to the extremely long tail of the analytical steady wave in its adjustment in the step from u_{L^*} to u_L^*. In fact, with 560 mesh points and shock tube initial data, a qualitatively different steady numerical profile emerged from the calculation differing by about 15%-20% in the maximum norm. However, we emphasize here that this second profile is a numerical artifact—a second steady-state solution of the difference equations on a finite interval with a fixed mesh. Under further mesh refinement the shape of this steady solution changed substantially and finally disappeared—about 880 mesh points on $[-5, 2]$ were needed for a similar test problem with q_0 near q_{cr} to have a unique numerical steady state emerge from the dynamic calculations with a wave profile differing from the analytical profile by 2.5%.

Case 4. *Bifurcating wave structure for* $q_0 < q_{cr}$. As $q_0 \downarrow q_{cr}$, the flat step in the profile corresponding to $u_{L^*} = .4$ in Fig. 3(a) becomes even longer and as in Fig. 3(a) most of the reactant is consumed at the front of this flat segment. Once Z is nearly zero as in the back of this wave, u becomes essentially a solution of the Burgers equation and the second hump in Fig. 3(a) is an ordinary fluid dynamic shock with speed $s = (u_L^* + u_{L^*})/2 = .7$. What happens for $q_0 < q_{cr}$? No steady detonation profiles moving with speed s exist for values of q_0 with $q_0 < q_{cr}$. For a fixed u_R, q_{cr} becomes a smoothly varying function of the wave speed, s; we denote this function by $q_{cr}(s)$. By continuing the above wave profile for $q_0 > q_{cr}(s)$ to $q_0 < q_{cr}(s)$, it is natural to expect that given u_R there is a wave speed s' satisfying $s' > s$ and

$$q_0 = q_{cr}(s'). \tag{3.8}$$

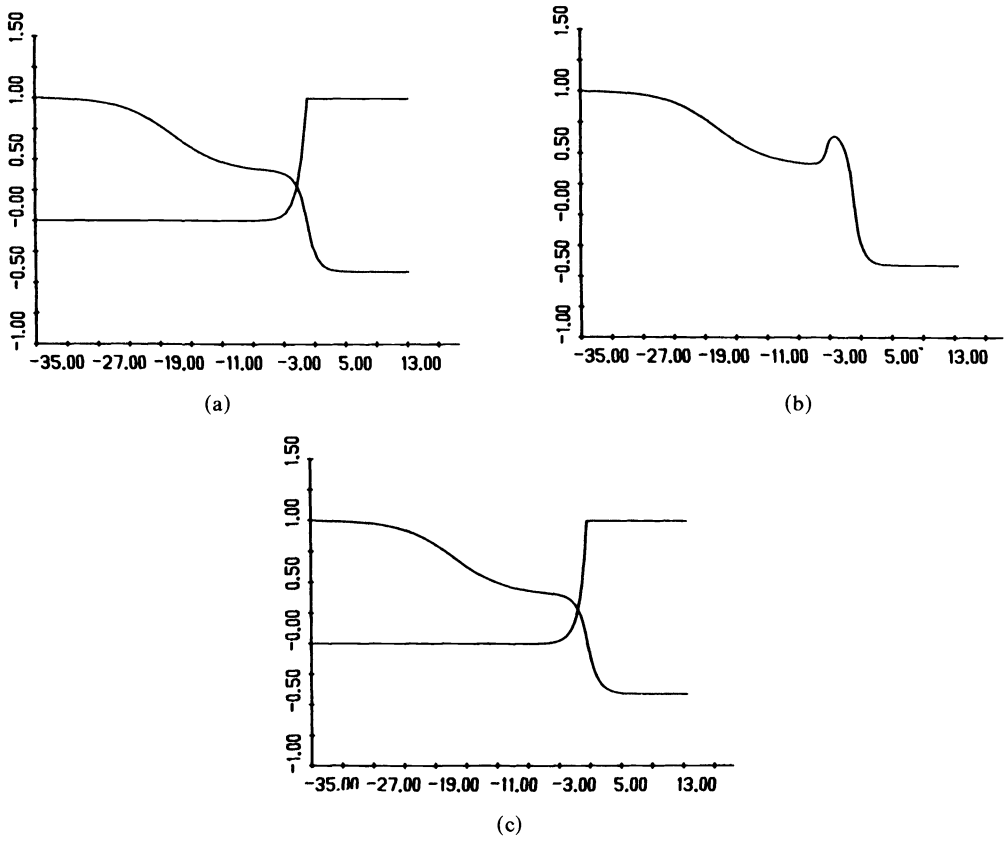


FIG. 3. *The dynamic stability of detonation profiles for* $q_0 \cong q_{cr}$; *the exact steady profile* (*Fig.* 3(a)); *the perturbed initial data* (*Fig.* 3(b)); *the dynamically emerging profile* (*Fig.* 3(c)).

If we let $u_{L^*}(s')$ with $u_{L^*}(s') < u_{L^*}(s)$ denote the value of the weak detonation satisfying (3.2) and (3.8) for the fixed u_R, then the behavior for $q_0 > q_{cr}$ suggests by continuity that the basic strong detonation shock tube initial data evolve into the following bifurcating wave pattern: an approximately self-similar wave pattern given by the faster moving weak detonation moving with speed s' from (3.8) and connecting $(u_R, 1)$ to $(u_{L^*}(s'), 0)$ with all chemical energy released in this wave followed by a slower moving fluid dynamic shock moving with the speed $\tilde{s} < s$ with $\tilde{s} = (u_{L^*}(s') + u_L^*)/2$.

Next, we describe the results of numerical experiments which confirm the behavior conjectured above. For this experiment, we used $u_R = -.02$ and $q_0 = .214$ (so that $q_0 < q_{cr}$) and retained the values of $u_{L^*} = .4$ and $u_L^* = 1.0$ used in the previous calculation; we also increased the value of K to $K = 10$. With shock tube initial data and 400 mesh points on $[-5, 2]$ the bifurcating weak detonation pattern emerged from the dynamic calculations depicted in Fig. 4 at 160, 320, and 400 time steps and persisted under mesh refinement. This precursor weak detonation has a wave speed s' exceeding s since this speed exceeds zero in Fig. 4, while the trailing fluid dynamic shock has a slightly negative wave speed.

As a second test of the stability of the weak detonation wave and also as a test of the explanation given above, we kept u_R and the heat release q_0 as in the earlier calculation, but we altered the initial data by using the initial value, $u_L = .8$ for $x < 0$. This value of u_L satisfies $u_{L^*}(s') < u_L < u_L^*$. The calculation with this initial data will

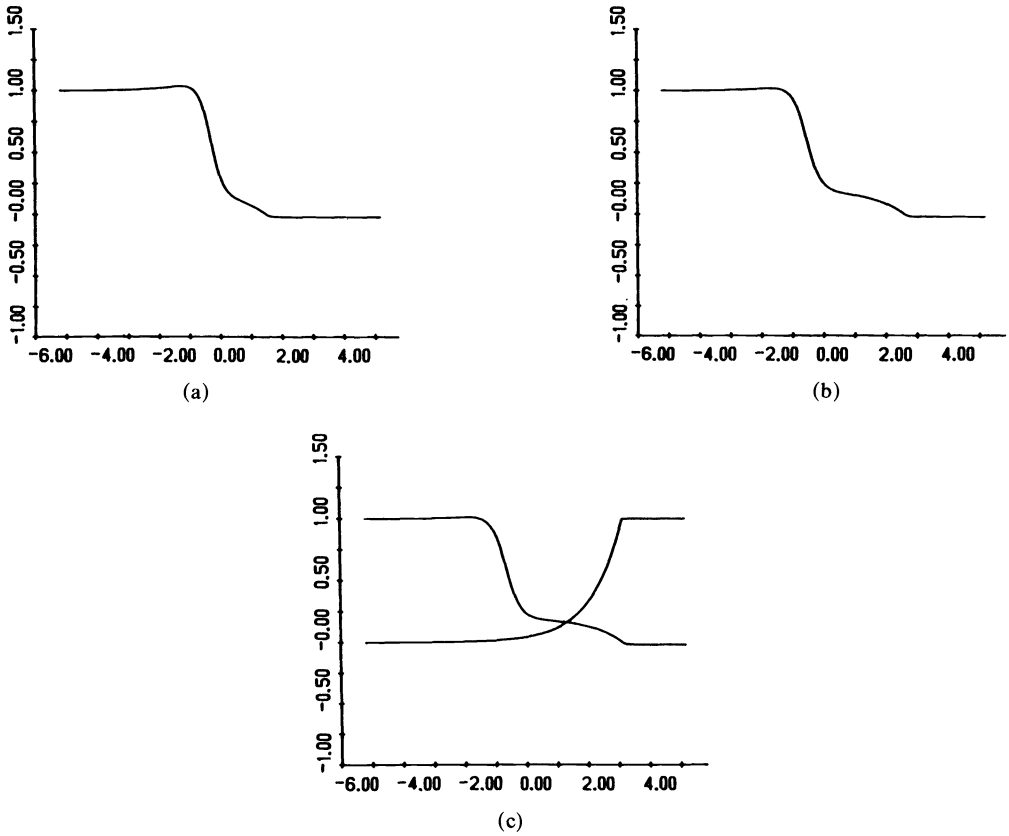


FIG. 4. *The dynamically developing bifurcating wave pattern for* $q_0 < q_{cr}$ *at* 160 (*Fig.* 4(a)), 320 (*Fig.* 4(b)), *and* 400 (*Fig.* 4(c)) *time steps.*

confirm the explanation advanced above provided that the same weak detonation as depicted in Fig. 4 emerges as a prescursor wave followed by a fluid dynamic shock moving at the slower speed $\tilde{s} = (u_L + u_{L^*}(s'))/2$. The time history of this calcuation in Fig. 5, displayed at the corresponding number of time steps as in Fig. 4, completely confirms our earlier explanation and also the stability of the weak detonation. Thus, within the context of the simplified model, we have demonstrated the existence of stable weak detonations. Similar results for these calculations with $\beta = 1$ occurred with any of the three inviscid schemes for Burgers' equation in the fractional step method. We also performed similar numerical experiments with a truncated Arrhenius kinetics form, as described below (2.3). Qualitatively similar phenomena, as documented above, always occur but for somewhat different parameter ranges.

Wave structure for the reacting compressible Navier-Stokes equations. The theory of combustion wave profiles for the reacting gas flow equations from (2.1) is considerably less complete than that for the model equations [12]. Nevertheless, Gardner [13] has recently proved the existence of viscous strong (and weak) detonations for varying (and exceptional) values of the heat release and wave speed. One consequence of the results in [13] is a scenario for the wave structure with varying heat release qualitatively similar to that mentioned in (3.3) for the model equations; in fact, his method of proof involves deformation to the travelling waves of the qualitative model from [7]. This fact both provides a partial rigorous justification for the model and also suggests that

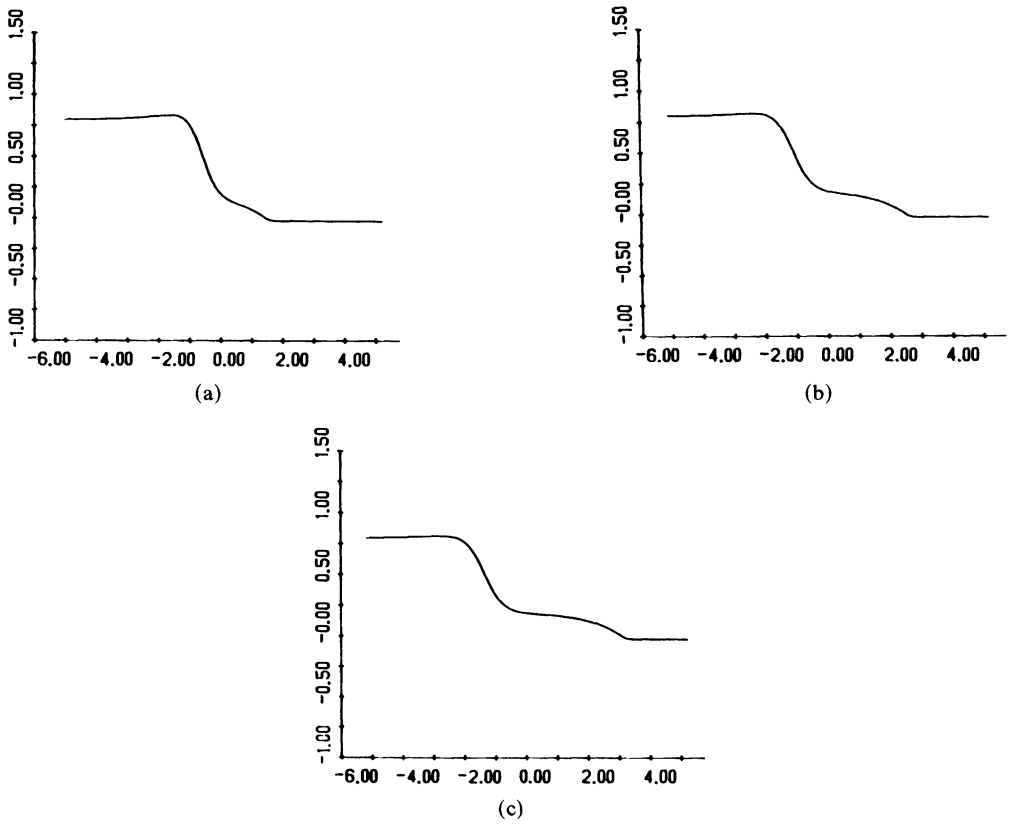


FIG. 5. *Another test of the time-dependent stability of the weak detonation from Fig.* 4 *for* $q_0 < q_{cr}$ *at* 160, 320, *and* 400 *time steps.*

similar dynamically stable wave structures, as documented earlier in this section for the model, would also occur for the reacting compressible Navier-Stokes equations. In the remainder of this section we describe a series of numerical experiments confirming this conjectured behavior.

We used the fractional step method described in § 2 with the second order Godunov method in the numerical experiments described below. We introduced the rescaled variable $\tilde{Z} = q_0 Z$ rather than Z and the initial data was always taken as the piecewise constant initial data defining a C-J (Chapman-Jouguet) detonation; i.e. the initial data for $(p, \rho, u, \tilde{Z})$ had the form

$$(p_0, \rho_0, 0, q_0), \quad x > 0,$$
$$(p_1, \rho_1, u_1, 0), \quad x \leqq 0$$

where given the preshock state for $x > 0$, the postshock state defined for $x \leqq 0$ satisfied the Rankine-Hugoniot relations defining a C-J detonation. The numerical calculations were performed on a finite interval with Dirichlet boundary conditions, and to avoid the computational expense of a very long interval, the solution was allowed to run until the wave came with a fixed number of zones from the right edge of the grid; then the solution was shifted from the right to the left to keep it fixed on the interval with new values for the zones on the right defined by $(p_0, \rho_0, 0, q_0)$—our graphical displays retain this computational artifact and focus on the fastest moving wave pattern.

In this section, diffusive length scales are completely resolved computationally, but for emphasis we will work in dimensional units which are typical ones for a viscous reacting shock layer. The detonation waves which we study have fairly small heat release and are modelled on initial data for the preshock state corresponding to 25% ozone and 75% oxygen at roughly room temperature in the ozone decomposition C-J detonation; thus, we use the documented sizes of all constants reported in the deflagration calculations from [9]. We use CGS units and the following parameter values:

$$R = 8.3143 \times 10^7, \qquad \mu = 2 \times 10^{-4},$$
$$\gamma = 1.4, \qquad p_{\text{Atm.}} = 1.0135 \times 10^6,$$
$$\lambda = \mu = D, \qquad \rho_{\text{Atm.}} = 1.29 \times 10^{-3},$$
$$M = 36.$$

For the ambient initial data, we used

$$\rho_0 = .931\ \rho_{\text{Atm}},$$
$$p_0 = .821\ p_{\text{Atm}},$$
$$e_0 = \frac{p_0}{(\gamma - 1)\rho_0}, \qquad T_0 = \frac{M p_0}{R \rho_0},$$
$$\tilde{Z}_0 = q_0 = 3 e_0.$$

With the speed of sound c_0 given by

$$c_0 = (\gamma p_0 / \rho_0)^{1/2}$$

the scalings of time, t_0, and of space, R_0, were defined by

$$t_0 = \frac{\mu}{\rho_0 c_0^2}, \qquad R_0 = t_0 c_0. \tag{3.9}$$

This choice of time and space scales corresponds to scaling compatible with the size of the reacting shock layer. Finally, in modelling the chemistry, we sometimes used the Arrhenius factor

$$K(T) = B T^{5/2}\, e^{-A/kT} \tag{3.10}$$

with $k \equiv MR$, $A = 1.00 \times 10^{12}$, and $B = 6.76 \times 10^6$; this is the value of the dominant forward rate in the ozone decomposition reaction (see [9]). In other calculations we used ignition temperature kinetics with the form

$$K(T) = \begin{cases} \dfrac{K_0}{t_0} & \text{if } T > \tilde{T}_0, \\ 0 & \text{if } T < \tilde{T}_0. \end{cases} \tag{3.11}$$

For the ignition temperature with the above detonation, we used $\tilde{T}_0 = 500°\text{K}$. We always used 300 mesh points in all computations on the fixed interval but increased (decreased) the resolution by setting $\Delta x = \alpha R_0$ with α a scaling factor. To avoid repetition, we only report the results of computations with the kinetics scheme in (3.11) because the kinetics structure function in (3.10) gave qualitatively similar behavior.

Case 1. *A* C-J *detonation with a nearly* Z-N-D *spike.* We set $K_0 = 1$ and report on the time dependent development of the wave that emerged from the C-J initial data described above with $\Delta x = .025 R_0$. The Z-N-D detonation (see § 4) has a pressure peak

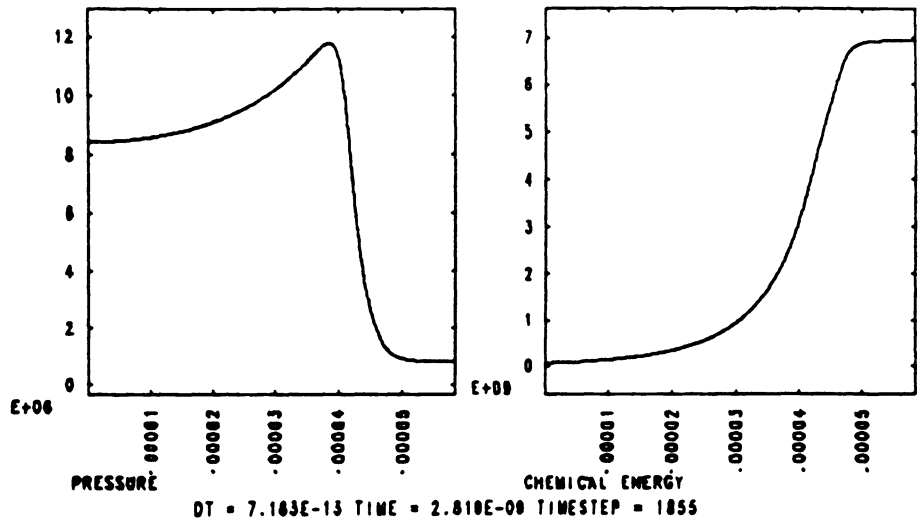


FIG. 6. *Dynamically emerging* C-J *detonation wave with nearby* Z-N-D *spike for* $K_0 = 1$.

of 12 atm. The pressure and chemical energy wave profiles of the solution that emerged from the dynamic calculation is given in Fig. 6. This solution is numerically steady in a reference frame moving with wave speed and is nearly a Z-N-D detonation since the pressure rises to a value of nearly 12 atm, then drops to the C-J value slightly below 8 atm. The width of this C-J detonation wave is roughly 10^{-4} cm. This is compatible with older estimates using explicit integration in the phase plane for the laminar ozone detonation wave thickness [10]. This calculation is a refinement of one with $\Delta x = .05 R_0$ where a profile of identical size and structure emerged.

Case 2. *Bifurcating wave patterns and dynamically stable weak detonations.* By increasing the value of the reaction prefactor K_0 but keeping the heat release and the initial C-J data fixed, by analogy with the structure documented earlier for the model system, one might anticipate a bifurcating wave pattern with a dynamically stable precursor weak detonation wave once q_0 satisfies $q_0 < q_{cr}(K_0)$. In the calculations reported in the time sequence from Fig. 7 we have kept all parameters in the calculation from Case 1 fixed except K_0. We have increased K_0 from $K_0 = 1$ to $K_0 = 5$. Only the pressure and chemical energy plots are displayed in Fig. 7. The graphs display successive time plots of the profile but focus increasingly on the precursor hump given by the stable weak detonation wave. The reader can see that all chemical energy is released in this precursor weak detonation wave as anticipated in the model system; furthermore, this wave is supersonic from both the front and back. The slower moving trailing wave profile is an ordinary fluid dynamic shock. We remark that the same wave profile emerged under the mesh refinement with $\Delta x = .015 R_0$. It is somewhat surprising that a change in the reaction prefactor of 5 in the given detonation wave accounts for a transition from a dynamically stable strong detonation to a bifurcating wave pattern with a stable precursor weak detonation.

4. The behavior of fractional step methods for computing Z-N-D detonations. The computational meshes used in the calculations from § 3 are several orders of magnitude finer than those that could be used in a typical large scale computing problem. On much larger spatial scales the effects of diffusion are ignored so in this section we report on calculations with the inviscid reacting compressible Euler equations. Since it is an interesting problem to develop numerical methods which can capture the

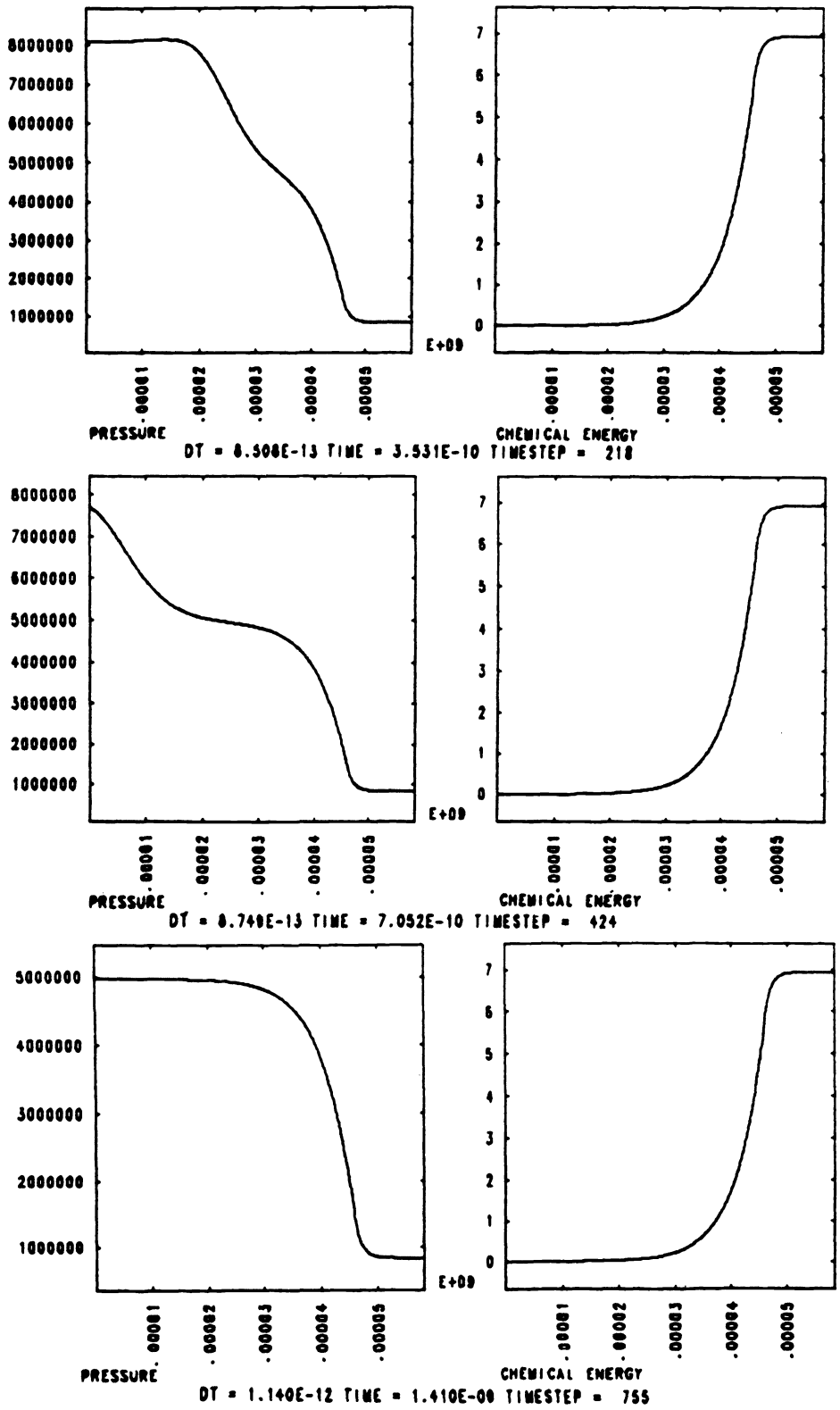


FIG. 7. *Dynamically emerging precursor weak detonation with* $K_0 = 5$ *but all other parameters and initial data fixed as in Fig. 6.*

significantly higher pressure peaks which occur in the structure of Z-N-D waves, we assess the performance of the inviscid fractional step methods of § 2 in such a calculation.

Coarse mesh calculations for the reacting Euler equations. For comparison, we used as initial data the same C-J detonation wave which we used previously in § 3. In the reported calculations we always used 300 mesh points with $\Delta x = \alpha R_0$. We recall that R_0 is a characteristic length scale which measured the internal structure of the reaction zone. In fact, by using Fig. 6, we see that $30\ R_0 = 1.5 \times 10^{-4}$ cm = "approximate width of the nearly Z-N-D detonation" computed in § 3. We used either the Godunov or second order Godunov scheme in the inviscid calculations below with $L_D^{\Delta t} = I$.

The graphs in Fig. 8 display the values of the pressure and chemical energy for the travelling waves that emerged from these calculations with the C-J initial data. The

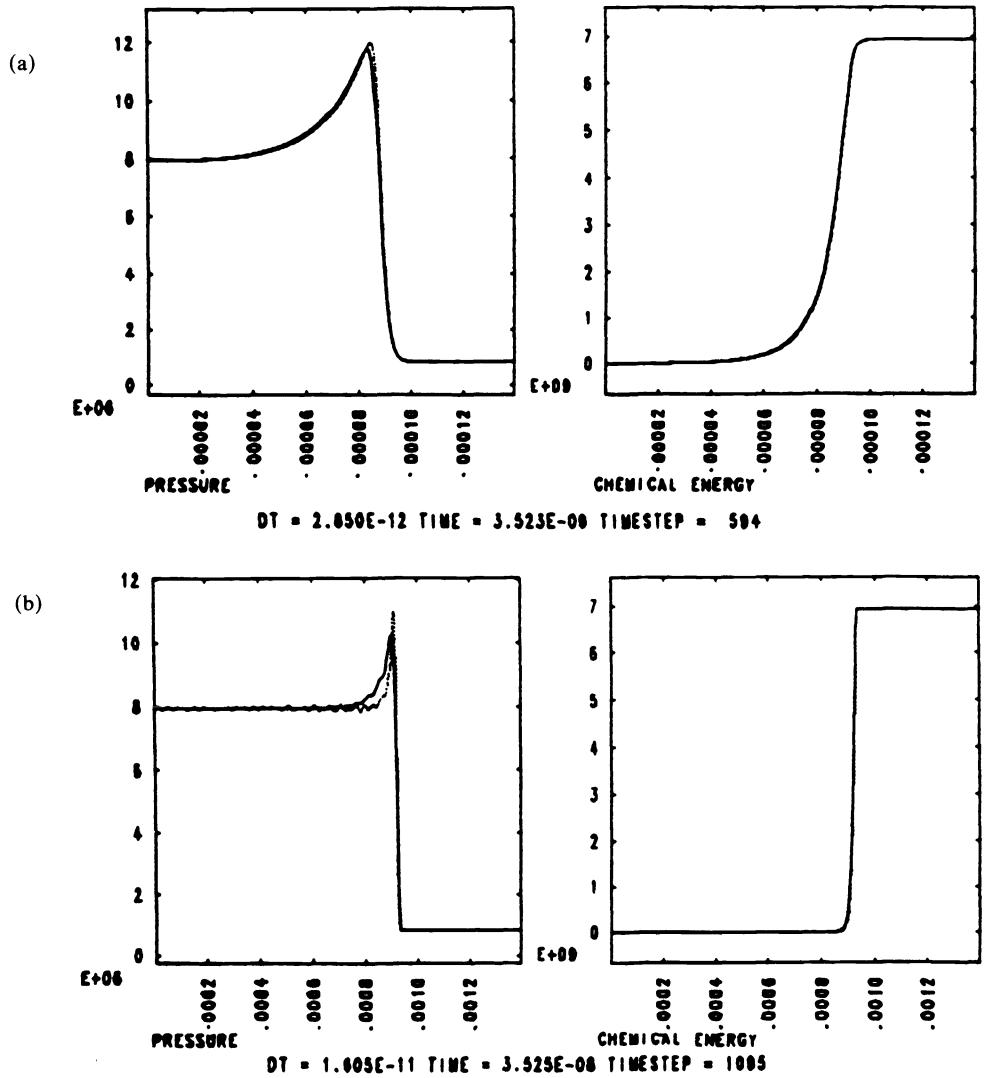


FIG. 8. *Dynamically emerging numerical wave patterns with the Godunov schemes and meshes* $\Delta x = .1\ R_0$, $\Delta x = R_0$, $\Delta x = 10\ R_0$, $\Delta x = 10^2\ R_0$, $\Delta x = 10^5\ R_0$. *Only the pressure and chemical energy are displayed. The black line represents the Godunov method while the dashed line represents the high order Godunov method.*

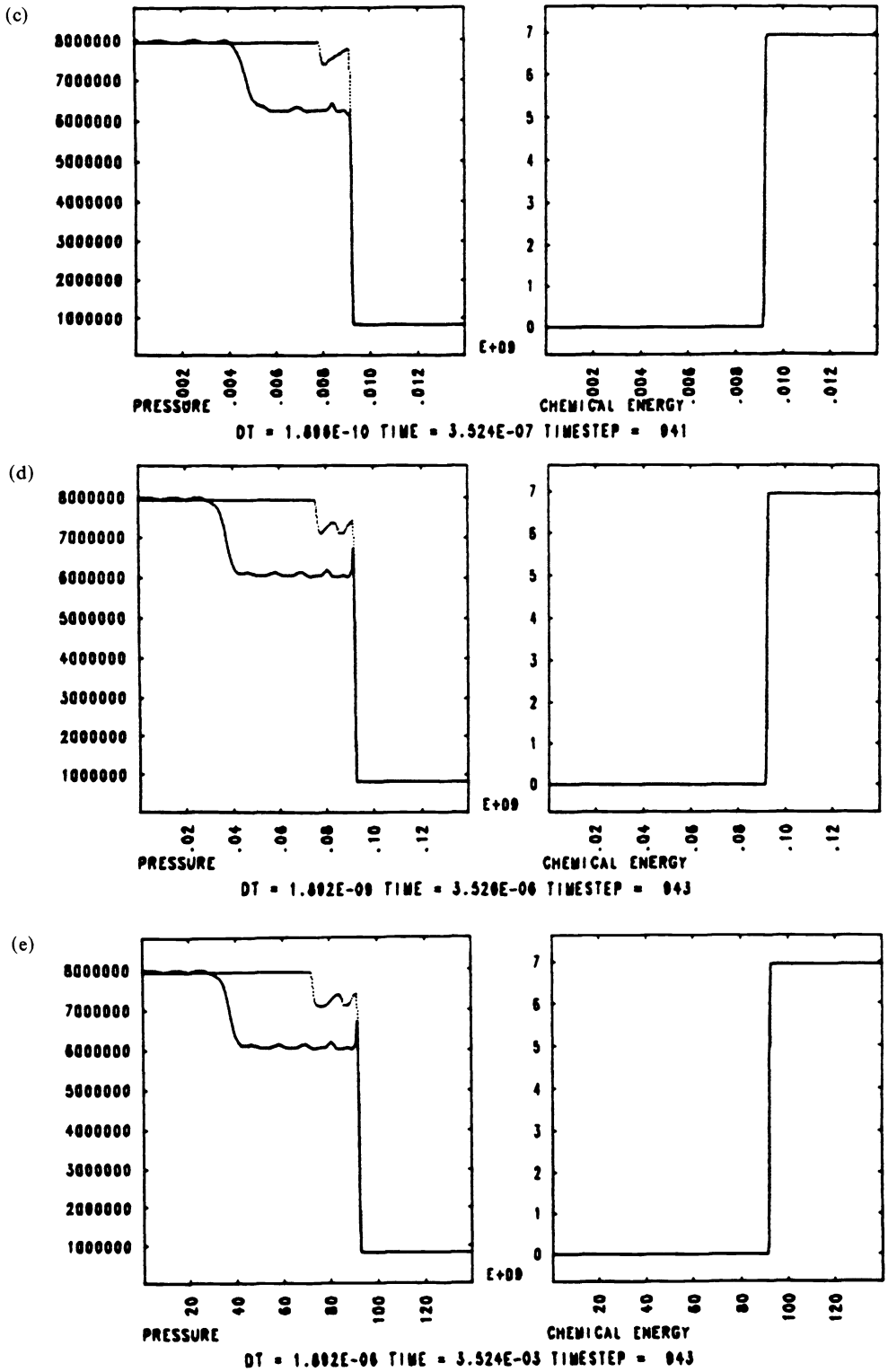


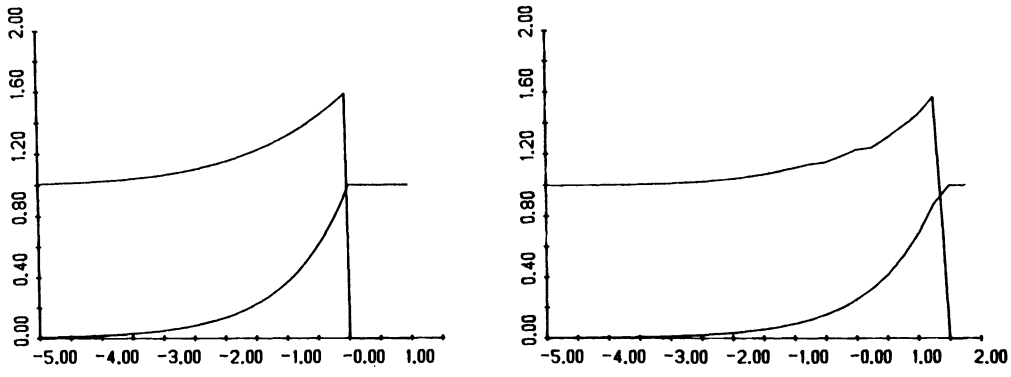
FIG. 8. (*Continued.*)

dashed line describes the results of computations using the second order Godunov method while the black line describes the results for the Godunov method. We increased the value of α in the calculations reported in successive plots and thus, we used increasingly coarse meshes.

For $\Delta x = .1 R_0$, the reaction zone was completely resolved and the expected Z-N-D profile was computed by either method. For $\Delta x = R_0$ so that there are roughly 30 points in the reaction zone, both methods gave a C-J detonation moving at the correct speed but the Z-N-D pressure peak predicted by the Godunov method was only 10 atm, rather than the expected 12 atm. Already at $\Delta x = 10\ R_0$, neither numerical method has any pressure peak higher than 8 atm. On this mesh the Godunov scheme already clearly exhibits a numerical bifurcating weak detonation pattern qualitatively similar to the one described in § 3 with all chemical energy released too soon in the precursor numerical weak detonation wave. The second order Godunov method also exhibits an incorrect wave pattern on this mesh and this value of α is at the critical value for numerical wave bifurcation for this numerical method. On a mesh with $\Delta x = 10^2\ R_0$, both methods clearly exhibited totally nonphysical bifurcating wave patterns with precursor numerical weak detonations. On even coarser meshes, the same approximately self-similar nonphysical discrete wave pattern emerged as indicated by a comparison of the graphs in Fig. 8(e) with $\Delta x = 10^5\ R_0$ and Fig. 8(d) with $\Delta x = 10^2\ R_0$. We recall that the mesh with $\Delta x = 10^5\ R_0$ has 300 mesh points in a region only 1.5 meters long. Although we do not report the detailed time history here for these calculations, the numerical weak detonation wave that emerges is always moving at the speed of one mesh point per time step. Qualitatively similar results occurred in our computations with an Arrhenius kinetics structure function. The theory for numerical weak detonations developed in § 5 indicates that this numerical bifurcating wave phenomenon should occur on even finer meshes for detonations with larger heat release (our test problem has rather small heat release).

Coarse mesh calculations for the model equations. A similar computational phenomenon occurred for the fractional step schemes for the model system with the Godunov or second order Godunov methods. On the other hand, the inviscid fractional step scheme using the random choice method performed extremely well and a numerical bifurcating wave pattern was never observed on even the coarsest meshes tested. For example, in Fig. 9 we compare the exact Z-N-D profile and the numerical wave profile for a calculation with only 25 mesh points on the interval $[-5, 2]$ for the random choice fractional step method. The agreement is astonishing given the coarse mesh, and almost the complete pressure peak has been captured. In contrast, for the same initial data the fractional step scheme with Godunov's method produced the nonphysical numerical bifurcating wave pattern with 100 mesh points. These experiments suggest that at least in a single space dimension, the fractional step scheme using the random choice method might be capable of coarse mesh resolution of pressure peaks in wave structure for solutions of the reacting compressible Euler equations involving complex chemistry.

5. Discrete weak detonations: nonphysical but stable discrete travelling waves. The calculations from § 4 on coarser meshes with the Godunov fractional step schemes yield a bifurcating numerical wave pattern with a discrete weak detonation wave as a precursor. These wave patterns qualitatively resemble the analytic bifurcating wave structures documented as stable exact solutions of the reacting Navier-Stokes equations in § 3. However, the wave patterns from § 4 are purely a numerical artifact since the numerical solution converged to the expected Z-N-D detonation under further mesh refinement.



(a) (b)

FIG. 9. *A coarse mesh calculation for the model system using the random choice fractional step method (Fig. 9(b)) compared with the exact steady profile (Fig. 9(a)).*

Here we provide a theoretical explanation for the numerical results presented in § 4. We work within the context of the simplified model and derive a new class of nonphysical discrete travelling waves for the basic inviscid fractional step scheme introduced in § 2. These exact solutions of the difference equations will be numerical weak detonations moving at the speed, $\bar{s} = \Delta x/\Delta t$, i.e. one grid space per time step, as observed in the calculations from § 4. Of course, we have already demonstrated the stability of such nonphysical discrete weak detonations in the calculations reported in § 4 for sufficiently coarse meshes.

Within the context of the simplified model, in the last section we considered the problem of computing the Z-N-D detonation dynamically as a solution of

$$\begin{aligned} &u_t + (\tfrac{1}{2}u^2 - q_0 Z)_x = 0, \\ &Z_x = K\phi(u)Z \end{aligned} \tag{5.1}$$

from initial data given by a C-J or strong detonation wave, i.e., $w = {}^t(u, Z)$ has initial data with the form in (3.7) for the fixed wave speed s. We introduce the Hugoniot function defined by

$$H(u, u_R, s) = s(u - u_R) - (\tfrac{1}{2}u^2 - \tfrac{1}{2}u_R^2). \tag{5.2}$$

For simplicity we always assume that the initial data from (3.7) satisfy $u_R > 0$ so that for this strong or C-J detonation, we have

$$u_L^* \geqq s > u_R > 0. \tag{5.3}$$

These initial data also satisfy the reacting Hugoniot equation

$$H(u_L^*, u_R, s) = q_0. \tag{5.4}$$

For the inviscid fractional step schemes of the last section, we required the C-F-L stability condition

$$\frac{\Delta t}{\Delta x} u_L^* = \alpha < 1. \tag{5.5}$$

Given the mesh, we introduce the discrete wave speed $\bar{s} = \Delta x/\Delta t$. From (5.5) and (5.3) it follows that $\bar{s}$ satisfies $\bar{s} > s$ and one easily verifies the following fact:

For any $\bar{s} > s$, there are always exactly two solutions $\bar{u}_L^*$, $\bar{u}_{L^*}$ satisfying

$$\begin{aligned} &H(\bar{u}_L^*, u_R, \bar{s}) = q_0, \qquad \bar{u}_L^* > \bar{s} > \bar{u}_{L^*}, \\ &H(\bar{u}_{L^*}, u_R, \bar{s}) = q_0, \qquad \bar{u}_L^* > u_L^* > \bar{u}_{L^*} > u_R. \end{aligned} \tag{5.6}$$

The wave defined by $(\bar{u}_{L^*}, u_R)$ is an inviscid weak detonation wave with speed $\bar{s}$ while the wave $(\bar{u}_L^*, u_R)$ is an inviscid strong detonation wave travelling with the same speed. We observe that

$$\bar{u}_{L^*}\frac{\Delta t}{\Delta x} < u_L^*\frac{\Delta t}{\Delta x} < \bar{u}_L^*\frac{\Delta t}{\Delta x} > 1$$

and the weak detonation always satisfies the C-F-L stability condition from (5.5) on the computational mesh but the strong detonation will always violate this C-F-L condition is (5.5). The numerical computations from § 4 indicate that on sufficiently coarse meshes, the difference equations for the inviscid fractional step schemes based on Godunov's method should have discrete travelling wave solutions, $w_j^N = (u_j^N, Z_j^N)$, satisfying the equations

$$w_j^N = w_{j-N}^0 \quad \text{for all } N \geqq 0 \text{ and } j, \tag{5.7A}$$

with the discrete wave profile w_j^0 having the structure

$$\begin{aligned} &w_j^0 = (u_R, 1), \qquad j \geqq 1, \\ &\lim_{j \to -\infty} w_j^0 = (\bar{u}_{L^*}, 0). \end{aligned} \tag{5.7B}$$

Such solutions of the numerical scheme define the nonphysical discrete weak detonations moving at mesh speed which were observed computationally in the last section. Here we will verify the following result:

PROPOSITION (existence of numerical weak detonations). *For a simplified inviscid fractional step scheme (see* (5.10) *and* (5.11) *below) based on the upwind scheme rather than Godunov's scheme, explicit nonphysical travelling waves satisfying the structure in* (5.7A) *and* (5.7B) *exist under the following conditions on heat release,* q_0, *reaction rate,* K, *and mesh spacing,* Δx:

A) *For ignition temperature kinetics with ignition temperature* $\tilde{u}$ *satisfying* $\tilde{u} > u_R$, *nonphysical discrete travelling waves with a monotone profile exist provided the two explicit inequalities*

$$\tilde{u} < \bar{u}_{L^*} \text{ and } H(\tilde{u}, u_R, \bar{s}) < q_0(1 - e^{-K\Delta x/2}) \tag{5.8}$$

are satisfied.

B) *For a general kinetics structure function* $\phi(u)$ *satisfying* $\phi(u_R) = 0$ *and* $\phi(u) > 0$ *for* $u_R < u$, *a numerical weak detonation profile exists with the structure in* (5.7) *provided that there is a solution* u_0 *with* $u_R < u_0 < \bar{u}_{L^*}$ *to the nonlinear algebraic equation*

$$H(u_0, u_R, \bar{s}) + q_0\, e^{-K\Delta x \phi(u_0)/2} = q_0. \tag{5.9}$$

Remark 1. It is easy to see that either of the quantitative algebraic conditions in (5.8) or (5.9) is satisfied provided that either $K\Delta x$ is sufficiently large or the heat release q_0 increases. In fact, the quantity $\bar{K} = K\Delta x$ for these inviscid fractional step methods for reacting gases has an analogous role as the mesh Reynolds number in viscous incompressible flow. The behavior of the numerical methods for $\bar{K}$ large for the reacting compressible Euler equations mimics the behavior for high reaction rate K_0 documented in § 3 for the reacting compressible Navier-Stokes equations.

Remark 2. The same construction which we give below in the proof of the proposition will establish the existence of a spiked Z-N-D strong detonation discrete wave profile moving with mesh speed, i.e., a discrete travelling wave satisfying (5.7A) with

$$w_j^0 = (u_R, 1), \qquad j \geqq 1,$$
$$\lim_{j \to -\infty} w_j^0 = (\bar{u}_L^*, 0),$$

and $u_j > \bar{u}_L^*$ for $j \leqq 0$. Because we have the C-F-L restriction below (5.6), this wave is never realized on the given computational mesh; however, it might occur in similar fractional step schemes based on implicit methods.

First, we describe the variant of the inviscid fractional step schemes from § 2 based on the upwind scheme. Given $w_j^N = (u_j^N, Z_j^N)$, in the first fraction step, we compute Z_j^{N+1} via numerical integration of the ODE to obtain the formula

$$Z_{j-1}^{N+1} = Z_j^{N+1} \exp\left(-\bar{K}\frac{\phi(u_j^N) + \phi(u_{j-1}^N)}{2}\right) \tag{5.10}$$

with initial condition $Z_j^{N+1} = 1$ for j large enough and $\bar{K}$ defined by $\bar{K} = K\Delta x$. For waves moving with positive wave speed as guaranteed by (5.3), Godunov's scheme reduces to the upwind scheme. In the second step of the simplified algorithm, we compute u_j^{N+1} from $\{u_j^N\}$, $\{Z_j^{N+1}\}$ by applying the upwind difference approximation to the first equation in (5.1). This results in the formula for u_j^{N+1} given by

$$u_j^{N+1} = u_j^N - \frac{\Delta t}{\Delta x}\left(\frac{1}{2}(u_j^N)^2 - \frac{1}{2}(u_{j-1}^N)^2\right) + q_0\frac{\Delta t}{\Delta x}(Z_j^{N+1} - Z_{j-1}^{N+1}). \tag{5.11}$$

The formulae in (5.10), (5.11) describe how to compute $\{w_j^{N+1}\}$ from $\{w_j^N\}$ in this fractional step method. Next we prove the proposition for this scheme.

The equations in (5.7A) will be satisfied provided that we find an initial wave profile $w_j^0 = (u_j, Z_j)$ satisfying

$$w_j^1 = w_{j-1}^0 \quad \text{for all } j. \tag{5.12}$$

By explicitly computing w_j^1 from the fractional step method in (5.10), (5.11), we see that (5.12) will be satisfied provided that

$$Z_{j-1} = \exp\left(-\bar{K}\frac{\phi(u_{j+1}) + \phi(u_j)}{2}\right)Z_j, \tag{5.13A}$$

$$H(u_{j-1}, u_j, \bar{s}) = q_0(Z_{j-1} - Z_{j-2}) \tag{5.13B}$$

for j with $-\infty < j < \infty$. First, we concentrate on the case of ignition temperature kinetics. With $w_j = (u_R, 1)$ for $j \geqq 1$, the equations in (5.13) are trivially satisfied for $j \geqq 2$. From (5.13A) we see that $Z_0 = 1$ and if a solution $u_0 > \tilde{u}$ is found, u_0 is the solution of the equation

$$H(u_0, u_R, \bar{s}) = q_0(1 - e^{-\bar{K}/2}). \tag{5.14}$$

The Hugoniot function, $H(u, u_R, \bar{s})$ has the three properties

$$\begin{aligned} &H(u_R, u_R, \bar{s}) = 0, \\ &H(\bar{u}_{L^*}, u_R, \bar{s}) = q_0, \\ &H(u, u_R, \bar{s}) \text{ is monotone increasing in } u \text{ for } u_R < u < \bar{u}_{L^*}. \end{aligned} \tag{5.15}$$

Given the conditions in (5.8) and the above three properties, we see that there is a solution u_0 to the equation in (5.14) satisfying

$$\tilde{u} < u_0 < \bar{u}_{L^*}. \tag{5.16}$$

Next, we generate the u_j for $j<0$ recursively from u_{j+1} by a similar procedure. We anticipate the fact to be verified a posteriori that u_j for $j<0$ also satisfies $\tilde{u} < u_j < \bar{u}_{L^*}$. We define α to be the factor $\alpha = e^{-\bar{K}/2}$; if u_j for $j \leqq 0$ inductively satisfies $u_j > \tilde{u}$, then from (5.13A), we compute that Z_j is given by the formula

$$Z_j = \alpha^{-2j-1}, \qquad j = -1, -2, -3, -4, \cdots. \tag{5.17}$$

With the formula in (5.17), the equations in (5.13B) will be satisfied inductively provided that

$$H(u_j, u_R, \bar{s}) = q_0(1 - Z_{j-1}) = q_0(1 - \alpha^{-2j+1}) \equiv q_{j-1} \tag{5.18}$$

for $j = -1, -2, -3, \cdots$. Since we have the monotone sequence

$$q_0 > q_{j-1} > q_j, \qquad j = -1, -2, -3, \cdots,$$

it follows from (5.15) that there are always solutions u_j to the equations in (5.18) with the monotone structure

$$\tilde{u} < u_0 < u_j < u_{j-1} < \bar{u}_{L^*}$$

for $j = -1, -2, -3, \cdots$. From the above monotone structure and the equations in (5.18), it is easy to see that the unique limit $\bar{u}$ of this sequence as $j \downarrow -\infty$ satisfies

$$\bar{u} \in (\tilde{u}, u_{L^*}], \qquad H(\bar{u}, u_R, \bar{s}) = q_0.$$

The only solution of these equations is $\bar{u} = \bar{u}_{L^*}$ and clearly from (5.17), $Z_j \downarrow 0$ rapidly as $j \downarrow -\infty$. This completes the construction of the explicit travelling wave for ignition temperature kinetics. Obviously a similar recursive construction can be applied for the more general kinetics schemes. The only difference is that the right-hand side of (5.14) or (5.18) also depends on u_j. However, the assumption in (5.9) guarantees that u_0 can be found and the other equations are easily solved inductively—we omit the details.

BIBLIOGRAPHY

[1] A. CHORIN, *Random choice solution of hyperbolic systems*, J. Comput. Phys., 22 (1976), pp. 517-533.

[2] ———, *Random choice methods with applications for reacting gas flow*, J. Comput. Phys., 25 (1977), pp. 253-272.

[3] P. COLELLA AND P. WOODWARD, *The Piecewise-parabolic method* (PPM) *for gas-dynamical simulations*, J. Comput. Phys., 54 (1984), pp. 174-201.

[4] W. FICKETT AND W. C. DAVIS, *Detonation*, Univ. California Press, Berkeley, CA, 1979.

[5] W. FICKETT, *Detonation in miniature*, Amer. J. Phys., 47 (1979), pp. 1050-1059.

[6] A. HARTEN, J. M. HYMAN AND P. D. LAX, *On finite difference approximations and entropy conditions for shocks*, Comm. Pure Appl. Math., 29 (1976), pp. 297-322.

[7] A. MAJDA, *A qualitative model for dynamic combustion*, SIAM J. Appl. Math., 41 (1981), pp. 70-93.

[8] A. MAJDA AND J. RALSTON, *Discrete shock profile for systems of conservation laws*, Comm. Pure Appl. Math., 32 (1979), pp. 445-482.

[9] S. B. MARGOLIS, *Time-dependent solution of a premixed laminar flame*, J. Comput. Phys., 27 (1978), pp. 410-427.

[10] A. K. OPPENHEIM AND J. ROSCISZEWSKI, *Determination of detonation wave structure*, Ninth International Symposium on Combusition, Academic Press, New York, 1963, pp. 424-434.

[11] R. ROSALES AND A. MAJDA, *Weakly nonlinear detonation waves*, SIAM J. Appl. Math., 43 (1983), pp. 1086-1118.

[12] F. A. WILLIAMS, *Combustion Theory*, Addison-Wesley, Reading, MA, 1965.

[13] R. GARDNER, *On the detonation of a combustible gas*, Trans. Amer. Math. Society, 277 (1983), pp. 431-468.

SIAM J. SCI. STAT. COMPUT.
Vol. 7, No. 4, October 1986

A MULTIGRID CONTINUATION METHOD FOR ELLIPTIC PROBLEMS WITH FOLDS*

JOHN H. BOLSTAD† AND HERBERT B. KELLER‡

Abstract. We introduce a new multigrid continuation method for computing solutions of nonlinear elliptic eigenvalue problems which contain limit points (also called turning points or folds). Our method combines the frozen tau technique of Brandt with pseudo-arc length continuation and correction of the parameter on the coarsest grid. This produces considerable storage savings over direct continuation methods, as well as better initial coarse grid approximations, and avoids complicated algorithms for determining the parameter on finer grids. We provide numerical results for second, fourth and sixth order approximations to the two-parameter, two-dimensional stationary reaction-diffusion problem:

$$\Delta u + \lambda \exp(u/(1 + \alpha u)) = 0.$$

For the higher order interpolations we use bicubic and biquintic splines. The convergence rate is observed to be independent of the occurrence of limit points.

Key words. multigrid, arc-length continuation, nonlinear elliptic eigenvalue problems, limit points, folds, frozen tau method, tau extrapolation, deferred correction, defect correction

AMS(MOS) subject classifications. 65N20, 65N25

1. Introduction. Many problems of computational interest can be viewed in the general form:

$$G(u, \boldsymbol{\lambda}) = 0, \tag{1.1}$$

where X is some Banach space, $u \in X$, $\boldsymbol{\lambda} \in R^{\nu}$, and $G: X \times R^{\nu} \to X$. Of course u represents a "solution" field (e.g., flow field, displacements, etc.), and $\boldsymbol{\lambda}$ is a real vector of physical parameters (e.g., Reynolds number, load, etc.). It is required to find the solution for some $\boldsymbol{\lambda}$-intervals (or some $\boldsymbol{\lambda}$-arcs), that is, a path (or manifold) of solutions: $(u(\boldsymbol{\lambda}), \boldsymbol{\lambda})$. We consider problems of the form (1.1) which are nonlinear elliptic eigenvalue problems.

A common feature on solution paths of such problems is the frequent occurrence of *limit points* (also called *turning points* or *folds*). Figure 1 illustrates two limit points in the $(\lambda, \|u\|_{\infty})$ plane for two different families of numerical solutions. The limit points are at $\lambda = \lambda_f$ and λ_c. A *limit point* $P^0 = (u^0, \lambda^0)$ is defined as a solution of (1.1) for which the Fréchet derivative G_u^0 of G with respect to u evaluated at P^0 satisfies:

a) G_u^0 is singular;

b) $G_\lambda^0 \notin R(G_u^0)$.

The limit point or fold is said to be *simple* if in addition:

c) $\dim N(G_u^0) = \operatorname{codim} R(G_u^0) = 1$.

Here N and R denote null space and range, respectively. In the rest of this paper, when we refer to limit points, we shall mean simple limit points.

Pseudo-arc-length continuation methods (Keller [19], [20]) have been used successfully for computing limit points and bifurcation points of nonlinear elliptic eigen-

* Received by the editors February 8, 1984, and in revised form June 13, 1985. This work was typeset at the Lawrence Livermore National Laboratory using a *troff* program running under UNIX. The final copy was produced on July 23, 1985. It was supported by the U.S. Department of Energy under contract EY-76-S-03-070 and by the U.S. Army Research Office under contract DAAG-29-78-C-0011.

† Present address: Lawrence Livermore National Laboratory, Box 808 L-16, Livermore, California 94550.

‡ Applied Mathematics 217-50, California Institute of Technology, Pasadena, California 91125.

value problems (e.g., Meyer-Spasche and Keller [24], Schreiber and Keller [32]). However, the size of problems that can be solved by these methods is limited principally by the storage required for the Jacobian G_u, as well as the time required for the direct factorization of these Jacobians. For these reasons it is natural to consider multigrid methods, which, for a grid with N^2 mesh points, require $O(N^2)$ storage and approximately $O(N^2)$ arithmetic operations to solve to the accuracy of the truncation error.

Straightforward implementations of multigrid methods work well for nonlinear elliptic eigenvalue problems, but they fail near singular points. One type of difficulty near a limit point has been noted by several workers (e.g., Chan and Keller [9], Meis, Lehmann and Michael [23]). Assume that the solution branches on coarse and fine grids look like those in Fig. 1, say Γ_1 and Γ_M, respectively. Suppose we use "natural continuation" with λ as parameter on the coarse grid, and use the full approximation scheme (FAS) full multigrid algorithm while keeping λ fixed. Assume we start at λ_1 on the coarse grid, and wish to compute the fine grid solution at the limit point λ_f.

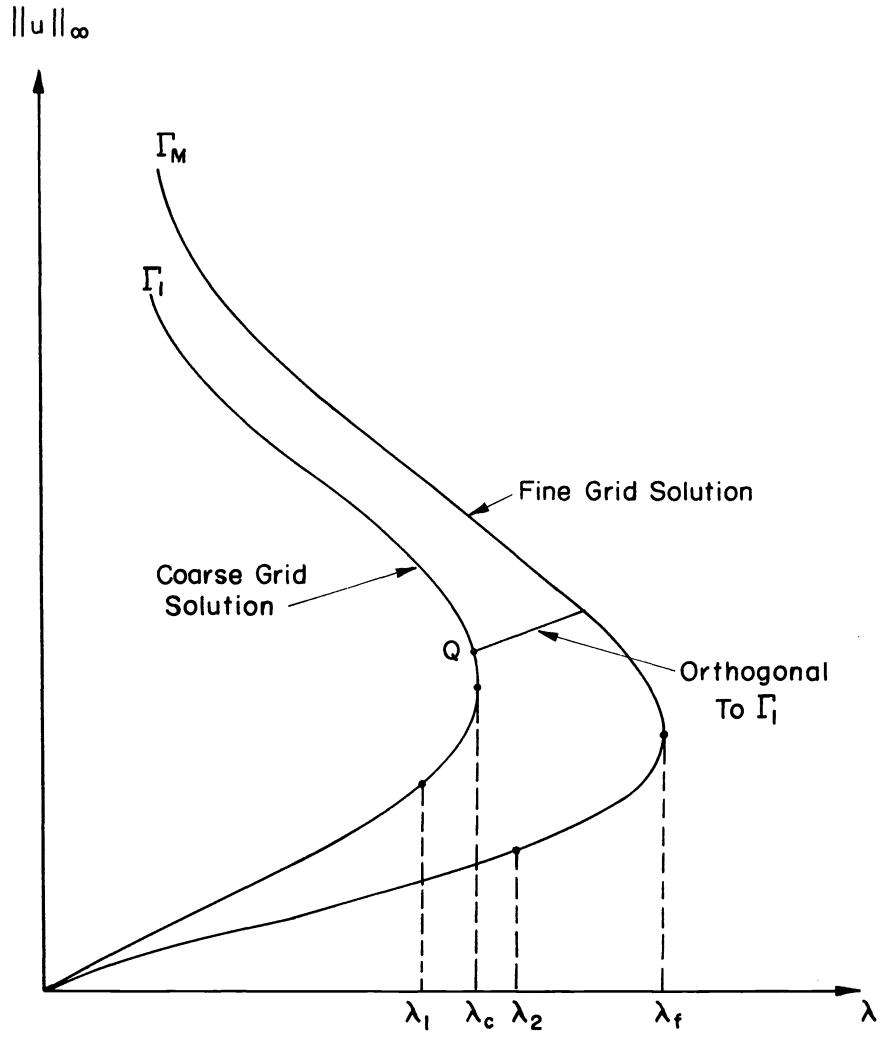


FIG. 1. *Limit points for different grids.*

As we approach the coarse grid limit point λ_c, there is a drastic slowing in the rate of convergence of the multigrid method, followed by divergence at λ_c. The divergence occurs first on the coarsest grid. (Although Chan and Keller [9] used the Cycle C "Correction Scheme" multigrid algorithm, we found the same results when we used the FAS full accommodative multigrid algorithm.) For points $\lambda_2 > \lambda_c$ there is no coarse grid solution, so we will not be able to approach λ_f without some remedy, such as deleting coarse grid(s).

Another (less well-known) type of difficulty occurs when the coarse and fine grid solution paths in Fig. 1 are interchanged. If we approach the fine grid limit point λ_f in the direction of increasing λ, the multigrid method converges, but the solution is not very accurate. However, if we do not know the location of λ_f in advance, we will try to compute solutions for λ_2 with $\lambda_f < \lambda_2 < \lambda_c$. For these λ_2 no fine grid solution exists. In practice, for λ_2 not "sufficiently close" to λ_f, the residuals of the fine grid solution cannot be reduced below a certain level, i.e., we encounter divergence on the fine grid.

Both configurations occur in practice, even using different approximations on the same problem. See §6, Table 1.

One remedy for the first type of difficulty was suggested by Chan and Keller [9]. They approximated the eigenfunction of the discrete differential operator that was causing the divergence, and then altered the interpolation operator and skipped a grid to prevent divergence.

All other methods do not fix the parameter λ during the multigrid iterations, i.e., they use different parameters λ^k on different grid levels k. One class of methods has been suggested by several authors, e.g., Meis, Lehmann and Michael [23], Stüben and Trottenberg [35]. Consider the model problem (1.2). In addition to the usual difference approximations, they impose an additional constraint on the finest grid, say:

$$u^M(P) \equiv u^M(1/2, 1/2) \equiv \|u^M\|_\infty = \text{constant},$$

where u at the center point is specified but λ^M remains unknown. This amounts to switching the role of unknown $u^M(P)$ and parameter λ^M. This parameter switching is one of the earliest techniques for treating limit points for single grid methods (see, e.g., Abbott [1]).

More specifically, Stüben and Trottenberg suggest modifying the multigrid smoothing step at one level as follows:

(a) Apply one (nonlinear) smoothing (relaxation) step to the actual approximation u^k, including the prescribed value at P.

(b) Multiply the relaxed approximation by a factor such that the constraint is satisfied afterwards.

(c) Compute a new value of λ^k for this level by using the difference equations together with a one-dimensional rootfinder (e.g., using a suitable average over all the equations).

Of course, this procedure will produce different λ^k at different grid levels if the constraint has the same value on all levels. At convergence, however, the parameters λ^k will be (approximately) the same if the FAS multigrid algorithm is used.

An objection to this procedure is, of course, its lack of generality. In more complicated problems (e.g., Lentini and Keller [22]) there is no clear "natural" parameter which can be used to eliminate limit points. Rheinboldt [29] circumvents this problem for single grid algorithms by treating the parameter as an additional unknown, and at each step appropriately selecting one of the unknowns as the continuation parameter.

A second class of methods has been proposed by Hackbusch [17], and by Bank and Chan [2]. Assume we are given the solution structure of Fig. 1. Having found a coarse grid solution point $Q \equiv (u, \lambda) \in R^{J+1}$ on Γ_1, these workers suggest finding a corresponding fine grid point on the line orthogonal (in the space R^{J+1}) to the coarse grid curve passing through Q. The projection of this orthogonal onto the $\lambda - \|u\|_\infty$ plane is shown in Fig. 1. Furthermore, Bank and Chan use additional diagonal shifts of Jacobians to assure that all matrices have the same number of negative eigenvalues. A third method, the generalized inverse iteration method, has been proposed in Mittlemann [25], and Mittlemann and Weber [26]. Their method determines the parameter λ^k on each grid as a generalized Rayleigh quotient.

Instead of using different parameter values on different grids, our method changes the structure of the solutions on different grids, by using the frozen tau technique of Brandt [7], [8]. In Fig. 1, our method in effect "moves" the coarse grid curve so that locally it is very close to the fine grid curve. Then the parameters λ needed on different grids are very near to each other, so we may use the same λ on all grids, and change λ (using pseudo-arc-length continuation) only on the coarsest grid. The multigrid and continuation methods interact through the full approximation scheme (eq. (3.3)). A special procedure (eq. (4.7)) produces accurate initial approximations.

Since we use the frozen tau method, the coarse grid equations produce approximations which, in general, more closely approximate the fine grid solutions than the solutions of the unmodified coarse grid equations. This is an advantage when computing problems with multiple solutions. Together with Mittlemann's method, our method is *robust* in the sense that the same algorithm can be used for both regular points and limit points, without significant loss of efficiency. That is, we need not detect the presence of limit points and try to switch algorithms. Our method requires a Jacobian only on the coarsest grid, so it is well-suited to large problems, especially those obtained from discretizing coupled systems of partial differential equations on fine grids [24], [31]. Finally, in our numerical results we observed that the rate of convergence (i.e., the rate of decrease of residual norms) did not degrade near limit points.

Throughout this paper we consider the following model problem, which will be used to illustrate the methods and to furnish numerical examples.

A two-dimensional stationary reaction-diffusion problem is formulated as:

$$Lu(\lambda, \alpha) \equiv \Delta u + \lambda \exp(u/(1 + \alpha u)) = 0 \quad \text{in } \Omega, \tag{1.2a}$$

$$u = 0 \quad \text{on } \partial\Omega. \tag{1.2b}$$

We take $\Omega \equiv$ the unit square: $[0, 1]\times[0, 1]$. In the notation of (1.1), $\boldsymbol{\lambda} = (\lambda, \alpha)$.

The solution of this problem has interesting geometric features (see Fig. 2). There exists a critical value of α, say α^*, for which: i) $u(\cdot, \lambda, \alpha)$ has two simple (quadratic) limit points in λ for fixed $\alpha < \alpha^*$; ii) $u(\cdot, \lambda, \alpha)$ has no limit points for fixed $\alpha > \alpha^*$; iii) $u(\cdot, \lambda, \alpha^*)$ has one (cubic) limit point. In case i), in a sufficiently small neighborhood of each limit point P_i, the solution curve lies on one side of the (vertical) tangent line at P_i, while in case iii) the solution curve lies on both sides of the tangent line at the limit point. See Spence and Werner [34] for a discussion of methods to compute α^*.

In § 2 of this paper we review arc-length continuation methods. Section 3 summarizes the accommodative FAS full multigrid algorithm for a fixed parameter value. In §4 we review the frozen tau algorithm and then describe our method. We also describe implementation details such as the use of high order splines for interpolation. An outline of a convergence proof, which is based on methods of Hackbusch [16], [17] is given in § 5. In § 6 we give numerical results for second, fourth and sixth order approximations to our model problem (1.2).

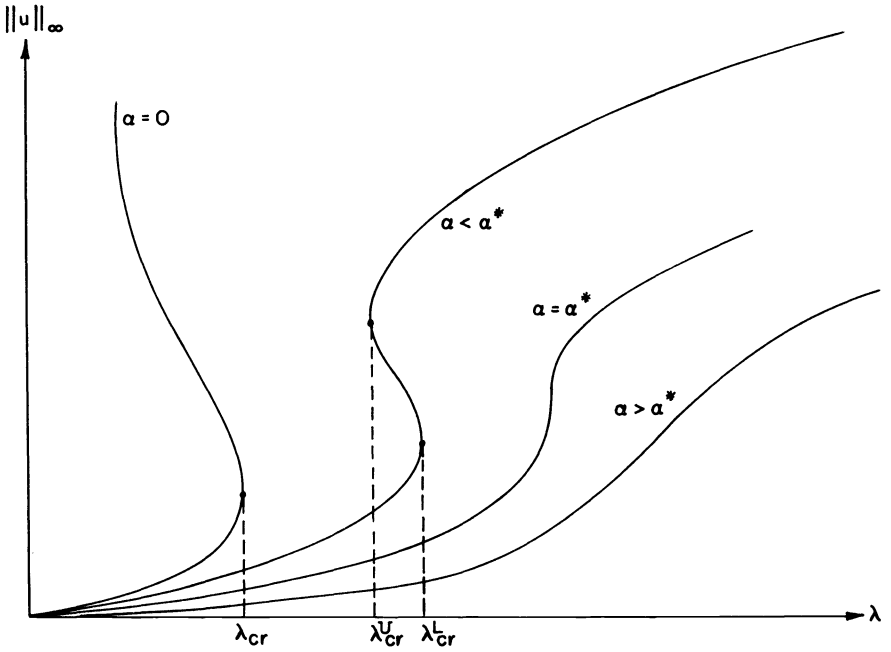


FIG. 2. *Solution curves of reaction-diffusion equation* (1.2) *for various values of parameter* α.

2. Continuation methods. In this section we briefly review continuation methods for computing a family or path of solutions of (1.1), without using multigrid methods.

2.1. Newton's method. Given a value of λ and an initial guess u^0 of the solution $u(\lambda)$, we perform the following steps repeatedly until $\|\delta u^i\| < \varepsilon$ is satisfied:

$$G_u(u^i, \lambda)\delta u^i = -G(u^i, \lambda), \tag{2.1}$$

$$u^{i+1} = u^i + \delta u^i. \tag{2.2}$$

This procedure will generally converge quadratically when it does converge. However, as is well known, it can fail to converge when the initial guess is not sufficiently close to a solution, or if a solution does not exist for the given λ value.

2.2. Natural continuation. To overcome the former difficulty, we can start at a known solution (u_0, λ_0) on the solution path and use it as an initial guess for a Newton-type iteration to find the solution for a neighboring point on the solution path with λ close to λ_0. The procedure is then repeated. We can improve on this by computing the derivative u_λ at a known solution and use it to get a better initial guess for the next value of λ in a predictor-corrector fashion. We call this a *natural* continuation procedure because it corresponds to parametrizing the solution path by λ, the naturally occurring parameter. A specific form of this is the well-known:

Euler-Newton continuation procedure. Given a known solution (u_0, λ_0), compute the solutions at nearby values of λ as follows:

1. First compute the derivative u_λ at (u_0, λ_0) from

$$G_u u_\lambda = -G_\lambda.$$

2. Perform an Euler predictor step:

$$u^0 = u_0 + u_\lambda(\lambda - \lambda_0).$$

3. Use u^0 as initial guess in Newton's method. Repeat

$$G_u(u^{i+1} - u^i) = -G(u^i, \lambda),$$

until convergence.

4. Use $(u(\lambda), \lambda)$ as the new (u_0, λ_0) and go to Step 1.

Unfortunately, this procedure needs some modification in order to handle general nonlinear systems because of the possibility of nonunique solutions. The nonuniqueness usually manifests itself in the form of existence of "singular" points where the Jacobian, G_u, is singular. The most common (i.e., generic) singular point is a simple limit point. (Another type of singular point, a bifurcation point, at which $G_\lambda \in R(G_u)$, will not be considered in this paper.) A natural continuation procedure will encounter two difficulties at limit points. First, since G_u is singular at these points, Newton's method (unaltered) will at best be linearly convergent, making it much more costly to compute the solution. Second, and more serious, the Euler-Newton procedure may lead us to a value of λ (such as λ_2 for the coarse grid in Fig. 1) for which no nearby solution exists, and the iterations will generally fail to converge.

2.3. Pseudo-arc-length continuation. In the pseudo-arc-length approach (Keller [19]), these difficulties are overcome by not parametrizing the solution u by λ. Instead, we parametrize the solution branches using a pseudo-arc-length parameter s, and specify how far along the current solution branch we want to try to march in s. This requires us to add an "arc-length" equation to our system of equations.

To be more specific, we let s be the arc-length-like parameter, and treat $u(s)$ and $\lambda(s)$ as functions of s. We then replace the Euler-Newton continuation procedure by the following:

Pseudo-arc-length Euler-Newton continuation procedure [19]: Assume given a solution $(u(s_0), \lambda(s_0))$.

1. Compute a tangent

$$(\dot{u}_0, \dot{\lambda}_0) \equiv (\dot{u}(s_0), \dot{\lambda}(s_0))$$

to the solution branch (where the dots denote differentiation with respect to s) satisfying:

$$G_u\dot{u}_0 + \dot{\lambda}_0 G_\lambda = 0, \tag{2.3}$$

$$\|\dot{u}\|^2 + |\dot{\lambda}|^2 - 1 = 0. \tag{2.4}$$

Equation (2.3) is obtained by differentiating (1.1) with respect to s, and (2.4) is an arc-length condition. The norm $\|\cdot\|$ is a discrete vector norm which approximates the continuous L_2 norm for integrals. For example, in m dimensions,

$$\|u\|^2 = h^m \sum_i u_i^2,$$

where u_i are the vector components. The equations (2.3)-(2.4) are easily solved [20] as follows.

First solve the system of linear equations

$$G_u\phi = -G_\lambda; \tag{2.5a}$$

then determine $\dot{u}(s_0)$ and $\dot{\lambda}(s_0)$ by

$$\dot{\lambda}_0 = \pm(1 + \|\phi\|^2)^{-1/2}, \tag{2.5b}$$

$$\dot{u}_0 = \dot{\lambda}_0\phi. \tag{2.5c}$$

Here the sign in (2.5b) determines the direction in which we traverse the solution path. If two solutions (u_{-1}, λ_{-1}) and (u_0, λ_0) have been computed, then we choose the sign such that:

$$\langle \dot{u}_0, u_0 - u_{-1} \rangle + \dot{\lambda}_0(\lambda_0 - \lambda_{-1}) > 0,$$

and the continuation proceeds in the direction from (u_{-1}, λ_{-1}) to (u_0, λ_0). Here $\langle\ ,\ \rangle$ is an inner product which induces the norm in (2.4).

2. Select a step size $s - s_0$. (See § 4.5.) Then take an Euler step of length $s - s_0$ along the tangent:

$$u^0 = u_0 + (s - s_0)\dot{u}_0, \tag{2.6a}$$

$$\lambda^0 = \lambda_0 + (s - s_0)\dot{\lambda}_0. \tag{2.6b}$$

In general (u^0, λ^0) will not be a solution of (1.1).

3. We now use the *pseudo-arc-length condition* to return to the solution branch. We require our new solution point to satisfy the equations

$$G(u(s), \lambda(s)) = 0, \tag{2.7}$$

$$N(u(s), \lambda(s), s) = 0, \tag{2.8a}$$

where

$$N(u(s), \lambda(s), s) \equiv \langle \dot{u}_0, (u(s) - u(s_0)) \rangle + \dot{\lambda}_0(\lambda(s) - \lambda(s_0)) - (s - s_0) = 0. \tag{2.8b}$$

Equation (2.8) is a linearization of the arc-length condition (2.4) and is used because it contains $(u(s), \lambda(s))$ and not $(\dot{u}(s), \dot{\lambda}(s))$. Equation (2.8) forces the new solution to lie on a hyperplane perpendicular to the tangent vector to the solution curve at s_0, and at a distance $|s - s_0|$ from it.[1] We solve the coupled system (2.7)-(2.8) for $u(s)$ and $\lambda(s)$ by using Newton's method with initial guess (u^0, λ^0). This requires solving the following system at each iteration:

$$A\begin{bmatrix} \delta u^i \\ \delta \lambda^i \end{bmatrix} \equiv \begin{bmatrix} G_u & G_\lambda \\ N_u & N_\lambda \end{bmatrix}\begin{bmatrix} \delta u^i \\ \delta \lambda^i \end{bmatrix} = -\begin{bmatrix} G \\ N \end{bmatrix}. \tag{2.9}$$

The quantities G, N and their derivatives are all evaluated at $(u^i(s), \lambda^i(s))$.

It can be shown [19] that at simple limit points, the linear system in (2.9) is nonsingular, and so Newton's method for the coupled system (2.7)-(2.8) is well-defined. Hence simple limit points present no computational problems and even quadratic convergence is achievable.

In order to solve the linear system in (2.9) by direct or iterative methods, several approaches are possible. One way is to perform Gaussian elimination on the inflated matrix A, with some form of pivoting to ensure stability. But this approach completely ignores the sparse structure which is usually found in the Jacobian G_u arising from discretizations of nonlinear elliptic eigenvalue problems. In order to take advantage of the structure in the Jacobian, Keller [19], [21] used the following bordering algorithm:

Solve

$$G_u y = G_\lambda \tag{2.10}$$

[1] Another choice is: $N = \|u - u_0\|^2 + |\lambda - \lambda_0|^2 - |s - s_0| = 0$. This forces the solution to lie on a sphere of radius $|s - s_0|$ about the previous solution.

and

$$G_u z = -G. \tag{2.11}$$

Set

$$\delta\lambda = (N + N_u^T z)/(N_\lambda - N_u^T y), \tag{2.12}$$

$$\delta u = z + \delta\lambda y. \tag{2.13}$$

Now only systems with the coefficient matrix G_u have to be solved, so structures in G_u can be exploited. Moreover, only one factorization of G_u is needed. It has been shown (Szeto [34]), and observed in practice, that even when G_u becomes singular, this bordering algorithm produces iterates that converge quadratically at simple limit points. Some loss in accuracy (cancellation errors in (2.13)) are to be expected, however [21]. We discuss this further in § 4.5.

But, as mentioned before, a major disadvantage of these continuation methods for many problems is the need to store and factor large Jacobians, G_u. For that reason we consider multigrid methods.

3. Multigrid methods. In this section we shall give a brief description of the multigrid method we use for a fixed value of the parameter vector λ. It is assumed (in this section only) that (u, λ) is not a limit point (or bifurcation point). In the next section we shall give modifications necessary for continuation. For a survey and more complete description of multigrid methods, see Brandt [5], Brandt and Dinar [6], Brandt [8], Stüben and Trottenberg [35], and other papers in the latter volume. We use *accommodative, full* multigrid with the *full approximation scheme* (FAS). Unlike Chan and Keller [9] we do not use the "Cycle C" algorithm.

We consider an elliptic partial differential equation defined on a region Ω with boundary $\partial\Omega$:

$$\begin{aligned} LU &= F \quad \text{on } \Omega, \\ U &= 0 \quad \text{on } \partial\Omega. \end{aligned} \tag{3.1}$$

(If U or F depend on a parameter λ, fix the parameter.) Then we construct a hierarchy of grids $\Omega^1, \Omega^2, \cdots, \Omega^M$, all approximating the domain Ω, with corresponding mesh sizes $h_1 > h_2 > \cdots > h_M$. The discrete approximation on grid Ω^k is written as

$$\begin{aligned} L^k U^k(x) &= F^k \quad \text{on } \Omega^k, \\ U^k &= 0 \quad \text{on } \partial\Omega^k, \end{aligned} \tag{3.2}$$

where $\partial\Omega^k$ approximates the boundary $\partial\Omega$. We wish to solve this discrete problem on the finest grid, Ω^M.

Figure 3, adapted from [6], gives a flow chart of the FAS full multigrid (FMG) algorithm. In the FAS method, each U^k is an approximation to the exact solution U. The FAS method (as opposed to the correction scheme) is particularly suited to the solution of nonlinear problems. When properly employed, it eliminates the need for large Jacobians.

The FMG method can be divided into two phases: the "initialization" phase, in which we start with a solution obtained in some manner on the coarsest grid Ω^1, and "bootstrap" our way up to a first approximation u^M, on the finest grid Ω^M. (For continuation problems it is important to have good first approximations on Ω^M.) The second phase improves the first approximation on grid Ω^M. This phase is common to all multigrid algorithms. In the flow chart, these phases are combined by replacing the

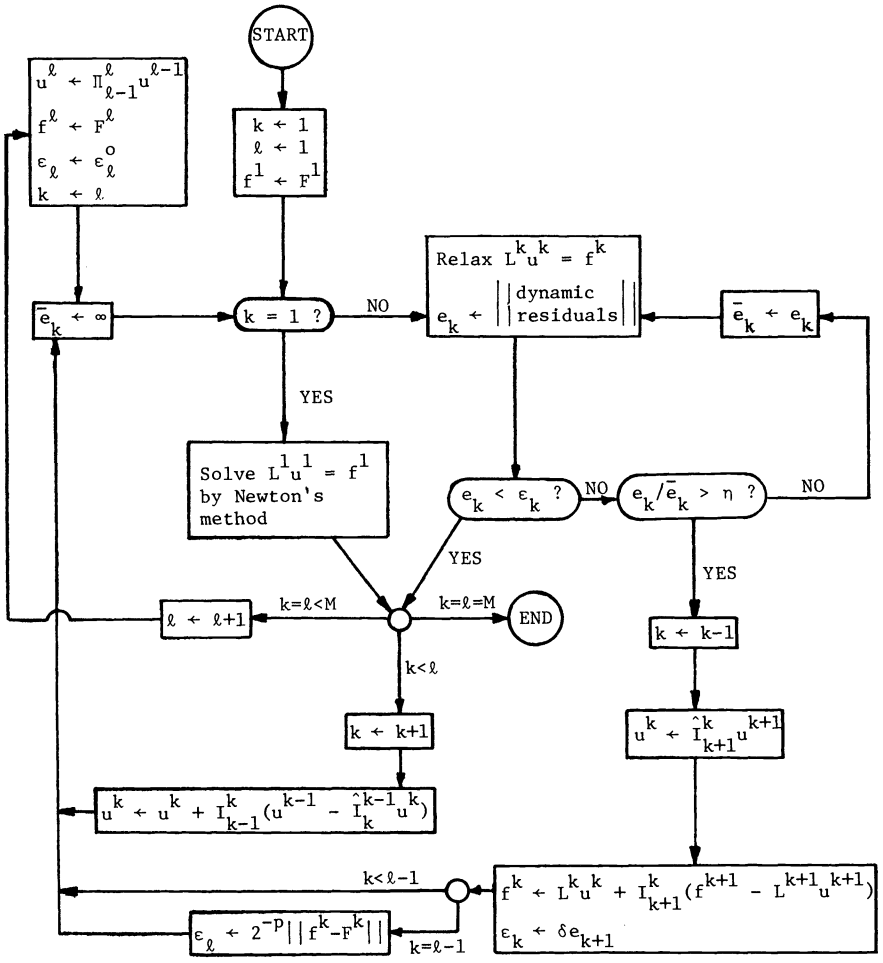


FIG. 3. *Accommodative FAS full multigrid algorithm. The notation is explained in the text.*

maximum level M with a level l, which denotes the highest grid level seen up to this time. The index l increases from 1 to M in phase one, and is fixed at M in phase two.

It is somewhat more natural to view the FMG method recursively; such a formulation is given in Hackbusch [16].

We now explain the quantities in the flow chart. The current grid level is denoted by k. In the FAS multigrid method, equations (3.2) are solved only at the highest current level l. The current approximation to U^l at this level is denoted by u^l. In the process, eq. (3.2) on coarser grids ($k < l$) are modified by changing their right-hand sides. The modified right-hand sides of the kth level equations are given by

$$f^k = L^k(\hat{I}^k_{k+1}u^{k+1}) + I^k_{k+1}(f^{k+1} - L^{k+1}u^{k+1}), \tag{3.3}$$

and depend on u^{k+1}, the current approximation on the next finer level. The solutions to these modified equations are denoted by $\tilde{U}^k$, and their computed approximations by u^k. Thus, the FAS multigrid method attempts to solve the equations

$$L^k\tilde{U}^k = f^k \tag{3.4}$$

at each level $k < l$, not (3.2).

The order of accuracy of the difference approximation is p. The norm of the dynamic residual on level k is

$$e_k = \|f^k - L^k u^k\|,$$

and its tolerance is ε_k. The interpolation operator I_{k-1}^k interpolates *corrections* from a "coarse" grid to the next finer grid. The interpolation operator Π_{l-1}^l interpolates *solutions* from a coarse grid to the next finer grid; it is used only in phase one, and frequently must be of higher order than I_{k-1}^k. The operators I_{k+1}^k and $\hat{I}_{k+1}^k$ are projection operators from a "fine" grid to the next coarser grid. The former projects residuals and the latter projects approximate solutions.

The parameter η controls switching to a coarser grid. When the norm e_k of the residual for this iteration of the relaxation process on grid k exceeds η times the corresponding residual norm $\bar{e}_k$ of the preceding iteration (i.e., the convergence rate becomes too slow), we switch to grid $k-1$. The parameter δ controls the accuracy with which we solve the equations on grid $k-1$ before using the result to correct the solution on grid k. The use of η and δ characterizes an *accommodative* multigrid algorithm, where the number of relaxation sweeps on a grid is determined dynamically by the progress of the iterations. This is in contrast to *fixed* algorithms, which perform a fixed number of relaxation sweeps on grid k before going to grid $k-1$, and then a fixed number of sweeps after returning to grid k. Accommodative methods are more robust, especially for nonlinear problems.

On the coarsest grid we do not use a relaxation method. On the upper branches of multiple solutions (Figs. 1 and 2) the matrix G_u becomes indefinite. This leads to divergence on the coarsest grid if a standard relaxation method is used ([9], [35]). Hence we use the full Newton method on the coarse grid. In § 4.2 we will see further reasons for using Newton's method here.

We defer until § 4.5 further details of our multigrid implementation.

4. Description of the algorithm. In this section we describe our algorithm. This description will involve Brandt's frozen tau method, and some modifications to the methods described in §2 and 3. Our goals in designing a continuation method are to do as much as possible on the coarse grid, and to avoid the need for large order Jacobians.

4.1. The frozen tau technique. We first describe the "dual view" (Brandt [8], Stüben and Trottenberg [35]) of the multigrid algorithm, and the relative truncation error. The local truncation error of the kth level approximation is

$$\tau^k \equiv L^k(\hat{I}^k U) - F^k = L^k(\hat{I}^k U) - I^k(LU),$$

where $\hat{I}^k$ projects continuous solutions onto the kth grid, and I^k (possibly different from $\hat{I}^k$) does the same for right-hand sides. This is obtained by substituting the solution U of the differential equation into the difference equations (3.2).

To estimate τ^k, we replace U by the "converged" approximate solution u^m, $k < m \leqq M$, and replace the projections I^k and $\hat{I}^k$ by I_m^k and $\hat{I}_m^k$, respectively. We then obtain the (m, k)-*relative truncation error*, or *fine-to-coarse defect correction*

$$\tau_m^k = L^k(\hat{I}_m^k u^m) - I_m^k(L^m u^m), \tag{4.1}$$

where

$$I_m^k = I_{k+1}^k I_{k+2}^{k+1} \cdots I_m^{m-1},$$

and similarly for $\hat{I}_m^k$. Using (3.4) with $k = m$, this can be rewritten

$$L^k(\hat{I}_m^k u^m) = I_m^k f^m + \tau_m^k. \tag{4.2}$$

The relative truncation error is related to the local truncation error by

$$\tau_m^k \approx \tau^k - \tau^m, \qquad 1 \leqq k < m \leqq M,$$

where $\approx$ indicates equality of the leading terms in an asymptotic expansion in powers of h_k. (Here we have assumed that the global error can be expanded in a power series in h_k whose coefficients are independent of k.)

This leads to the "dual" interpretation of the multigrid method. Instead of regarding the coarse grid as a device for accelerating convergence of the fine grid equations, we can view the fine grid as a device for calculating the correction τ_m^k to the coarse grid equations. In other words, if grid k is a coarsening of grid M, and if $\hat{I}_M^k$ is the straight injection operator, then τ_M^k is that quantity which has to be added to a (modified) right-hand side, $I_M^k f^M = I_M^k F^M$, to obtain values of the fine grid solution u^M by solving the coarse grid equations. That is, at convergence, the coarse grid solutions are simply projections of the finest grid solution. This would not be true in general, if one solved problems (3.2) independently, without FAS.

We now present a modified version of Brandt's [7], [8] frozen tau technique for continuation. Suppose we have computed a multigrid solution for parameter λ_0 by the method of § 3, and we wish to compute an FAS full multigrid solution at the "nearby" parameter λ_1. Assume we have an initial approximation to the solution on the coarsest grid at λ_1. We can make the coarse grid solution equation "appear" like the (as yet unknown) fine grid solution at λ_1 as follows.

We first determine the relative truncation errors $\tau_M^k(\lambda_0)$, $k = 1, 2, \cdots, M-1$. If $F^k \equiv 0$, as in our model problem (1.2), τ_M^k is, at convergence, simply the modified right-hand side f^k. Otherwise, from (3.3) and induction, τ_M^k is given by

$$\tau_M^k = f^k - I_M^k f^M. \tag{4.3}$$

Before the computation even begins at λ_1, we add $\tau_M^k(\lambda_0)$ to the right sides of the λ_1 equations, for $k = 1, 2, \cdots, M-1$. If λ_1 is close to λ_0, this makes the coarse grid equations at λ_1 locally look like the fine grid "corrected" equations (4.2) with $m = M$. The effect in Fig. 1 is approximately and locally to shift the coarsest grid curve (and all "intermediate" grid curves) close to the fine grid curve. Away from the point where we are computing, the curves corresponding to different grids are not necessarily close to each other.

Of course, this procedure initially ignores the error in $\tau_M^k(\lambda_1)$ due to the change in λ. However, the error so produced will not depend on the high-frequency components, but only on the changes in those components from their values at λ_0. Normally these changes are small compared with the components themselves [7], [8]. Furthermore, the $\tau_M^k(\lambda_0)$ terms are implicitly improved by the multigrid iterations at λ_1, as will be explained in §4.3. More accuracy could be obtained initially by extrapolation in λ using, say, $\tau_M^k(\lambda_0)$ and $\tau_M^k(\lambda_0 - \delta\lambda)$ to better approximate $\tau_M^k(\lambda_1)$.

4.2. Combining multigrid and continuation methods. Despite this improvement, it is clear that we still must vary λ_1 during the multigrid iteration. As we observed in § 1, many workers use different values of λ on different grids. But our shifting of the coarse grid curves allows us to use a very simple algorithm for changing λ_1. We propose correcting λ_1 during the multigrid iteration *only on the coarsest grid.*

We need a method to produce a first approximation u^1 on the coarsest grid at λ_1, and values u^1 and λ_1 for the coarsest-grid correction at λ_1. The pseudo-arc-length continuation method of § 2 will supply both, while being insensitive to limit points. This method interacts with the FAS-FMG algorithm in a natural manner, through the right-hand sides of the coarsest grid equations.

We will now change our notation slightly, and refer to the old and new parameters as $\lambda(s_0)$ and $\lambda(s_1)$, not λ_0 and λ_1. The old parameter $\lambda(s_0)$ is fixed. We allow $\lambda(s_1)$ to vary, but it will be the same on all s_1-grids.

4.3. Algorithm for continuing through limit points. We now summarize the steps of our multigrid continuation algorithm. Then we will explain each of these steps. This algorithm will follow solution curves as they pass through limit points, but in general none of the computed solution points will coincide with any limit point. We will show how to accurately locate limit points in the next subsection.

1. Assume given an initial parameter value λ_0, and an initial coarse grid approximation u^1 to a solution of (1.1) which is not "too close" to a limit point. Let $s_0 = 0$ and define $\lambda(s_0) \equiv \lambda_0$, $u^1(s_0) \equiv u^1$.

2. Use Newton's method (2.1)-(2.2) to obtain an improved solution on the coarsest grid. Note λ_0 is kept fixed.

3. Perform the accommodative FAS full multigrid algorithm, as described in § 3 and Fig. 3. Use a method such as Gauss-Seidel-Newton to relax (smooth) on grids $2, 3, \cdots, M$, and use Newton's method (2.1)-(2.2) to solve on the coarsest grid. The parameter λ_0 is still fixed.

4. When the multigrid algorithm has converged, project the solution from the finest grid to all coarser grids,

$$u^k(s_0) = \hat{I}^k_{k+1} u^{k+1}(s_0), \quad k = M-1, M-2, \cdots, 1. \tag{4.4}$$

5. If $F^k \neq 0$, determine the relative truncation errors $\tau^k_M(s_0)$. First compute

$$\tau^k_{k+1} = f^k - I^k_{k+1} f^{k+1}, \quad k = 1, 2, \cdots, M-1,$$

and overwrite on f^k. Then compute

$$\tau^k_M = \tau^k_{k+1} + I^k_{k+1} \tau^{k+1}_M, \quad k = M-2, M-3, \cdots, 1,$$

and overwrite on f^k.

6. Choose a step length Δs. (See §4.5.) Let $s_1 = s_0 + \Delta s$.

7. Apply the frozen tau technique by adding $\tau^k_M(s_0)$ to the right sides $f^k(s_1) \equiv F^k(s_1)$ of the grid equations (3.4) at levels $k = 1, 2, \ldots, M-1$.

8. Perform the Euler step (2.5)-(2.6) to obtain a first approximation $(u^1(s_1), \lambda(s_1))$ on the coarsest grid at s_1.

9. Use the pseudo-arc-length Newton method (2.10)-(2.13) to improve this approximation on the coarsest grid. The "right-hand side" term $F^1(s_1)$ in

$$G(u^1(s_1), \lambda(s_1)) \equiv L^1(s_1) u^1(s_1) - F^1(s_1), \tag{4.5}$$

which appears in (2.9) or (2.11), was replaced in step 7 by

$$f^1(s_1) \equiv F^1(s_1) + \tau^1_M(s_0). \tag{4.6}$$

10. Perform the multigrid iteration to obtain $\lambda(s_1)$ and $u^M(s_1)$. This is the same as step 3, except: (a) The right sides F^k will have already been modified by step 7; (b) Initial approximations u^k on grids $2, 3, \cdots, M$ are obtained by a method to be described; (c) We use the pseudo-arc-length Newton method (2.10)-(2.13) to correct

on the coarsest grid. The right side $F^1(s_1)$ will have been replaced by the usual FAS right-hand side in (3.3). We obtain not only a new $u^1(s_1)$, but also a new $\lambda(s_1)$.

This procedure yields the solution for two parameter values s_0, s_1. To compute the solution at further values $s_2, s_3, \cdots$, repeat steps 4 to 10, but replace s_i by s_{i+1}. In practice we reuse the storage for s_{i-1} when we start computing at s_{i+1}. Thus only two sets of grids are needed, each set consisting of values of u and f on all levels. Hence the total storage is approximately eight times the number of grid points in Ω^M, times the number of differential equations, plus a small amount for one coarsest-grid Jacobian.

We will now expand on some of these steps.

In steps 2 and 3 we fix λ because we do not have past information to enable us to use pseudo-arc-length continuation. Step 4, projection onto coarser grids, is not strictly necessary, since before it is applied, the FAS method ensures that $u^1(s_0)$ is approximately equal to the right-hand side of (4.4).

Step 5 can be omitted when there are no inhomogeneous terms in the differential equations (e.g., problem (1.2)). Then step 7 can be slightly simplified by *copying* $\tau_M^k(s_0)$ to the right sides of the s_1-grids rather than adding.

For the Euler step 8 and the Newton steps 2 and 9 we must calculate the Jacobian G_u, but this is on the coarsest grid, so there is no storage problem.

Just as in the ordinary pseudo-arc-length method, the Newton step 9 brings us back to the solution curve. Since we have added $\tau_M^1(s_0)$ to the right side of the coarse grid equation, we are brought back (approximately) to the finest grid curve, rather than the coarse grid curve at s_1.

During the multigrid iterations at the new parameter s_1, we return to the coarsest grid for corrections. Just before doing so, we replace the right-hand-side term $f^1(s_1)$ of (4.6) by a new $f^1(s_1)$, as prescribed by the FAS-multigrid method in (3.3). We then use pseudo-arc-length continuation (2.10)–(2.13) to obtain a new $\lambda(s_1)$ and $u^1(s_1)$. The new $\lambda(s_1)$ is used on all finer grid levels until the next coarsest-grid correction.

At first glance it appears that λ_1, since it is changed only on the coarsest grid, incorporates information only from the coarsest grid and the frozen tau term $\tau_M^1(s_0)$. But as the full multigrid iteration at s_1 progresses, λ_1 actually incorporates information about all other grids and all other terms $\tau_M^k(s_0)$, $1 < k < M - 1$, as well. For example, when the finest grid seen so far at s_1 is 2 (i.e., $l = 2$ in Fig. 3), the coarsest-grid correction involves $f^2(s_1)$, by (3.3). But $f^2(s_1)$ has incorporated the term $\tau_M^2(s_0)$. Thus the new λ_1 will also incorporate this term. Meanwhile, the $\tau_M^k(s_0)$ are in effect being replaced by "new" $\tau_M^k(s_1)$. When the iteration at s_1 returns to the coarsest grid for the first time in phase 2 (i.e., when $k = 1$ for the first time after $l = M$ in Fig. 3), the $\tau_M^k(s_0)$ in effect will have been completely replaced by new and better $\tau_M^k(s_1)$. This implicit updating of the frozen tau terms is a consequence of eq. (3.3) and step 7 of our algorithm as stated. No further computations beyond steps 1–10 are required. The frozen tau terms are *explicitly* formed only in Step 5.

To obtain initial approximations $u^k(s_1)$ for grids other than the coarsest, Brandt [8] and Hackbusch [17] suggest starting with the old (s_0) solution at the same level, and correcting it by the difference between the old and new solutions at the next coarser level:

$$u^k(s_1) = u^k(s_0) + \Pi_{k-1}^k(u^{k-1}(s_1) - u^{k-1}(s_0)), \tag{4.7}$$

where Π is the high-order interpolation described in § 3.2. We found that this works well, and it makes possible the error analysis in § 5.

4.4. Locating limit points. The algorithm of the last subsection gave a method for continuing through limit points. In this section we describe how to accurately locate limit points.

If the Jacobian G_u is available on the finest grid, then Keller [20] suggests locating zeros of $\dot{\lambda}^M(s)$. (Here the superscript M means that the the derivative is computed on the finest grid.) If we are performing pseudo-arc-length continuation on a single grid (i.e., $M = 1$), then this derivative is free, since we solve equations (2.5) (with superscripts M added) anyway. But if $M > 1$ then $\dot{\lambda}^M(s)$ requires the Jacobian *on the finest grid*, which we wish to avoid.

A first approach is to use our algorithm of the preceding subsection, and compute the root of $\dot{\lambda}^1(s)$ on the coarse grid where this derivative is computed anyway. Unfortunately, this does not work very well. Although this derivative does become small in the neighborhood of a limit point $P = (u^M(s^*), \lambda(s^*))$, it has so far been too inaccurate in locating P to many decimal places. For example, we used the method of Keller [20] for problem (1.2) with $\alpha = 0$ on a single grid Ω^M, $M = 1$. We located a turning point P by finding s^* for which $\dot{\lambda}^M(s^*) \approx 10^{-12}$. We then repeated the calculation with multigrid (using a method about to be described), in which the finest grid Ω^M had the same mesh size as before. We found a limit point $\tilde{P}$, which was extremely close to P (i.e., the parameters λ agreed to more than 12 digits). But at $\tilde{P}$ we had only $\dot{\lambda}^1(s) \approx 10^{-3}$.

The next approach is to try to imitate the frozen tau method. To do this we would need to replace

$$G(u^M(s_0), \lambda(s_0)) = 0$$

by

$$G(u^1(s_0), \lambda(s_0)) + \tau_M^1(u^M(s_0), \lambda(s_0)) = 0. \tag{4.8}$$

Recall that we obtain (2.3) by differentiating (1.1) with respect to s. If we differentiate (4.8) in the same way, we obtain derivatives of τ with respect to λ and u. The former could be approximated by difference quotients, but it is difficult to find the Jacobian $(\tau_M^1)_u$ without knowing the explicit form of τ_M^1.

Our approach, then is to use a derivative-free method. The algorithm proceeds in two parts. We first apply the method of the preceding section to obtain points P_1, P_2 lying on a solution curve and on either side of the limit point. Then we restart at one of the points, say $P_1 = (u^M(s_1), \lambda(s_1))$, and use $\Delta s \equiv s - s_1$ as the independent variable in a one-dimensional derivative-free optimizer to find an extremum of $\lambda(s)$. Each "outer" iteration of the optimizer requires a complete multigrid solution $(u^M(s), \lambda(s))$ on the finest grid ("inner iteration") with Δs chosen by the optimizer.

We chose the optimizer FMIN, written by R. Brent and appearing in Forsythe, Malcolm and Moler [14]. FMIN uses a combination of golden section search and successive parabolic interpolation. Typically about twelve outer iterations are required to locate the limit point to machine precision.

The only disadvantage of this method (compared with that of Keller [20]) is that it cannot locate limit points which are also inflection points (e.g., on the curve in Fig. 2 for $\alpha = \alpha^*$). However, such points are very rare.

4.5. Implementation details. In this subsection we illustrate our algorithm by applying it to the model reaction-diffusion problem (1.2). In contrast to the description of the previous subsections, most of these implementation details (difference approxi-

mations, smoothing algorithm, etc.) are problem-dependent. They are included to fully characterize the numerical results of § 6.

For simplicity, we consider a square region. This is not, however, a restriction on our method. Removing this restriction would only change some of the implementation details, especially the interpolations. Similarly, our method is not restricted to Dirichlet conditions on $\partial\Omega$.

Let $N_1 > 0$ be the number of coarse grid intervals, and $h_1 = 1/N_1$ be the mesh size on the coarsest grid. The finer mesh sizes are given by letting $N_k = 2N_{k-1}$, and $h_k = h_{k-1}/2$, $k = 2, 3, \cdots, M$. Grid Ω^k is then defined as

$$\Omega^k = \{(x_i^k, y_j^k): x_i^k = ih_k, y_j^k = jh_k, i, j = 0, 1, \cdots, N_k\}, \tag{4.9}$$

for $k = 1, 2, \cdots, M$. We let U_{ij}^k be an approximation to the exact solution $U(x_i^k, y_j^k)$, and u_{ij}^k be the approximation to U_{ij}^k computed by the FAS method. We shall omit the superscript k on U, x, and y whenever possible. For our Dirichlet problem, the difference equations, the right-hand sides $F^k \equiv 0$ and f^k, and the residuals $f^k - L^k u^k$ are defined only on interior grid points, i.e., those for which neither i nor j is 0 or N_k. Naturally, we set the discrete solution to zero at the boundary points.

We use several difference approximations. The second-order approximation to (1.2) is obtained by replacing the Laplacian Δ by the usual five-point approximation Δ_h^5:

$$\Delta_h^5 U_{ij} + \lambda \exp(U_{ij}/(1 + \alpha U_{ij})) = 0. \tag{4.10}$$

Since this approximation is relatively inaccurate, we also consider two fourth order approximations, both obtained from Collatz's Mehrstellen Verfahren. The local truncation error for the nine-point approximation to the Laplacian is

$$\Delta_h^9 U - (\Delta U) = h^2 \Delta(\Delta U)/12 + O(h^4). \tag{4.11}$$

We use the differential equation (1.2) to replace (ΔU) on both left and right sides of (4.11). Then we replace the remaining Δ operator on the right side by the 5-point operator Δ_h^5, to obtain the $O(h^4)$ accurate approximation:

$$\Delta_h^9 U_{ij} + \lambda(I + (h^2/12)\Delta_h^5) \exp(U_{ij}/(1 + \alpha U_{ij})) = 0. \tag{4.12}$$

The second fourth order approximation proceeds as before, up to the last step. But we do not replace the outer Δ operator on the right side of (4.11) by Δ_h^5. Instead we analytically differentiate. In addition to the obvious terms Δu, we obtain terms in U_x^2 and U_y^2. We now replace ΔU by the five-point operator, and U_x and U_y by centered differences $D_{0x}U$ and $D_{0y}U$, respectively. The result is another $O(h^4)$ approximation:

$$\Delta_h^9 U + \lambda \exp(U/\beta)[1 + h^2(\beta^2 \Delta_h^5 U + (1 - 2\alpha\beta)(D_{0x}^2 U + D_{0y}^2 U))/12\beta^4] = 0. \tag{4.13}$$

Here we have defined $\beta = 1 + \alpha U$; and we have omitted subscripts i, j and superscript k on U. This more complicated approximation is less accurate in practice than (4.12), although both are fourth order.

We can obtain a sixth-order accurate approximation to (1.2) by applying *tau extrapolation* [6] to approximation (4.11). We will discuss this in the next subsection.

We now discuss the smoothing (relaxation) method. On the kth grid Ω^k we must solve $J = (N_k - 1)^2$ nonlinear equations, say:

$$g_j(u_1, u_2, \cdots, u_J) = 0, \qquad j = 1, 2, \cdots, J, \tag{4.14}$$

in J unknowns.

On all but the coarsest grid, we use a method called Gauss-Seidel-Newton by Ortega and Rheinboldt [28]. In the jth equation (4.14) we fix all but the jth variable u_j, and apply Newton's method to that equation. In the process, we must replace the "old" iterates $u_j^{(i)}$ by new ones $u_j^{(i+1)}$ in all equations (4.14) as soon as they are available. Were it not for this updating, this method would be equivalent to replacing the full Jacobian for the system (4.14) by its diagonal.

More specifically, the new approximation $u_j^{(i+1)}$ to the jth unknown is given by

$$u_j^{(i+1)} = u_j^{(i)} - g_j(u^{j,i})/(\partial g_j/\partial u_j)(u^{j,i}),$$

where we have set

$$u^{j,i} = (u_1^{(i+1)}, u_2^{(i+1)}, \cdots, u_{j-1}^{(i+1)}, u_j^{(i)}, \cdots, u_J^{(i)}).$$

Together with the full approximation scheme, this method avoids using large Jacobians. Of course, this smoothing algorithm does not work in all problems, see [8].

We used "checkerboard" (or red/black) ordering of the unknowns for relaxation. For the second order approximation, we first relax all the points (x_i, y_j) with $i+j$ even, then all the points with $i+j$ odd. Foerster, Stüben and Trottenberg [13] have shown that this method speeds up the rate of convergence by a factor of two for Poisson's equation with Gauss-Seidel smoothing, compared with lexicographic ordering.

For the nine-point approximations, we replaced the two colors (red and black) with four colors [15], [35]. Consider a "fundamental square" with corners $P_{ij} \equiv (x_i, y_j)$, i or $j = 1$ or 2. Each corner is given a different color. Then the colors are extended to the whole grid by periodicity: P_{ij} has the same color as P_{lm} if and only if $i \equiv l \bmod 2$ and $j \equiv m \bmod 2$. For one iteration we relax all points with the color of P_{11}, then P_{22}, P_{12}, and P_{21}. Computations by Stüben and Trottenberg [35] have shown that, for Poisson's equation, this ordering produces better smoothing properties than any other commonly used smoothing method for the fourth order Mehrstellen Verfahren approximation.

For the projection operator $\hat{I}_{k+1}^k$ on approximate solutions we use simple injection:

$$u_{ij}^k \leftarrow \hat{I}_{k+1}^k u_{2i,2j}^{k+1} \equiv u_{2i,2j}^{k+1}.$$

For the projection operator I_{k+1}^k on residuals we use full weighting, given in stencil form as

$$I_{k+1}^k \equiv \frac{1}{16}\begin{pmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{pmatrix}_{k+1}. \tag{4.15}$$

For the second order approximation, it is sufficient to use bilinear interpolation I_{k-1}^k for corrections. This is easy to program. For the interpolation operator Π_{l-1}^l of solutions used in phase 1, it is necessary to use bicubic interpolation. We also used bicubic interpolation for both interpolation operators in conjunction with our fourth-order approximations, as did Schaffer [30]. Our sixth-order approximation requires biquintic interpolation for Π.

When the region Ω is, in some coordinate system, equal to the cross product of intervals, we can easily implement these high-order interpolations. We used de Boor's [4] program SPLI2D, which computes the tensor product of one-dimensional splines, using not-a-knot end conditions. It is sufficient to use SPLI2D with bicubic and biquintic interpolation. We placed the knots at the points x_i or y_j. However, bicubic (biquintic) splines require the number of intervals N_1 on the coarsest grid to be at least 3 (5).

Since we wished to use coarse grids that were as small as possible, we also used SPLI2D with biquadratic and biquartic interpolation for the cases $N_1 = 2$ and $N_1 = 4$, respectively. This requires us to place the knots midway between the points x_i or y_j, as well as at the endpoints 0 or 1. Further details are in de Boor [4].

For our second and fourth order approximations we chose $N_1 = 4$, but $N_1 = 3$ also proved acceptable in our computations. Some workers (e.g., [35]) have used $N_1 = 2$ (one interior grid point) on Poisson's equation. We found that this produced very sensitive behavior, even divergence, near limit points of problem (1.2). For our sixth order approximation (to be described shortly) we found that using $N_1 < 5$ caused a slowdown in the rate of convergence, since the interpolation is not sufficiently accurate on the coarsest grid. Therefore we used $N_1 = 8$ for comparison with other results.

The spline interpolations are relatively expensive compared to the cost of the relaxation sweeps. But the cost of the former is still linear in the number of grid points, since SPLI2D exploits the band structure of one-dimensional spline interpolation. Fortunately the number of relaxation sweeps is about three times the number of interpolations. The second order method requires only $M - 1$ bicubic interpolations. The rest are bilinear.

In the use of Newton's method for continuation, it is customary (e.g., [24], [32]) to economize in the computation of the Jacobian G_u. This is done by not always recomputing it after computing a new iterate $u^{(i)}$. In contrast, our computation of the Jacobian on the coarsest grid is so cheap that we recompute it after every iteration. For the convergence tolerance ε_1 on the coarsest grid, we solve to nearly machine accuracy, say

$$\|u_{i+1}^1 - u_i^1\| + |\lambda_{i+1} - \lambda_i| < 15\epsilon(\|u_i^1\| + |\lambda_i|),$$

where i denotes the iteration number, ε is machine epsilon (the smallest positive number for which $1 \oplus \varepsilon \neq 1$), and $\|\cdot\|$ is the Euclidean norm.

The storage required for the Jacobian is small. Typically $N_1 = 4$, so we need store only a 9 by 9 Jacobian. We used the banded solver DGBCO, DGBSL (with pivoting) in LINPACK [12] for our computations.

Our step-size control is based on the convergence behavior of Newton's method on the coarsest grid. (See Rheinboldt [29] for another algorithm.) We choose an initial step size $s_1 - s_0$ based on our knowledge of the problem. To determine all other step sizes $s_i - s_{i-1}$, $i > 1$, we first take a trial Euler step from s_{i-1} to s_i, using as trial step size the old step size $s_{i-1} - s_{i-2}$. We then count the number of (pseudo-arc) Newton iterations performed on the coarsest grid until the first switch to the next finer grid. If the residual norms of the iterates do not decrease, or if more than six iterations are required, we multiply the trial step size by one-third and restart at s_{i-1}. If five or six iterations are performed, we accept the trial step. If three or fewer (resp. four) iterations are required, we complete this step, and set the trial steplength $s_{i+1} - s_i$ to two (resp. 1.5) times our currrent steplength. (These figures for the number of Newton iterates depend slightly on the machine precision.)

This strategy can also be used for single-grid methods. Its advantage here is the low cost of the coarse grid operations, which simultaneously approximate the fine grid equations because of the frozen tau term. This step control is not used when locating limit points, as in the previous subsection, since the root-finder supplies the step control.

For the parameters η and δ which control switching between grids we used $\delta = 0.3$, $\eta = 0.25$.

An objection is raised by T. F. Chan [10] to our use of the bordering algorithm (2.10)-(2.13). Keller [21] shows that, in the absence of roundoff error, this procedure is valid in the neighborhood of limit points. Chan asks if this is also true in the presence of roundoff error. To resolve this question, R. Schreiber [33] suggested replacing the bordering algorithm (on the coarsest grid) by a full matrix solver for the system (2.9). This solver takes no advantage of the the structure of the matrix A in (2.9), but this imposes only a small storage penalty. We obtained exactly the same results as before.

4.6. Tau extrapolation. We now explain how tau extrapolation [6] is implemented to increase the order of accuracy of the approximation to problem (1.2) from four to six. We do not implement this exactly as in [6], and we do not claim that our method is optimal.

We retain phase one of the FMG algorithm (before the finest grid is reached) unaltered; that is, we do not apply tau extrapolation in phase one. (The frozen tau technique has in effect already produced an extrapolation here.) Upon reaching phase two, we smooth on the finest grid as before, until the convergence slows. This is the last time smoothing is done on the finest grid. Throughout we use sixth order interpolations I_k^{k+1}, so the order of the interpolation error is the same as the order of the truncation error.

When the algorithm switches from the finest grid M to grid $M-1$, we form the right-hand side f^{M-1} as in (3.3). Then we modify it by forming

$$f_{\text{new}}^{M-1} \leftarrow (1-2^{-p})^{-1}[f^{M-1} - I_M^{M-1}F^M] + I_M^{M-1}F^M, \tag{4.16}$$

with $p=4$. (Note $F^M = 0$ for problem (1.2)). The expression in square brackets is, by (4.3), the relative truncation error τ_M^{M-1}. Multiplying it by the factor in parenthesis produces the local truncation error τ^{M-1}, to leading order terms. Therefore, solving $L^{M-1}u^{M-1} = f_{\text{new}}^{M-1}$ produces an $O(h^6)$ approximation on grid $M-1$. The right-hand sides of all coarser grids $M-2, M-3, \cdots, 2, 1$ are modified as in (3.3) but not as in (4.16). Finally, our sixth order interpolation of the correction $u^M - u^{M-1}$ produces a sixth order approximation on grid M.

During the rest of phase two of the multigrid algorithm, we do not smooth on the finest grid. That would only force the finest grid solution to satisfy the original (fourth order) equations.

Of course, tau extrapolation is the same as applying deferred correction once. However, we need not explicitly compute the leading terms of the local truncation error, nor form large order Jacobians.

After completing this paper we learned of the work of Schaffer [31], which extends the tau extrapolation method as follows. The procedure given above requires a minimum of two grids and applies deferred correction once. Schaffer extends this to *iterated* deferred corrections. For example, to attain $O(h^8)$ accuracy with an $O(h^4)$ basic scheme requires a minimum of three grids. Clearly this method will require less work to obtain the same $O(h^8)$ accuracy than Richardson extrapolation. Schaffer gives numerical results only for linear problems, but we believe that his method could be extended to our problem if more accurate interpolations are used. As he points out, the second-order-accurate projection operator (4.15) still suffices, due to a fortunate cancellation of errors.

5. Convergence. In this section we outline a local convergence proof for our algorithm. Further details will appear elsewhere. Our analysis is based on the techniques

of Hackbusch [16], [17]. We assume the reader is familiar with these papers, and use the notation in them with only slight changes.

Throughout we assume that all functions are as smooth as necessary. We shall also ignore round-off errors since they are dominated by truncation errors.

Let us first recall the linear theory for a fixed parameter λ. Hackbusch shows that the interpolation, projection and smoothing operators satisfy a "smoothing property" and an "approximation property". The analysis proceeds stepwise from two levels to full multigrid.

The two-level multigrid method has iteration matrix

$$M_k^{k-1} = S_k^{\nu_2}(I_k - I_{k-1}^k L_{k-1}^{-1} I_k^{k-1} L_k) S_k^{\nu_1},$$

where I_k is the identity matrix on level k and S_k is the smoothing iteration matrix on level k. Under appropriate assumptions he then shows that the norm of the two-level matrix is less than one. The iteration matrix M_M^1 for the "cycle C" multigrid algorithm (which starts with an approximation on the finest level) is then found to be a perturbation of the two-grid iteration matrix. By recursion Hackbusch shows that this matrix has norm less than one. It is characteristic of the multigrid method that this bound is independent of the grid level k. Finally, an inductive proof can be given to show that the full multigrid method will produce approximations u^M on the finest level whose "iteration error" (the difference between u^M and the solution U^M of the discretized equations) is bounded by the local truncation error.

Hackbusch provides a convergence proof in the nonlinear case (with fixed λ), but not for the FMG full approximation scheme. Instead he proves convergence for what he calls the "multigrid method of the second kind". To do this, he proves a contraction property which corresponds to the boundedness of the norm of M_M^1 by a quantity less than one in the linear case.

Let $\phi_k(u_j^k, f^k)$ be a nonlinear iteration function (the multigrid algorithm) on level k which produces a new iterate u_{j+1}^k from old iterates u_j^k and right hand side f^k. For u_j^k in a sufficiently small neighborhood of the solution U^k of the difference equation (3.2), he requires

$$\|u_{j+1}^k - U^k\| \leqq \rho \|u_j^k - U^k\|, \tag{5.1}$$

with $\rho < 1$, $k = 1, 2, \cdots, M$. Again ρ is independent of the level k.

The contraction property (5.1) is appropriate for "phase one" of the full multigrid algorithm (§ 3) when a first approximation on the finest level is being obtained. After that, for the full approximation scheme a more appropriate contraction property is

$$\|u_{j+1}^k - I_M^k(U^M)\| \leqq \rho' \|u_j^k - I_M^k(U^M)\|,$$

where $0 < \rho' < 1$. For $k = M$ this is identical to (5.1); for other k it can be deduced from (5.1). From this we can show the convergence of the FAS multigrid method for fixed λ, provided the initial guesses are sufficiently close to the solution U^M.

Now we discuss the differences between our approach to the continuation problem and that of Hackbusch. First, we parametrize our solutions in terms of pseudo-arc-length instead of the natural parameter λ. That is, given differential equation (1.1), we parametrize a solution branch as in § 4.3 by choosing pseudo-arc-length points $s_0, s_1, s_2, \cdots$, and letting the exact solution path

$$(U(s), \Lambda(s)), \tag{5.2}$$

and approximate solutions

$$(U^k(s), \lambda(s)), \quad k = 1, 2, \cdots, M, \tag{5.3}$$

depend on the pseudo-arc-length parameter s. Assume that the solution path (5.2) encounters no bifurcation points and only simple limit points, and that the starting point s_0 is such that path (5.2) at s_0 is not "too close" to a limit point. Then the results of Keller [18] and Decker and Keller [11] show that the Jacobian G_u at s_0, and the augmented Jacobian matrices (2.9) at $s = s_1, s_2, \cdots$, are nonsingular in suitable neighborhoods of the exact solution (5.2). Hence our method will not have any difficulties at limit points.

Our analysis proceeds in three steps. We examine the multigrid convergence at s_0, where λ_0 is fixed. Then we examine the errors in the initial approximations at s_1. Finally we obtain bounds on the errors in the multigrid method at s_1, where λ varies. For all other intervals $[s_i, s_{i+1}]$ we repeat the last two steps.

The proof of convergence at s_0 was already mentioned above in the FAS modification of Hackbusch's proof. The result is that

$$\|u^M(s_0) - U^M(s_0)\| \leqq C_1 h_M^p,$$

where C_1 is a constant independent of the the level k and of the mesh spacing, and p is the order of accuracy of the difference approximation.

For step two we estimate the error in the initial approximations at s_1. Hackbusch [17] analyzed both the starting procedure (4.7) and the frozen tau method when λ is the parameter. Since he did not combine the two methods, he concluded that the frozen tau method was unsatisfactory since it did not produce sufficiently accurate initial approximations. By combining his results, and the results of Decker and Keller [11] for Newton's method, we can show that the initial approximations $(u^k(s_{i+1}), \lambda(s_{i+1}))$ have truncation error $O[(s_{i+1} - s_i)h_k^p]$.

For step three we must analyze the multigrid convergence at s_1, when both u and λ vary. Let the composite vector $\tilde{u}^k$ be the vector u^k with λ appended. For our composite vector we can prove a contraction property by induction on grid levels. For $k = 1$ this is certainly true since we use only Newton's method. The most difficult case is $k = 2$. All other levels introduce no more difficulties than the case of fixed λ. This is because the coarse grid correction for two levels $k-1$ and k ($k \geqq 3$) involves no change in λ. The result is that

$$\|\tilde{u}^M(s_1) - \tilde{U}^M(s_1)\| \leqq C_2(s_1 - s_0)h_M^p,$$

where again C_2 is independent of the level and mesh spacing.

The main difficulty in these arguments is assuring that the result of each step of the algorithm lies in a sufficiently small neighborhood for the iteration of the next step to be well defined. Hackbusch has proven many of these results.

6. Numerical results. In this section we present results of our computations of the limit point locations for two different parameter values α in problem (1.2). We have also used this method in more realistic problems, e.g., the Taylor vortex problem [3]. The smoothing algorithm we used for that problem is alternating zebra [35], a variant of alternating line (or block) Gauss-Seidel-Newton [28, p. 225, eq. (39)].

We first computed the location of the limit point for $\alpha = 0$ (see Fig. 2). This is a good test problem because the very accurate results of Meis, Lehmann and Michael [23] can be used for comparison. Meis, Lehmann and Michael computed solution

points using the method described in §1, with second order approximation (4.10). They obtained three points on the solution curve near the limit point, then fitted a parabola to these points to find the location of the limit point. These calculations were done on each of a sequence of grids whose smallest mesh size was $h = 1/4, 1/8, \cdots, 1/128$. Repeated Richardson extrapolation (up to $O(h^6)$) was then used to better approximate the location of the limit point of problem (1.2) with $\alpha = 0$. The location thus obtained is:

$$\lambda_{cr} = 6.808124, \qquad \|u(\lambda_{cr})\|_\infty \equiv \|u_{cr}\|_\infty = 1.39166.$$

The maximum of u always occurs at the center $(x, y) = (1/2, 1/2)$ of Ω.

We used our methods on essentially the same grids: the second-order scheme (4.10), the fourth order schemes (4.12) and (4.13), and a sixth order-scheme obtained by applying tau extrapolation to (4.12). We show the results in Table 1. The number of intervals on the finest mesh is N_M; hence the smallest mesh size is $h_M = 1/N_M$. For each method, the values labelled ∞ were obtained by repeated Richardson extrapolation to $O(h^8)$. Thus, the extrapolated values for the second (respectively fourth, sixth)-order method were obtained by three (resp. two, one) Richardson extrapolations from the four (resp. three, two) values immediately above. Each column was calculated independently of the others, so these results serve as a severe check on each other. To our knowledge, the existence of asymptotic expansions for the global error (i.e., the validity of Richardson extrapolation) for this problem has not been proved, but these computational results provide almost certain evidence that one exists for each scheme.

From Table 1 we observe, as expected, that the sixth-order scheme is much more accurate than the fourth-order or the second-order schemes. More striking is the relationship between limit point locations on different grids. Our results show that the limit points on coarser grids may lie on either side of the limit point of the finest grid, depending on the approximation chosen. In fact, the value of λ_{cr} at the limit point increases monotonically (with decreasing grid size) for schemes (4.10) and (4.12), and decreases for the others. The maximum solution value $\|u_{cr}\|_\infty$ increases for all approximations except the fourth-order scheme (4.13). Clearly, our method does not depend on the orientation of limit points on different grids, except for starting points. Thus, if we use the second-order scheme with $M = 2$, $h_1 = 1/16$, and $h_2 = 1/32$, then we must not use $\lambda = 6.804$, for example, as a starting point.

We give the results for $\|u\|_\infty$ to fewer digits than those for λ, because of the geometry of the limit point. Locally, the solution curve in the $\lambda - \|u\|_\infty$ plane is a quadratic in the neighborhood of a limit point (u^*, λ^*), that is, $\|u - u^*\| = O(|\lambda - \lambda^*|^{1/2})$ (see, e.g., Moore and Spence [27]). Thus λ^* can be known (at best) to the machine precision, but $\|u^*\|_\infty$ can be known only to the square root of the machine precision. Our computer (VAX 11/750) has about 17 decimal digits of precision. Thus we seem to obtain 7 correct decimal digits of $\|u_{cr}\|_\infty$ and about 10 correct digits of λ_{cr}.

The second order results and the extrapolated results agree with those in [23] to the seven and six digits they gave for λ_{cr} and $\|u_{cr}\|_\infty$, respectively. The fourth order results (4.12) for $N = 16$ agree with those of [27] to the 11 digits they gave for λ. For $\|u_{cr}\|_\infty$ our eight-digit results agree with the first eight of their eleven digits.

Tables 2 and 3 show results similar to those of Table 1, but for the case $\alpha = 0.2$ in problem (1.2). Here there are two limit points. For the upper limit point we can partially compare our results with those of Mittlemann [25] for $N_M = 32$. He used second-order approximation (4.10) with his generalized inverse iteration multigrid method [25], [26], and gave results to four digits. Rather than locate the upper limit point, he showed successive values of the parameter λ on the solution path near this

TABLE 1
Location of limit point of (1.2) *with* $\alpha = 0$.

		Order and Scheme			
	N_M	2 (4.10)	4 (4.12)	4 (4.13)	6 [extrap. (4.12)]
λ_{cr}	16	6.80217409563	6.80808657467	6.80830691585	6.80812517811
	32	6.80665272920	6.80812207169	6.80813582072	6.80812443571
	64	6.80775749456	6.80812427588	6.80812513486	6.80812442280
	128	6.80803275282	6.80812441342	6.80812446710	6.80812442259
	∞	6.80812442263	6.80812442259	6.80812442258	6.80812442259
$\|u\|_\infty$	16	1.3888573	1.3916567	1.3917381	1.3916593
	32	1.3909601	1.3916609	1.3916661	1.3916612
	64	1.3914859	1.3916612	1.3916615	1.3916612
	128	1.3916174	1.3916612	1.3916612	1.3916612
	∞	1.3916612	1.3916612	1.3916612	1.3916612

TABLE 2
Location of lower limit point of (1.2) *with* $\alpha = 0.2$.

		Order		
	N_M	2	4 (4.12)	6
λ_{cr}^{l}	16	9.12131236518	9.13630924484	9.13638435052
	32	9.13263701343	9.13637838604	9.13638298751
	64	9.13544784102	9.13638268079	9.13638296698
	128	9.13614927051	9.13638294880	9.13638296666
	∞	9.13638296666	9.13638296667	9.13638296666
$\|u\|_\infty$	16	2.8756967	2.8857321	2.8858002
	32	2.8832818	2.8857962	2.8858004
	64	2.8851712	2.8858001	2.8858004
	128	2.8856430	2.8858003	2.8858004
	∞	2.8858003	2.8858003	2.8858004

TABLE 3
Location of upper limit point of (1.2) *with* $\alpha = 0.2$.

		Order		
	N_M	2	4 (4.12)	6
λ_{cr}^{u}	16	7.08025536111	7.10152536891	7.10195697086
	32	7.09656018055	7.10187720721	7.10190065819
	64	7.10056845538	7.10189761734	7.10189897803
	128	7.10156658312	7.10189886674	7.10189894998
	∞	7.10189894893	7.10189894958	7.10189894953
$\|u\|_\infty$	16	18.207497	18.195894	18.192661
	32	18.195850	18.192933	18.193740
	64	18.193507	18.192778	18.192767
	128	18.192951	18.192768	18.192768
	∞	18.192768	18.192767	18.192768

point. These values of λ, in increasing order of $\|u\|_\infty$, are 7.103, 7.097, 7.096, 7.098, 7.104. Our value of λ_{cr}^{u} for $N_M = 32$ in the first column of Table 3 agrees well with these. Unfortunately, he did not supply values of $\|u\|_\infty$.

Our starting guess for the solution values in Tables 1 and 2 was zero. For Table 3 our starting guess was $u = 12$ in the interior of the coarse grid, and $u = 0$ on the boundary. Of course, we obtained the same results by starting on the lower branch and continuing to the upper limit point.

In all cases we iterate the multigrid algorithm until two successive solutions are sufficiently close:

$$\|u_{i+1}^M - u_i^M\| < \varepsilon \|u_i^M\|.$$

Here the subscript denotes the iteration number and ϵ is a small number related to the machine precision. Normally we would iterate only until the norm of the residual is less than the norm of the relative truncation error τ_M^{M-1}. But we wished to ensure that the "convergence error" was much less than the truncation error so that we could perform Richardson extrapolation.

The work required to reduce the l_2 norm of the residuals by a factor of 10^{-12} was never more than 35 work units, and usually about 25 units (after the first step s_0). A work unit (following Brandt) is the work required to relax all the grid points once on the finest grid. It does not include the work required for interpolations or projections. The work required to reduce the residuals to the level of the truncation error was 4–10 work units after the first step, depending on the length of the step $s_i - s_{i-1}$. Computing one solution of problem (1.2) on a 129 by 129 grid ($N_M = 128$) took approximately 20 minutes on a DEC VAX 11/750. This reduced the l_2 norm of the residuals by a factor of 10^{-12}. (This machine is about 70% of the speed of a VAX 11/780.) We know from § 2 that the pseudo-arc-length Newton method eliminates slow convergence or divergence near limit points when used with a "single grid" method. Similarly, it is experimentally observed that our method eliminates convergence difficulties with the multigrid method at simple limit points. This is confirmed by the experimental observation that *the multigrid convergence rate is the same near limit points as away from them.* An exception to this observation occurs at the starting point s_0, where we cannot use the pseudo-arc-length procedure during the multigrid iteration. If this point is too close to the limit point, the rate of convergence will of course be slow, but *only* at s_0.

Acknowledgments. We wish to thank Achi Brandt for technical discussions. He discovered this method before us but has not published it. We are also grateful for the use of computer time on the Fluid Dynamics VAX and the Applied Math-IBM 4341 at Caltech. The former is supported by the Office of Naval Research, and the latter is supported by the IBM Corporation. Finally, we thank the referees for carefully reading the manuscript.

REFERENCES

[1] J. P. ABBOTT, *An efficient algorithm for the determination of certain bifurcation points*, J. Comput. Appl. Math., 4 (1978), pp. 19–27.

[2] R. E. BANK AND T. F. CHAN, *PLTMGC: A multigrid continuation package for solving parametrized nonlinear elliptic systems*, Report 261, Dept. of Computer Science, Yale University, New Haven, CT, 1983.

[3] J. H. BOLSTAD AND H. B. KELLER, *Computation of anomalous modes in the Taylor experiment*, to appear.

[4] C. DE BOOR, *A practical guide to splines*, Springer Verlag, New York, 1978.

[5] A. BRANDT, *Multi-level adaptive solutions to boundary-value problems*, Math. Comp., 31 (1977), pp. 333–390.

[6] A. BRANDT AND N. DINAR, *Multigrid solutions to elliptic flow problems*, in Numerical Methods for Partial Differential Equations, S. Parter, ed., Academic Press, New York, 1979, pp. 53–147.

[7] A. BRANDT, *Multigrid solvers on parallel computers*, in Elliptic Problem Solvers, M. Schultz, ed., Academic Press, New York, 1981, pp. 39–83.

[8] A. BRANDT, *Guide to multigrid development*, in Multigrid Methods, Lecture Notes in Mathematics 960, W. Hackbusch and U. Trottenberg, eds., Springer Verlag, New York, 1982, pp. 220-312.

[9] T. F. CHAN AND H. B. KELLER, *Arc-length continuation and multigrid techniques for nonlinear elliptic eigenvalue problems*, this Journal, 3 (1980), pp. 173-194.

[10] T. F. CHAN, personal communication.

[11] D. W. DECKER AND H. B. KELLER, *Path following near bifurcation*, Comm. Pure Appl. Math., 34 (1981), pp. 149-175.

[12] J. J. DONGARRA, J. R. BUNCH, C. B. MOLER AND G. W. STEWART, LINPACK *Users' Guide*, Society for Industrial and Applied Mathematics, Philadelphia, 1979.

[13] H. FOERSTER, K. STUEBEN AND U. TROTTENBERG, *Non-standard multigrid techniques using checkered relaxation and intermediate grids*, in Elliptic Problem Solvers, M. Schultz, ed., Academic Press, New York, 1981, pp. 285-300.

[14] G. FORSYTHE, M. MALCOLM AND C. MOLER, *Computer Methods for Mathematical Computations*, Prentice-Hall, Englewood Cliffs, NJ, 1977.

[15] W. HACKBUSCH, *On the multigrid method applied to difference equations*, Computing, 20 (1978), pp. 291-306.

[16] W. HACKBUSCH, *Multigrid convergence theory*, in Multigrid Methods, Lecture Notes in Mathematics 960, W. Hackbusch and U. Trottenberg, eds., Springer Verlag, New York, 1982, pp. 177-219.

[17] W. HACKBUSCH, *Multigrid solution of continuation problems*, in Iterative Lösung Nichtlinearer Gleichungssysteme, Lecture Notes in Mathematics 953, R. Ansorge, T. Meis and W. Törnig, eds., Springer Verlag, New York, 1982.

[18] H. B. KELLER, *Approximation methods for nonlinear problems with application to two-point boundary value problems*, Math. Comp., 29 (1975), pp. 464-474.

[19] H. B. KELLER, *Numerical solutions of bifurcation and nonlinear eigenvalue problems*, in Applications of Bifurcation Theory, P. Rabinowitz, ed., Academic Press, New York, 1977, pp. 359-384.

[20] H. B. KELLER, *Global homotopies and Newton methods*, in Recent Advances in Numerical Analysis, C. de Boor and G. Golub, eds., Academic Press, New York, 1978, pp. 73-94.

[21] H. B. KELLER, *Practical procedures in path following near limit points*, in Computing Methods in Applied Sciences and Engineering, R. Glowinski and J. Lions, eds., North-Holland, New York, 1982.

[22] M. LENTINI AND H. B. KELLER, *The von Karman swirling flows*, SIAM J. Appl. Math., 38 (1980), pp. 52-64.

[23] T. MEIS, H. LEHMANN AND H. MICHAEL, *Application of the multigrid method to a nonlinear indefinite problem*, in Multigrid Methods, Lecture Notes in Mathematics 960, W. Hackbusch and U. Trottenberg, eds., Springer Verlag, New York, 1982, pp. 545-557.

[24] R. MEYER-SPASCHE AND H. B. KELLER, *Computations of the axisymmetric flow between rotating cylinders*, J. Comp. Phys., 35 (1980), pp. 100-109.

[25] H. D. MITTLEMANN, *Multigrid methods for simple bifurcation problems*, in Multigrid Methods, Lecture Notes in Mathematics 960, W. Hackbusch and U. Trottenberg, eds., Springer Verlag, New York, 1982, pp. 558-575.

[26] H. D. MITTLEMANN AND H. WEBER, *Multigrid solution of bifurcation problems*, International Multigrid Conference, Copper Mountain, CO, 1983.

[27] G. MOORE AND A. SPENCE, *The calculation of turning points of nonlinear equations*, SIAM J. Numer. Anal., 17 (1980), pp. 567-576.

[28] J. ORTEGA AND W. RHEINBOLDT, *Iterative Solution of Nonlinear Equations in Several Variables*, Academic Press, New York, 1970.

[29] W. RHEINBOLDT, *Solution fields of nonlinear equations and continuation methods*, SIAM J. Numer. Anal., 17 (1980), pp. 221-237.

[30] S. SCHAFFER, *High order multigrid methods to solve the Poisson equation*, in Multigrid Methods, H. Lomax, ed., NASA Conference Publication 2202, 1982, pp. 275-284.

[31] S. SCHAFFER, *Higher order multigrid methods*, Math. Comp., 43 (1984), pp. 89-115.

[32] R. SCHREIBER AND H. B. KELLER, *Driven cavity flows by efficient numerical techniques*, J. Comp. Phys., 49 (1983), pp. 310-333.

[33] R. SCHREIBER, personal communication.

[34] A. SPENCE AND B. WERNER, *Non-simple turning points and cusps*, IMA J. Numer. Anal., 2 (1982), pp. 413-427.

[35] K. STUEBEN AND U. TROTTENBERG, *Multigrid methods: Fundamental algorithms, model problem analysis and applications*, in Multigrid Methods, Lecture Notes in Mathematics 960, W. Hackbusch and U. Trottenberg, eds., Springer Verlag, New York, 1982, pp. 1-176.

[36] R. K. H. SZETO, *The flow between rotating coaxial disks*, Ph.D. Thesis, California Institute of Technology, Pasadena, CA, 1978.

SIAM J. SCI. STAT. COMPUT.
Vol. 7, No. 4, October 1986

ON AN ADAPTIVE GRID REFINING TECHNIQUE FOR FINITE ELEMENT APPROXIMATIONS*

HELMUT JARAUSCH†

Abstract. We consider a family of finite element spaces and minimize an energy functional over each space. The space which allows the lowest energy is considered "optimal." Such a family is constructed by starting with an initial "triangulation" and refining one or more "triangles" at a time. We estimate the profit in energy gained by refining a triangle and set up a discrete optimization problem which determines the optimal refinement strategy according to a prescribed bound of the costs. This enables us to construct the final grid by using as few as possible intermediate grids. Instead of solving the original optimization problem we set up a partially dualized form of it which produces a nearly optimal solution and can be solved very efficiently.

Key words. adaptive grid generation, finite element error estimation, multiple choice knapsack problem

AMS(MOS) subject classifications. 65M15, 65N30, 65K99

0. Introduction. To compute finite element approximations to elliptic differential equations efficiently one has to minimize the energy functional over a *suitable* finite element space. In general the achievable energy (or energy norm of the error) depends on the given finite element space even if the dimension of that space, which influences the costs of computations, is prescribed. The optimal space (with fixed dimension) is not known in advance but depends strongly on the data of the given problem. Therefore one tries to approximate such a space by what is called a "feedback finite element method" (see [8], [13]). Roughly, a feedback finite element method consists of the following ingredients (see [13] for a theory on adaptive processes):

—an a posteriori error estimator;
—a transition operator which refines certain elements of the old grid based on the information given by the error estimator;
—a sophisticated data structure.

This paper is primarily concerned with a new type of transition operator. We refer the reader to [3]–[6], [17] for a posteriori error estimates and to [9], [14]–[16] for suitable data structures.

We apply the new transition operator to the so-called "h-version" of the finite element method. It may well be adapted to the "p-version" (see [17], e.g.) as well as to a combined "h-p-version" (see [7]).

We restrict ourselves to the context of conforming finite elements. We study the problem how to minimize the energy functional

$$E(u) \equiv \tfrac{1}{2}a(u, u) - f(u) \quad \text{for } u \in H, \tag{0.1}$$

where H is a Hilbert space which is assumed to be a subspace of some Sobolev space over a region $\Omega \subset \mathbb{R}^n$. The solution to (0.1) will be denoted by

$$u^* \equiv \arg\min_H E(u).$$

For some (finite-dimensional) subspace $V_h \subset H$ let

$$u_h \equiv \arg\min_{V_h} E(u) \quad \text{and} \quad E(V_h) \equiv E(u_h).$$

* Received by the editors June 6, 1984, and in revised form March 19, 1985.

† Institut für Geometrie und Praktische Mathematik, Rheinisch-Westfälische Technische Hochschule Aachen, D-5100 Aachen, West Germany.

Within a *given class* $\mathcal{V}$ of subspaces V_h of H we define the *optimal subspace* V_h^* by

$$V_h^* \equiv \arg\min_{\mathcal{V}} E(V_h).$$

Since we have $\|u^* - u_h\|_E^2 = 2(E(u_h) - E(u^*))$ for the *energy norm* $\|u\|_E = \sqrt{a(u,u)}$ the solution in the optimal space V_h^* is best with respect to the energy norm.

In the following we consider abstract triangulations $\mathcal{T}$ of Ω, where $\mathcal{T} = \cup\{T_i\}$ and the "triangles" T_i cover Ω. For ease of notation we restrict ourselves to F.E. spaces of piecewise polynomials of fixed degree:

$$V_h = \{u \in H: u_{|T_i} \text{ is a polynomial of degree at most } k\}.$$

We assume a *cost function* $W(V_h)$ which assigns to each F.E. space V_h the costs of minimizing E over this space. $W(V_h)$ is required to be strictly increasing with respect to the dimension of V_h.

Given some initial F.E. space V_0 and some budget w^* with $w^* \geqq W(V_0)$ we consider the class of spaces

$$\mathcal{V} := \{V_h: V_0 \subset V_h \subset H,\ W(V_h) \leqq w^*\}.$$

Each space $V_h \in \mathcal{V}$ can be considered as a knot of a tree structure with root V_0 where each son can be constructed from his father by "refining" (at least) one "triangle" T_i of his father's triangulation (see [12]).

Let V be an intermediate space $V_0 \subset V \subset V_h^*$ with the corresponding triangulation $\mathcal{T} = \{T_1, \cdots, T_m\}$. For simplicity we assume that "refining" a triangle will always be done in a fixed way. We introduce the *decision vector* $z \in \{0,1\}^m$ and denote by $V[z]$ the space obtained from B by refining exactly those triangles T_i with $z_i = 1$. For each triangle T_i we introduce the *energy profit* $r_i \equiv E(V) - E(V[e_i]) \geqq 0$ and the *cost coefficient* $c_i \equiv W(V[e_i]) - W(V) > 0$ where e_i is the ith unit vector in $\mathbb{R}^m$.

Note that the energy profit serves as one possible error estimator. We are concerned with the construction of a transition operator which maps V into $V[z]$ and approximates the optimal path from V_0 to V_h^* efficiently. A simple but theoretically very attractive transition operator refines only the most profitable element at a time (OMP-strategy), i.e. $z = e_k$ where $r_k \geqq r_i$ for $1 \leqq i \leqq m$.

The OMP-strategy produces very good grids and enjoys very good theoretical properties. Rheinboldt shows a certain optimality of this strategy since his assumptions in [12, Thm. 10] seem to be fulfilled for the energy profit in practice. The OMP-strategy will therefore be used to judge the quality of the grids produced by the new transition operator.

The only drawback of this strategy is its high costs since the grids are modified only very little from step to step. Therefore other transition operators have been devised, see e.g. [3], [7]–[9], [12], [15]–[17]. All of these include the most profitable element but try to refine several elements simultaneously.

To use *as few as possible* intermediate spaces, any triangle of an intermediate space which has to be refined anyway on the path to the optimal space V_h^* should be *refined as soon as possible.* But to do so we have to know how many times a triangle will have to be refined on that path and if there will be some money left to refine other triangles as well. Thus we end up in an optimization problem which predicts how many times each triangle can be refined with the given budget.

We go even one step further and introduce long and short passes as suggested in [3], [7], [15]. In a long pass we solve for the finite element approximation on the latest intermediate space before we apply the transition operator once more. In contrast to

that within a short pass we bypass the expensive solution process by solving only local inversion problems (see § 5 and [15]). Based on these locally solved solutions we can compute the energy profit of the new elements and start the next optimization step just as within a long pass. In our numerical examples we obtained identical grids to those given by the OMP-strategy in all cases but one even if we insert as many short passes as are necessary to increase the number of elements by at least 100% between successive long passes.

In § 1 we introduce the optimization problem involved in the transition operator in long as well as in short passes.

In § 2 we estimate the energy profit involving both local and global data. This serves as an error estimator for our purposes.

Section 3 motivates an extrapolation scheme for the energy profit of descendants of an element. It includes a simple a posteriori error estimator for the final approximation.

Section 4 deals with the approximate solution of the discrete optimization problem that was set up in § 1.

In § 5 we give the overall algorithm involving long and short passes. This is applied to some numerical examples in § 6.

Some concluding remarks will be given in § 7.

1. The optimization problem. The problem of finding the grid associated with $V[z]$ may be formulated as a discrete optimization problem for the decision vector z. To make the approach computationally feasible we need some

Simplifying assumptions.

The energy profit is approximately additive:

$$E(V)-E(V[z])\approx \sum_{z_i=1} r_i,$$

and the cost function is approximately additive:

$$W(V[z])-W(V)\approx \sum_{z_i=1} c_i.$$

These assumptions have been proven practically valid in our numerical tests.

When proceeding from V to $V[z]$ we have to decide whether it is more profitable to refine triangle T_k several times opposed to refining it only once together with some other triangles T_j, $j\neq k$. Since we have not yet refined the triangle T_k our procedure to estimate the energy profit cannot be applied to the sons of T_k. Therefore we have to predict these profits by some local extrapolation technique, see § 3. This leads to a predicted energy profit of the form $p_k\cdot r_k$ for the sons, $p_k^2\cdot r_k$ for the grandsons, and so on.

The predicted energy profit of the triangle T_k, when refined once, is given by

$$r_{k,l}\equiv r_k\cdot \sum_{\nu=0}^{l-1} p_k^{\nu}.$$

Similarly we assume for the cost coefficient

$$c_{k,l}\equiv c_k\cdot \sum_{\nu=0}^{l-1} q_k^{\nu}$$

for some $q_k\geqq 1$ depending on the refinement process.

Let us denote by (x_{kj}) a preliminary decision vector with $x_{kj}=1$ iff T_k has been refined j times. The problem of deciding how many times a given triangle T_k has to

be refined is equivalent to extracting exactly one member out of the *multiple choice class*

$$\text{(MCC)} \qquad \{(r_{k0}, c_{k0}), (r_{k1}, c_{k1}), \cdots, (r_{kj}, c_{kj}), \cdots\}$$

where we have added the *slack variables* $r_{k0} \equiv 0$, $c_{k0} \equiv 0$ corresponding to T_k being *not* refined.

The problem of finding the optimal successor $V[z]$ of a given intermediate space V can be modelled by the following.

Multiple choice knapsack problem.

$$\text{(MCK)} \qquad P(w) \equiv \min_{\substack{\langle c,x\rangle \leqq w \\ x \in \mathrm{MC}}} -\langle r, x\rangle$$

where

$$\langle r, x\rangle \equiv \sum_{k=1}^{m} \sum_{j=0}^{N} r_{kj} x_{kj},$$

$$x \in \mathrm{MC} \quad \text{iff} \quad \sum_{j=0}^{N} x_{kj} = 1 \quad \text{for all } k, \quad x_{kj} \in \{0, 1\}$$

and N is a sufficiently large integer.

Given an (approximate) solution x_{kj} to MCK (see § 4) each triangle which has been selected to be refined at least once (i.e. $x_{k0} = 0$) will be refined *only once.* We thus form the actual decision vector z by $z_k = 1$ iff $x_{k0} = 0$. We note that the transition $V \to V[z]$ does not in general use all of the given budget even though the MCK problem might have spent all of it ($\langle c, x\rangle = w$). Thus we update our budget according to $w \leftarrow w - \sum_{k=1}^{m} c_k \cdot z_k$, set $V \leftarrow V[z]$ and proceed until we have exhausted our budget.

Remarks.

—Since the exact solution of MCK is a NP-hard problem, we shall suggest a partially dualized problem which solves MCK approximately in very short time.

—In the implementation we do not store x_{kj} but y_k with $0 \leqq y_k \leqq N$ and $x_{kj} = 1$ iff $y_k = j$. Therefore the size of N does not matter in practical computations. Because of the fast progression of work ($q_k \geqq 2$ in general) the constraint $\langle c, x\rangle \leqq w$ limits y_k to very small numbers anyway.

2. Estimating the energy profit. In this section we propose an easily computable estimator of the energy profit by using a local 2-grid approach. We have two finite-dimensional subspaces $V_H \subset V_h$ with fixed bases. The matrix representation of the imbedding will be denoted by P. For any function $u_H \in V_H$ (0.1) leads to

$$E(u_H) = \tfrac{1}{2} u_H^T A_H u_H - f_H^T u_H \tag{2.1}$$

where the function u_H and the vector representation of it have been identified. If the minimum energy in V_H is achieved at $\bar{u}_H$ we have

$$E(\bar{u}_H) = -\tfrac{1}{2} f_H^T \bar{u}_H = -\tfrac{1}{2} \bar{u}_H^T A_H \bar{u}_H. \tag{2.2}$$

With the bases fixed in V_H and V_h, there is a canonical way to compute A_H, A_h, f_H, f_h. Note the relations

$$A_H = P^T A_h P \quad \text{and} \quad f_H = P^T f_h. \tag{2.3}$$

To motivate our energy estimator let us look at a typical 2-grid iteration to compute the solution $\bar{u}_h \in V_h$ given $\bar{u}_H \in V_H$.

Step 0. Prolongate $\bar{u}_H$ giving $u^0 \equiv P\bar{u}_H$.

Step 1. Smooth u^0 giving u^1. This can be done by some (local) relaxation scheme or local inversion.

Step 2. Compute the residuum $r_H \equiv P^T(A_h u^1 - f_h)$. Note that this can be done locally by only computing the new element matrices which have to be computed anyway in most cases. Simultaneously we can compute the difference in energy between $E(\bar{u}_H) = E(P\bar{u}_H)$ and $E(u^1)$. This is simply done by subtracting the old element energies from the new ones but only where new elements have been created.

Step 3. Solve $A_H \Delta_H = r_H$ and correct $u^2 \equiv u^1 + P\Delta_H$. Note that this step is not too expensive since we have already solved a linear system involving A_H, namely $A_H \bar{u}_H = f_H$. Thus we have an (incomplete) Cholesky factorization at hand and Δ_H can be computed (approximately) by a simple backsolve or some preconditioned CG steps or by some multigrid cycles.

One can easily verify the following.

LEMMA. *Let $\Delta_h \equiv \bar{u}_h - u^0$ and $r_h \equiv A_h u^0 - f_h$. Then the exact energy profit is given by $E(\bar{u}_H) - E(\bar{u}_h) = \frac{1}{2} r_h^T \Delta_h$. Similarly we have $E(u^1) - E(u^2) = \frac{1}{2} r_H^T \Delta_H$.*

We now estimate the energy profit by

$$E(\bar{u}_H) - E(\bar{u}_h) \approx E(\bar{u}_H) - E(u^1) + \tfrac{1}{2} r_H^T \Delta_H, \tag{2.4}$$

where the difference on the right-hand side has been computed in Step 2.

Remarks. Preliminary numerical experiments indicate that

—The estimate (4) differs from the exact energy profit by far less than 1%.

—The energy profit when refining triangle T_k is only weakly dependent on neighbouring triangles being refined or not. Thus the energy profit of a triangle has to be computed only when this triangle has just been created.

3. Extrapolating the energy profit. In this section we give a crude heuristic for extrapolating the energy profit if a triangle were to be refined several times. We restrict ourselves to second-order problems; the resulting extrapolation scheme will probably hold for other problems, as well.

In § 1 we have assumed that refining a triangle T_k and all of its descendants up to the lth generation gives the total energy profit

$$r_{k,l} = r_k \sum_{\nu=0}^{l-1} p_k^{\nu}. \tag{3.1}$$

We try to motivate this ansatz and give an interpretation of the factor p_k.

In the standard finite element error analysis the error in energy of the finite element solution can be dominated by the approximation error in the Sobolev space H^1 which in turn can be dominated by the interpolation error in that space. Furthermore the interpolation error estimate gives the correct order of convergence of the finite element error. The interpolation error can be computed locally and on each triangle since it depends on the local regularity of the finite element solution u^*.

Assume $u^* \in H^{1+r}(T)$, $r \in \mathbb{R}^+$ has to be interpolated by polynomials of degree at most k. Let $s \equiv \min\{k, r\}$. Then the interpolation error in the norm of H^1 can be estimated by

$$\text{const} \cdot h^s \|u^*\|_{1+s,T}$$

where h denotes the diameter of the triangle T and the norm $\|\cdot\|_{1+s,T}$ corresponds to the Sobolev space $H^{1+s}(T)$ and s depends on T: $s(T)$.

Therefore we introduce the error indicator

$$\varepsilon(\mathcal{T}) \equiv C \sum_{T} h^{2s(T)} \|u^*\|^2_{1+s(T),T} \quad \text{with some constant } C \tag{3.2}$$

and assume $E(\bar{u}_h) - E(u^*) \approx \varepsilon(\mathcal{T})$.

Next we try to predict the change in $\varepsilon(\mathcal{T})$ when one triangle $T \in \mathcal{T}$ is refined. We assume that the regularity of u^* within T does not change much, i.e.

$$\frac{1}{|T'|} \|u^*\|^2_{1+s,T'} \approx \frac{1}{|T|} \|u^*\|^2_{1+s,T} \quad \text{for all } T' \subset T, \tag{3.3}$$

where $|T|$ denotes the Lebesgue measure of T. Let us assume for simplicity that

(3.4) refining T produces m new elements T'_i with equal diameter γh.

Now the term $h^{2s}\|u^*\|^2_{1+s,T}$ in the sum (3.2) changes to

$$(\gamma h)^{2s} \sum_{i=1}^{m} \|u^*\|^2_{1+s,T'_i} \approx (\gamma h)^{2s} \|u^*\|^2_{1+s,T}$$

because of assumption (3.3). Thus the energy profit of dividing T into m parts amounts to

$$\Delta E \equiv [1 - \gamma^{2s}] h^{2s} C \|u^*\|^2_{1+s,T}. \tag{3.5}$$

Refining all triangles T'_i we get for the additional energy profit ΔE_+

$$\Delta E_+ = (\gamma^{2s}) \Delta E \tag{3.6}$$

where we have used assumptions (3.3) and (3.4) again. This motivates our ansatz (3.1) with

$$p = \gamma^{2s} \tag{3.7}$$

which shows the connection of this factor to the local regularity of the solution.

In practical computations we proceed as follows: Whenever a triangle is created we compute the energy profit of it and store this value with this new triangle. The same is done for each of its brothers. Summing up all these profits gives ΔE_+. This value is divided by the energy profit of the common father. Numerical computations have shown that it is advantageous not to use p itself in the ansatz (3.1) but to use the mean of p and the corresponding value of its father. The p-factors of all triangles belonging to the initial grid are set to 1. This makes the solution look very rough in the first two stages. A very careful strategy results since only elements of maximum profit are refined in the first stage.

Finally the ansatz (3.1) can also be used as an a posteriori estimate by summing up to infinity, i.e. the a posteriori error is estimated by

$$\|E(\bar{u}_h) - E(u^*)\| \approx \sum_{k} r_k (1 - p_k)^{-1}.$$

This estimate works quite well for examples with solutions which are not too rough, i.e. $p_k < 1$.

4. Solving the multiple choice knapsack problem. In § 1 we have modelled our grid adaption problem by a multiple choice knapsack problem:

$$P(w) \equiv \min_{\substack{\langle c,x\rangle \leqq w \\ x \in \mathrm{MC}}} -\langle r, x\rangle \tag{MCK}$$

where

$$\langle r, x\rangle \equiv \sum_{k=1}^{m}\sum_{j=0}^{N} r_{kj}x_{kj},$$

$$\mathrm{MC} \equiv \left\{ x \in \{0,1\}^n \;\middle|\; \sum_{j=0}^{N} x_{kj} = 1 \text{ for all } k \right\};$$

here $n = m(N+1)$.

Furthermore we had the slack variables $r_{k0} = c_{k0} = 0$ for all k. Now the solution of (MCK) is known to be a NP-hard problem. Since the MCK-problems originating from § 1 are big ones, an exact solution of MCK is prohibitive. We give a very cheap technique to compute a suboptimal solution which only uses less money $\tilde{w} \leqq w$, but would be the exact solution if we had only a budget $\tilde{w}$. Since our model is based on estimated (energy-) profits for multipally refined triangles is not essential to use all of the given budget but to use any money in an optimal fashion. We replace our primal problem MCK by a partially dualized one by giving up the constraint $\langle c, x\rangle \leqq w$ and adding a Lagrangian multiplier term instead. We have the following Lagrangian-relaxed problem (cf. [10]):

$$\text{(LRMCK)} \qquad D(\lambda) \equiv \min_{x \in \mathrm{MC}} -\langle r, x\rangle + \lambda(\langle c, x\rangle - w) \quad \text{for } \lambda \geqq 0,$$

and the *dual problem*

$$\text{(DMCK)} \qquad \lambda^* \equiv \arg^* \max_{\lambda \geqq 0} D(\lambda)$$

where arg* denotes the largest λ for which the maximum is attained. This is unique in most cases.

These problems are related by the following.

LEMMA 1. (i) (weak duality). *For any $w \in \mathbb{R}_+$ and any corresponding λ^* for which the maximum is attained in* DMCK, *we have $D(\lambda^*) \leqq P(w)$.*

(ii) (partial strong duality). *For any positive λ^* take a solution x^* of* LRMCK, *then for $\tilde{w} \equiv \langle c, x^*\rangle$ we have $D(\lambda^*) = P(\tilde{w})$.*

(iii) *Let λ^* be the solution of* DMCK *then there is a corresponding solution x^* to* LRMCK *with $\langle c, x^*\rangle \leqq w$, i.e. x^* is feasible for* MCK.

Proof. Parts (i) and (ii) are easy whereas part (iii) follows from Lemma 7 below.

Remark. Assertion (iii) implies that x^* is the optimal solution of MCK with a budget $\tilde{w} \equiv \langle c, x^*\rangle \leqq w$.

The main advantage of dualizing the problem comes from the fact that the Lagrangian relaxed problem LRMCK decomposes into m small problems within each multiple choice class.

LEMMA 2 (decomposition). *Given $\lambda \in \mathbb{R}_+$, x^* solves* LRMCK *iff for each k we have $x^*_{ki} = \delta_{ij}$ where j is a solution of* $\min_l (\lambda c_{kl} - r_{kl})$.

Since our solution algorithm to DMCK involves several LRMCK problems with decreasing values of λ it is profitable to give a more explicit solution formula for LRMCK.

First we have to eliminate some degenerate cases of MCK problems. However it will turn out that for MCK problems originating from our grid adaption process such pathological cases do not arise (see Lemma 6 below).

We have the immediate result.

LEMMA 3. *Assume problem* MCK *has a solution (i.e. there is at least one feasible point) then for all* $0 \leqq i, j \leqq N$, $i \neq j$ *with* $c_{ki} \leqq c_{kj}$ *but* $r_{ki} \geqq r_{kj}$ *there is an optimal solution* x *for which* $x_{kj} = 0$.

Such an x_{kj} is called an *integer dominated variable.* Thus we can assume that all integer dominated variables have been removed in advance.

LEMMA 4. *Given a variable* x_{kj}, *assume there are variables* x_{ki} *and* x_{kl} *with* $c_{ki} < c_{kj} < c_{kl}$ *but* $(r_{kj} - r_{ki})/(c_{kj} - c_{ki}) < (r_{kl} - r_{ki})/(c_{kl} - c_{ki})$.

Then $x_{kj} = 0$ *in each optimal solution to* LRMCK *for any* $\lambda \in \mathbb{R}_+$.

We call such a variable x_{kj} *Lagrangian dominated.* Thus, as far as DMCK is concerned, we can assume that all Lagrangian dominated variables have been removed in advance.

The proof of Lemma 4 can be easily reduced to the following.

GEOMETRIC LEMMA 5. *Assume we have three straight lines* $c_\alpha t - r_\alpha$, $c_\beta t - r_\beta$ *and* $c_\gamma t - r_\gamma$. *The point of intersection of lines* α *and* β *is denoted by* $\sigma_{\alpha\beta} \equiv (r_\alpha - r_\beta)/(c_\alpha - c_\beta)$. *Then with* $c_\alpha < c_\beta < c_\gamma$, $\sigma_{\beta\gamma} < \sigma_{\alpha\beta}$ *implies* $\sigma_{\beta\gamma} < \sigma_{\alpha\gamma} < \sigma_{\alpha\beta}$.

Proof of Lemma 4. Assume there is a λ and an optimal solution x to LRMCK such that $x_{kj} = 1$. From Lemma 2 we have

$$\lambda c_{kj} - r_{kj} \leqq \lambda c_{km} - r_{km} \quad \text{for all } m.$$

Now $m = l$ gives $\sigma_{jl} \leqq \lambda$ while $m = i$ gives $\lambda \leqq \sigma_{ij}$. Then it follows from the geometric lemma that $\sigma_{il} < \sigma_{ij}$ which contradicts the assumptions of Lemma 4.

In general one has to remove all dominated variables. This can be done by sorting the pairs (c_α, r_α) such that the c_α's are monotone increasing and then removing all pairs where either r_α is nonincreasing (integer dominated) or the quotients $\sigma_{\alpha\beta}$ with $\alpha = (i, k)$ and $\beta = (i, k-1)$ are nonincreasing. But in our case we had in § 1

$$r_1 = r \sum_{\nu=0}^{l-1} p^\nu \quad \text{and} \quad c_l = c \sum_{\nu=0}^{l-1} q^\nu$$

where we have suppressed the class-index k.

LEMMA 6. *For* $r > 0$, $c > 0$ *and* $0 < p < q$ *there are no dominated variables in* MCK *or* LRMCK.

Proof. For the Lagrangian dominated variables $\sigma_{k,k-1} = r/c\ (p/q)^{k-1}$ is monotone decreasing in k. The result now follows from the geometric lemma.

COROLLARY. *Since in our case we always have* $p \leqq 1$ (*see* § 3) *and* $q > 1$ *by the assumption of monotone work, we always have* $p < q$ *in our context.*

DEFINITION. The quotients $S_{kj} \equiv (r_{k,j} - r_{k,j-1})/(c_{k,j} - c_{k,j-1})$ for $j \geqq 1$ are called *incremental profit densities.* In addition we define $S_{k0} \equiv \infty$.

We are now in a position to characterize the solutions to LRMCK.

THEOREM 1. *Assume that all integer- and Lagrangian-dominated variables have been removed in the Lagrangian relaxed multiple choice knapsack-problem* (LRMCK). *Without loss of generality we may assume that the cost coefficients* c_{kj} *are monotone increasing within each class. Then all solutions to* LRMCK *have the form* $x_{ki} = \delta_{ij}$ *with* $j = j(k)$ *such that* $S_{kj} \geqq \lambda \geqq S_{k,j+1}$. (*Note that* λ *occurs in the definition of* LRMCK.)

Proof. By Lemma 2 we have only to show that for k fixed and all l

$$\lambda c_{kj} - r_{kj} \leqq \lambda c_{kl} - r_{kl}$$

or

$$\lambda (c_{kl} - c_{kj}) \geqq r_{kl} - r_{kj}.$$

We assume $l > j$ since the case $l < j$ can be dealt with similarly. Then we have to show

$\lambda \geqq \sigma_{\alpha\beta}$ with $\alpha = (k, j)$ and $\beta = (k, l)$. The case $l = j+1$ follows from our assumptions. Since the incremental profit densities S_{kj} are monotone decreasing in the second index the geometric lemma gives $\sigma_{\alpha\gamma} < \sigma_{\alpha\beta}$ for $\gamma = (k, j+i)$ and $i > 1$. This proves the theorem.

COROLLARY. *The solution to* LRMCK *is unique except for* $\lambda = S_{kj}$ *and some* (k, j).

Thus it makes sense to define

$$x_+(\lambda) \equiv \lim_{\varepsilon \to 0} \arg\min -\langle r, x\rangle + (\lambda + \varepsilon)(\langle c, x\rangle - w),$$

$$x_-(\lambda) \equiv \lim_{\varepsilon \to 0} \arg\min -\langle r, x\rangle + (\lambda - \varepsilon)(\langle c, x\rangle - w).$$

We have $x_+(\lambda) = x_-(\lambda)$ except for $\lambda = S_{kj}$ and some (k, j).

Let us now study the *dual functional*

$$D(\lambda) = \min_{x \in \mathrm{MC}} -\langle r, x\rangle + \lambda(\langle c, x\rangle - w).$$

LEMMA 7. $D(\lambda)$ *is piecewise linear and concave. The directional derivatives are given by* $D'_\pm(\lambda) = \langle c, x_\pm(\lambda)\rangle - w$.

COROLLARY. *For a value of* λ *which is not less than an optimal value the corresponding solution* $x_\pm(\lambda)$ *is feasible for the primal problem* MCK.

Proof of Lemma 7. For fixed x we see that $-\langle r, x\rangle + \lambda(\langle c, x\rangle - w)$ is linear in λ. Now the infimum of a family of linear functions is concave. Since there are only a finite number of lines (since $x \in \mathrm{MC}$) it is piecewise linear. Thus $D(\lambda)$ has a piecewise constant derivative and the jumps are the intersection points S_{kj}.

By the standard definition of a gradient of a convex (concave) function we have

$$\nabla D \equiv \begin{cases} \text{p} & \text{if } 0 \in [D'_+, D'_-] \subset \mathbb{R}, \\ \text{otherwise:} & \\ D'_+ & \text{if } |D'_+| \leqq |D'_-|, \\ D'_- & \text{if } |D'_-| \leqq |D'_+|. \end{cases}$$

Now maximizing $D(\lambda)$ can be simply done by looking up where the gradient ∇D vanishes. This is easy since ∇D is a monotone decreasing and piecewise constant function.

To formulate our final algorithm we need some more notation: according to the definition of MCK exactly one of the variables x_{kj} equals 1 in each class k. We say "class k is in state $s = s_k$" iff $x_{ks} = 1$.

For each class k we have:

s_k: state variable $0 \leqq s_k \leqq N$,

r_{ks}: profit of class k being in state s $(r_{k0} = 0)$,

c_{ks}: cost of realizing state s of class k $(c_{k0} = 0)$,

$\delta r(k, s) \equiv r_{k,s+1} - r_{k,s}$ (incremental profit),

$\delta c(k, s) \equiv c_{k,s+1} - c_{k,s}$ (incremental costs),

$\rho(k, s) \equiv \delta r(k, s)/\delta c(k, s)$ (incremental profit density).

ALGORITHM DMCK. Given the budget w and m multiple choice classes as above do:

STEP 0. Set all states to 0; $s_k = 0$, $1 \leqq k \leqq m$. Sort the classes such that $\rho(k, 0)$ is monotone decreasing. Set the total profit P to 0.

WHILE $\delta c(1, s_1) \leqq w$ DO step 1 and step 2.

STEP 1. $w \leftarrow w - \delta c(1, s_1)$
$P \leftarrow P + \delta r(1, s_1)$
$s_1 \leftarrow s_1 + 1$

STEP 2. Reinsert class $k = 1$ such that the list remains sorted with respect to $\rho(k, s_k)$.

Remarks.

—The Lagrangian variable λ is given only implicitly by $\rho(1, s_1)$.

—If there were upper bounds to the state variables one could add a step after Step 1 which removes class 1 from the list if s_1 has just been incremented beyond its bound.

—Step 0 and Step 2 can be implemented efficiently by not sorting the classes themselves but by maintaining a sorted list of pointers.

THEOREM 2. *The algorithm* DMCK *solves the dual multiple choice knapsack problem and gives an approximate but feasible solution for the primal problem* MCK.

Proof. According to Theorem 1 and Lemma 7 there are only a finite number of different Lagrange-multipliers λ to check since the gradient of the dual functional $\nabla D(\lambda)$ only changes at $\lambda = S_{k,j+1} = \rho(k, j)$.

Step 0 ensures that we start with the maximum of all $\rho(k, j)$. We proceed by taking the next smaller value of λ by Step 1 and Step 2 while our budget still lasts. By Lemma 7 this is equivalent to $D'_+(\lambda) \leqq 0$. When the algorithm exists the loop at $\lambda^* = \rho(1, s_1)$ we have $D'_+(\lambda^*) \leqq 0$ and $D'_-(\lambda^*) > 0$. Thus we have $\nabla D(\lambda^*) = 0$ and a corresponding feasible solution $x_+(\lambda^*)$ to LRMCK.

5. The overall algorithm. We start with an initial grid that has been specified by the user and models the geometry of the computational region to sufficient precision. We estimate the energy profit of each element and apply the DMCK algorithm to select those elements that should be refined at least once. Having refined those elements we normally solve for the current finite-element approximation on this grid before we proceed with computing the energy profit of the new elements and so on. This is called a long pass which is best but expensive. Therefore we optionally bypass this global solution process and solve only locally with those elements which have just been refined. The combination of long and short passes will be done on the basis of a user-specified "increase factor" (ICF). Whenever the number of elements has not yet increased by at least ICF times the number of elements within the last long pass we insert (another) short pass until the number of elements has increased sufficiently. Thus the user can force an arbitrary amount of new elements between long passes. Note that the discrete optimization process is invoked in both cases. The only penalty paid is a less accurate intermediate solution which harms the quality of the error estimator. Furthermore since we do not have an (inverse of a) valid coarse grid matrix in short passes we have to omit the coarse grid correction term on the right-hand side of (2.4).

We are now in a position to give the overall algorithm.

Perform the following steps:

(o) Input the initial grid, the budget w that we may spend and the increase factor ICF. Set the p-factors of all elements to 1. This corresponds to assuming

worst case regularity in the very beginning (see § 3). Assemble the element matrices and solve on the initial grid. This constitutes the first long pass. Next compute the energy profits according to (2.4). To get the terms in (2.4) we have to refine each element temporarily and compute the related element matrices. These can be stored and reused when the element is refined indeed.

(i) Invoke the algorithm DMCK to spend the "money" virtually.

(ii) Refine those elements (only once) which have been selected by DMCK to be refined at least once. In general this does not use all of the budget since elements are refined only once in contrast to the "plan" of the DMCK algorithm. We update the budget accordingly (see § 1). "Activate" the element-matrices for these new elements. Note these have been computed in step (o) or step (vi) before. If all of our budget has been used then solve the finite-element equations to final accuracy, perform the a posteriori error estimate given at the end of § 3 and STOP.

(iii) If the number of elements added since the last long pass is at least ICF times the number of elements at the last long pass, then GOTO step (v).
Otherwise proceed with a short pass:

(iv) To compute the nodal values at the added nodes within the just-created elements we do not solve the FEM-equations globally but proceed as follows (cf. [15]): assume "triangle" T has just been refined and thus replaced by its sons $T'_1, \cdots, T'_m$. We now solve a very small finite-element problem on $T'_1, \cdots, T'_m$ where the data on the boundary of T are now used to set up proper boundary values for this small problem. This procedure is sometimes called the "local inversion method" in the literature.
GOTO step (vi).

(v) Here we solve the global finite-element equations to get a solution on this intermediate space (this is called a long pass). The solution process might be done with somewhat lower accuracy if an iterative solver is implied.

(vi) Compute the energy profit for the new elements according to (2.4) but with one exception: if the last step was a short pass (i.e. control passed from step (iv) to this one) then we have no (inverse of the) coarse grid matrix at hand. Therefore we have to omit the coarse grid correction terms $\frac{1}{2}r_H^T\Delta_H$ from the right-hand side of (2.4). This corresponds to a pure local estimate.
GOTO step (i).

6. Numerical results. The suggested scheme for automatic grid adaption is applied to a one-dimensional problem only to avoid the nontrivial data handling. Although the "pattern of the regularity of the solution" and the problem of data handling strongly depends on the dimension of the problem the error estimator of §§ 2 and 3 as well as the transition operator given by §§ 1, 4 and 5 are clearly applicable in more dimensional problems and will hopefully produce similar results. Some tests concerning the energy profit (error estimation) have been done for two-dimensional problems in [11].

We use the following one-dimensional model problem,

$$-0.04y''+y=F(x), \qquad y(0)=y(1)=0 \tag{6.1}$$

with quadratic and conforming finite elements.

We always start with a single element on $[0, 1]$. The coarsest grid was plotted on top of the next finer grid and so on. We used continuous vertical lines to mark the element boundaries. Thus it is clearly visible when an element has been first created. Furthermore one can see that each element which has been subdivided on the way to

the finest grid has been done so nearly as early as possible. In the lower part of each figure we have plotted the fine grid finite element approximation and the force term (but) in a different scale.

The plotted grids are those computed by using long passes only (by setting the increase factor to 0). But with the single exception of Example 4 an increase factor of ICF = 1 which inserts short passes until the number of elements has doubled produces an identical grid sequence. The grids belonging to long passes have been marked by a "+".

For comparison we have applied the OMP-strategy which refines only one triangle at a time—the one with best profit density when adapted to our error estimator.

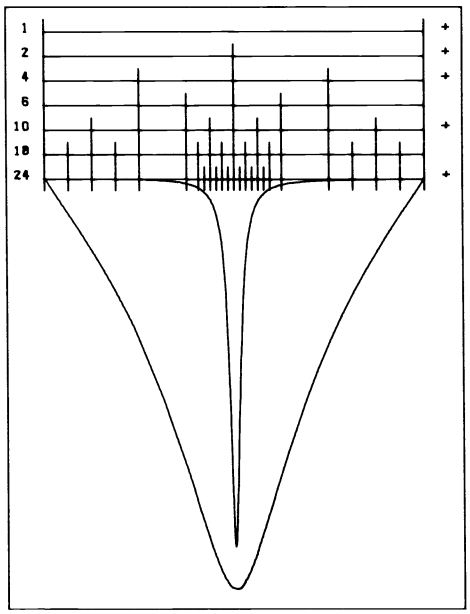


FIG. 1

Example 1 (Fig. 1). We start by using the "smooth" force term

$$F(x)=\frac{-0.1}{\pi[(x-0.5)^2+0.0025]}$$

which approximates the Dirac distribution at 0.5: $-2\delta_{0.5}$. The given budget amounts to 24 elements.

In the initial stage the region about 0.5 looks more promising than it turns out later on. Therefore the element nodes at 0.5 ± 0.375 appeared one stage too late. All remaining subdivisions have taken place as soon as possible. Note that up to 8 elements are refined simultaneously.

The resulting grid is identical to that obtained by the OMP strategy even if the increase factor is set to 1.

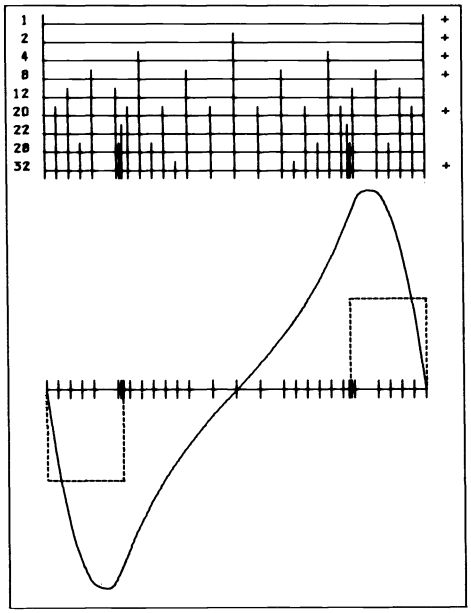


FIG. 2

Example 2 (Fig. 2). Here we use a step function as right-hand side which approximates the distribution $-2\delta_{0.1}+2\delta_{0.9}$:

$$F(x)=\begin{cases}\dfrac{-2}{0.2}, & 0\leqq x\leqq 0.2,\\ 0, & 0.2<x<0.8,\\ \dfrac{2}{0.2}, & 0.8\leqq x\leqq 1.\end{cases}$$

This time our budget amounts to 32 elements. The final grid is again identical to that obtained by the OMP strategy even with ICF = 1.

Since for this right-hand side one has the exact solution to (6.1) at hand, we look at the predicted and true energy profits in this example. The algorithm predicts the total energy profit which can be gained with the given budget. This is done each time a new intermediate space is at hand. In Table 1 these predictions are compared with the true differences between the energy of the final grid and the energy of the current grid. Furthermore for the finest grid (24 elements) we have computed the true error in energy and compared with the a posteriori estimate given in § 3. The significant overestimate of P for $m=1$ and $m=2$ comes from the fact that the order of convergence of the energy profit cannot yet be obtained by extrapolation and was assumed to be 0 (roughest possible case, cf. § 3).

Example 3 (Fig. 3). This example has a very rough force term. The problem (6.1) can only be posed in weak form with F "=" $-2\delta_{0.4}$. The given budget was 24 elements. The final grid is again identical to that given by the OMP-strategy even with ICF = 1.

Example 4 (Fig. 4). The last example has a force term consisting of the sum of 2 Dirac distributions of different strength F "=" $-3\delta_{0.3}+\delta_{0.8}$. The budget amounts to 32 elements. The adaptive grid identifies the "rough regions" very soon. The finest

TABLE 1
Total energy profit—predicted and exact.

m	$\tilde{P}$	P
1	9.9E 0	3.3E 0
2	3.7E 0	1.3E 0
4	5.7E−2	6.1E−2
8	2.7E−2	2.1E−2
12	6.5E−4	8.5E−4
20	4.9E−4	3.5E−4
22	1.5E−5	1.9E−5
28	6.3E−6	6.3E−6
32	3.7E−6	3.5E−6

m = number of elements of this intermediate grid.

P = difference in energy between current space and final space ($m = 32$). The value for $m = 32$ is the true error in energy to the continuous solution.

$\tilde{P}$ = predicted value of P. The value for $m = 32$ is the a posteriori estimate of the error to the continuous solution.

grid differed from that given by the OMP-strategy by only one element: the OMP-strategy did not refine the element [0.0, 0.125] but used 4 instead of 3 elements in the interval [0.2999, 0.3003] and achieved a somewhat smaller error in energy. This time an increase factor of ICF = 0.5 produced identical results, but with ICF = 1.0 we obtained a less optimal result. This is shown in Fig. 5.

7. Summary and conclusions. The problem of constructing an optimal triangulation for approximating an elliptic boundary value problem by finite elements is cast

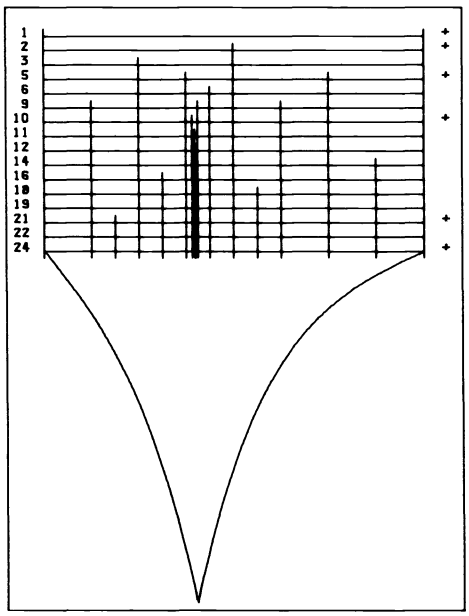


FIG. 3

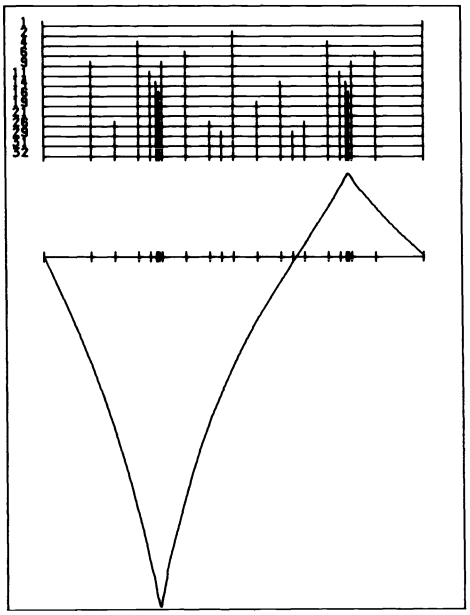


FIG. 4

in the form of determining that space for a given family which permits the lowest energy.

The difference in energy (profit) between a refined space and the original space behaves like the sum of the individual profits of each triangle which has been refined. Furthermore one can predict the future profits of a triangle by using a local extrapolation technique. Thus we can predict how many times each triangle will be refined on the path to the final grid. This has been modelled by a multiple choice knapsack problem.

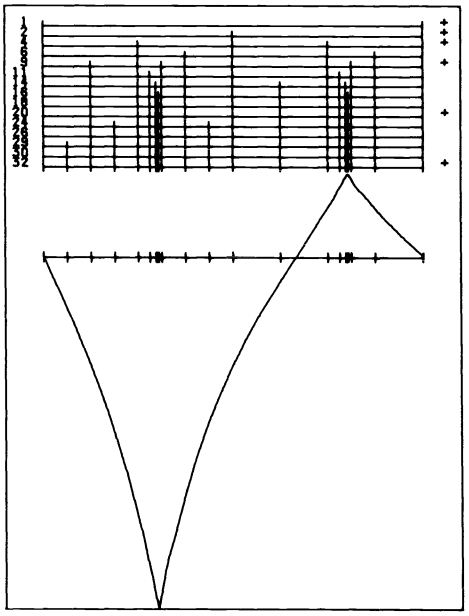


FIG. 5

Since the exact solution of that problem would have been much too expensive it was replaced by a partial dual of it which can be solved efficiently and does not change our model.

Since intermediate grids are expensive the user may specify the minimum amount of new elements that must be added between so-called "long passes" which compute a finite element approximation on the intermediate space.

The numerical examples show that the proposed technique produces grids nearly as good as those produced by the expensive strategy to refine only the most profitable element at a time. Since that strategy has proven optimality properties the proposed grid transition operator should work well in practice.

Furthermore the technique seems capable of giving a good a posteriori estimate of the error in energy on the final grid. Although the numerical examples were all one-dimensional the proposed technique is applicable to higher dimensional problems as well and it is hoped that similar results will be produced.

Acknowledgment. I wish to thank one of the anonymous referees for his or her very constructive criticism.

REFERENCES

[1] R. D. ARMSTRONG, D. S. KUNG, P. SINHA AND A. A. ZOLTNERS, *A computational study of a multiple choice knapsack algorithm*, ACM Trans. Math. Software, 9 (1983), pp. 184–198.

[2] I. BABUSKA, J. CHANDRA AND J. E. FLAHERTY, eds., *Proc. ARO Workshop on Adaptive Computational Methods for Partial Differential Equations*, Society for Industrial and Applied Mathematics, Philadelphia, PA, 1983.

[3] I. BABUSKA, A. MILLER AND M. VOGELIUS, *Adaptive methods and error estimation for elliptic problems of structural mechanics*, in [2, pp. 57–73].

[4] I. BABUSKA AND W. C. RHEINBOLDT, *A posteriori error estimates for the finite element method*, Internat. J. Numer. Meth. Engrg., 12 (1978), pp. 1597–1615.

[5] ———, *Error estimates for adaptive finite element computations*, SIAM J. Numer. Anal., 15 (1978), pp. 736–754.

[6] ———, *Analysis of optimal finite element meshes in* $\mathbb{R}^1$, Math. Comp., 33 (1979), pp. 435–463.

[7] ———, *Adaptive finite element processes in structural mechanics*, in Elliptic Problem Solvers II, G. Birkhoff and A. Schoenstadt, eds., Academic Press, New York, 1984.

[8] I. BABUSKA AND M. VOGELIUS, *Feedback and adaptive finite element solution of one-dimensional boundary value problems*, Numer. Math., 44 (1984), pp. 75–102.

[9] R. E. BANK, *The efficient implementation of local mesh refinement algorithms*, in [2, pp. 74–81].

[10] A. M. GEOFFRION, *Lagrangian relaxation for integer programming*, Math. Programming Study, 2 North-Holland, Amsterdam, 1974, pp. 82–114.

[11] R. NIEDERHAGEN, *Ein konstruktives Verfahren zur adaptiven Netzgenerierung bei finiten Elementen*, Diplomarbeit RWTH Aachen, 1983.

[12] W. C. RHEINBOLDT, *On a theory of mesh-refinement processes*, SIAM J. Numer. Anal., 17 (1980), pp. 766–778.

[13] ———, *Feedback systems and adaptivity for numerical computations*, in [2, pp. 3–19].

[14] W. C. RHEINBOLDT AND C. K. MESZTENYI, *On a data structure for adaptive finite element mesh refinements*, ACM Trans. Math. Software, 6 (1980), pp. 166–187.

[15] P. ZAVE AND G. E. COLE, JR., *A quantitative evaluation of the feasibility of, and suitable hardware architectures for an adaptive, parallel finite-element system*, ACM Trans. Math. Software, 9 (1983), pp. 217–292.

[16] P. ZAVE AND W. C. RHEINBOLDT, *Design of an adaptive, parallel finite-element system*, ACM Trans. Math. Software, 5 (1979), pp. 1–17.

[17] O. C. ZIENKIEWICZ AND A. W. CRAIG, *Adaptive mesh refinement and a posteriori error estimation for the p-version of the finite element method*, in [2, pp. 33–56].

SIAM J. SCI. STAT. COMPUT.
Vol. 7, No. 4, October 1986

004

COMPUTING THE CS-DECOMPOSITION ON SYSTOLIC ARRAYS*

FRANKLIN T. LUK† AND SANZHENG QIAO‡

Abstract. We describe a new parallel algorithm for computing the CS-decomposition, and compare it against a recently published method of Kaplan and Van Loan. For a $2n \times n$ orthonormal matrix that is partitioned into two square blocks, their procedure needs $O(n^2)$ time and $O(n^2)$ processors (probably in the form of two distinct arrays), whereas our procedure requires $O(n \log n)$ time and one triangular array of $O(n^2)$ processors.

Key words. systolic arrays, CS-decomposition, Jacobi-type methods, real-time computation, VLSI

AMS(MOS) subject classifications. 65F15, 68A10

1. Introduction. In this paper we describe an almost linear-time algorithm for computing a CS-decomposition (CSD) of an $(n+m) \times p$ partitioned orthonormal matrix Q, i.e.,

$$Q \equiv \begin{pmatrix} Q_1 \\ Q_2 \end{pmatrix}_{m \times p}^{n \times p} \tag{1}$$

with $n \geqq p$, $n \geqq m$ and $Q_1^T Q_1 + Q_2^T Q_2 = I_p$. The decomposition is simply a simultaneous diagonalization of both blocks Q_1 and Q_2 via the singular value decomposition (SVD):

$$U_1^T Q_1 V = C \equiv \operatorname{diag}(c_1, \cdots, c_p), \tag{2}$$

$$U_2^T Q_2 V = S \equiv \operatorname{diag}(s_1, \cdots, s_q), \tag{3}$$

where $q = \min\{p, m\}$, and $U_1(n \times n)$, $U_2(m \times m)$, $V(p \times p)$ are orthogonal matrices. Since $C^T C + S^T S = I_p$, it follows that

$$\begin{aligned} c_i^2 + s_i^2 &= 1, \qquad i = 1, \cdots, q, \\ c_i &= \pm 1, \qquad i = q+1, \cdots, p. \end{aligned} \tag{4}$$

We may regard the singular values of Q_1 and Q_2 as cosines and sines, accounting for the name of the decomposition. Various uses of the CSD in analyzing invariant subspace perturbation problems are given in [3], [12], [15]. Paige and Saunders [11] showed that computing the CSD can lead to a sound algorithm for the generalized singular value decomposition (GSVD). However, direct GSVD methods were recently developed by Paige [10] and Luk [8].

Both Stewart [13] and Van Loan [16] presented stable CSD algorithms. The latter procedure is simpler in that no cross-product matrix is required. Its implementation on systolic arrays was sketched by Kaplan and Van Loan [4]. If we assume $q = O(n)$, we may say that their technique requires $O(n^2)$ processors (possibly in the form of two distinct arrays) and $O(n^2)$ time. The purpose of this paper is to describe a different parallel CSD algorithm that uses $O(n+ps)$ time and one triangular array of $O(p^2)$ processors. The parameter s denotes the number of sweeps required by a Jacobi-SVD algorithm. We conjecture that $s = O(\log p)$ and that s is bounded by a small constant (say $\leqq 10$) for practical values of p (say $\leqq 200$) [1], [2], [7].

* Received by the editors February 5, 1985, and in revised form September 19, 1985.

† School of Electrical Engineering, Cornell University, Ithaca, New York 14853. The work of this author was supported in part by the Office of Naval Research under contract N00014-85-K-0074.

‡ Center for Applied Mathematics, Cornell University, Ithaca, New York 14853. The work of this author was supported by a Cornell University Sage Graduate Fellowship.

2. CSD algorithm. We sketch Van Loan's algorithm [16] on the assumption that $m \geqq p$. A "normalized" SVD of Q_2 is computed:

$$U_2^T Q_2 V = S \equiv \operatorname{diag}(s_1, \cdots, s_p), \tag{5}$$

where $U_2(m \times m)$ and $V(p \times p)$ are orthogonal matrices. By a "normalized" SVD, we mean that the singular values are nonnegative and sorted in ascending order:

$$0 \leqq s_1 \leqq \cdots \leqq s_k \leqq 1/\sqrt{2} < s_{k+1} \leqq \cdots \leqq s_p. \tag{6}$$

Let us assume that k singular values are "small" ($\leqq 1/\sqrt{2}$). A QR-decomposition (QRD) of the product $Q_1 V$ is made:

$$U_1^T (Q_1 V) = \begin{pmatrix} R \\ 0 \end{pmatrix},$$

where $U_1(n \times n)$ is orthogonal and $R(p \times p)$ is upper triangular. It is proved in [16] that

$$R = \begin{pmatrix} \operatorname{diag}(c_1, \cdots, c_k) & 0 \\ 0 & R_1 \end{pmatrix},$$

where R_1 is a $(p-k) \times (p-k)$ triangular matrix. That is, the first k ($k+1$ to be exact since R_1 is upper triangular) columns of $Q_1 V$ have been diagonalized by a QRD. Note from (4) that $c_1, \cdots, c_k$ are "large" singular values. Next, an SVD of R_1 is found:

$$\tilde{U}_1^T R_1 \tilde{V} = \operatorname{diag}(c_{k+1}, \cdots, c_p),$$

and a QRD of the $(p-k) \times (p-k)$ matrix $W \equiv D\tilde{V}$, with $D = \operatorname{diag}(s_{k+1}, \cdots, s_p)$, is computed:

$$\tilde{U}_2^T W = R_2.$$

Since $s_{k+1}, \cdots, s_p$ are "large" we get $R_2 = \operatorname{diag}(s_{k+1}, \cdots, s_p)$. The resulting CS-decomposition is

$$\begin{pmatrix} I_k & 0 & 0 \\ 0 & \tilde{U}_1^T & 0 \\ 0 & 0 & I_{n-p} \end{pmatrix} U_1^T Q_1 V \begin{pmatrix} I_k & 0 \\ 0 & \tilde{V} \end{pmatrix} = \operatorname{diag}(c_1, \cdots, c_p),$$

$$\begin{pmatrix} I_k & 0 & 0 \\ 0 & \tilde{U}_2^T & 0 \\ 0 & 0 & I_{m-p} \end{pmatrix} U_2^T Q_2 V \begin{pmatrix} I_k & 0 \\ 0 & \tilde{V} \end{pmatrix} = \operatorname{diag}(s_1, \cdots, s_p).$$

The crucial step of this algorithm is the "normalized" SVD step of (5). It separates each of Q_1 and Q_2 into two different blocks of "small" and "large" singular values. The key result of Van Loan [16] is that a block consisting solely of "large" singular values is diagonalized by a QRD.

3. Parallel implementations. From § 2 we see that we need both QRD and SVD processor arrays. Kaplan and Van Loan [4] selected the square SVD array of Brent-Luk-Van Loan [2] and the square QRD array of Luk [6]. Few details are given in [4], where it is stated that "important questions remain with respect to the parallel array implementation." Kaplan and Van Loan chose to compute a "normalized" SVD via the "parallel ordering" of [1]. Specifically, the ordering was used to first determine an "unnormalized" SVD and then to sort the singular values. The sorting part is unacceptable, for $q/4+1$ sweeps are required for sorting q singular values.

On the other hand, the triangular QRD-SVD array of Luk [7] is well-suited for implementing Van Loan's CSD algorithm. This array of $O(p^2)$ processors computes a QRD of an $n \times p$ matrix ($n \geqq p$) in time $(n+2p-2)\tau$, where τ is the time required for a 2×2 QRD. The array uses a Jacobi-SVD method to diagonalize the resultant triangular matrix, say R. Rotations are restricted to adjacent rows and columns of R (via an "odd-even ordering"), in order to preserve its triangular structure. Explicit row and column permutations are replaced by the choice of large rotation angles, called "outer rotations." The "odd-even ordering" and "outer rotations" were used by Stewart [14] to compute the Schur decomposition, and later by Luk [7] to find an SVD of a triangular matrix. The SVD algorithm is very simple:

Procedure SVD(R).
do until convergence
  for $i = 1, 3, 5, \cdots, 2, 4, 6, \cdots$ do
    $R \leftarrow J_{i,i+1}^T R K_{i,i+1}$.

The transformations $J_{i,i+1}$ and $K_{i,i+1}$ are rotations in the $(i, i+1)$ plane implementing the 2×2 SVDs. For a $p \times p$ matrix R one sweep ($= p(p-1)/2$ transformations) of Procedure SVD requires time approximately $2p\sigma$, where σ denotes the time required for a 2×2 SVD [7]. The QRD-SVD array thus determines an SVD of an $n \times p$ matrix A in $O(n+ps)$ time, by first reducing A to an upper triangular form R and then diagonalizing R.

We now show how the QRD-SVD array can be used to compute a CSD of Q. To simplify the presentation we assume $m \geqq p$, by adding a block of zeros to the bottom of Q_2 if necessary. Our initial step is to determine QRDs of the matrices Q_1 and Q_2:

Procedure QRD (Q_1, Q_2, R_1, R_2).
  begin

$$W_1^T Q_1 = \begin{pmatrix} R_1 \\ 0 \end{pmatrix};$$

$$W_2^T Q_2 = \begin{pmatrix} R_2 \\ 0 \end{pmatrix}$$

  end.

The matrices $W_1(n \times n)$, $W_2(m \times m)$ are orthogonal, and $R_1(p \times p)$, $R_2(p \times p)$ are upper triangular. This step can be completed in time $(n+m+2p-2)\tau$ if the rows of Q_2 are fed into the array immediately after those of Q_1. Next, compute an SVD of R_2:

$$U_2^T R_2 V = S,$$

and a QRD of the product $R_1 V$:

$$U_1^T (R_1 V) = \tilde{R}_1.$$

The two steps are performed by the following procedure with $j = 0$.

Procedure SVD-QRD (R_2, R_1, j).
do until convergence of SVD algorithm
  for $i = j+1, j+3, j+5, \cdots, j+2, j+4, j+6, \cdots$ do
    begin
    $R_2 \leftarrow J_{i,i+1}^T R_2 K_{i,i+1}$; {SVD}
    $R_1 \leftarrow P_{i,i+1}^T (R_1 K_{i,i+1})$ {QRD}
    end.

The rotations $P_{i,i+1}$ are chosen to maintain the triangular structure of R_1. An advantage of this procedure is that the QRD operations are free. Recall from [7], [14] that, to avoid broadcasting, at most half the processors are busy at any time. So we shall attain full efficiency if we stagger the SVD and QRD computations in an obvious manner. One sweep of Procedure SVD-QRD $(R_2, R_1, 0)$ is then implementable in time $2p\sigma$. After the procedure terminates, we get

$$R_2 \leftarrow S \equiv \text{diag}\,(s_1, \cdots, s_p).$$

The singular values $s_1, \cdots, s_p$ must be negated (if necessary) and sorted to give a "normalized" SVD. To save work, we decide to allow negative singular values and order them on size alone using the "odd-even ordering":

```
Procedure Sort(S).
do for one sweep
  for i = 1, 3, 5, ···, 2, 4, 6, ··· do
    if |s_i| > |s_{i+1}| then exchange s_i and s_{i+1}.
```

This procedure is called an "odd-even transposition sort" [5]. It requires one sweep ($=p$ time steps) to sort p numbers, and is optimal for sorting networks that involve only adjacent comparisons. However, we need to preserve the triangular structure of R_1. Procedure Sort is extended to a "normalized" SVD scheme:

```
Procedure NSVD (R_2, R_1).
do for one sweep
  for i = 1, 3, 5, ···, 2, 4, 6, ··· do
    if |s_i| > |s_{i+1}| then
      begin
      R_2 ← Π^T_{i,i+1} R_2 Π_{i,i+1};
      R_1 ← P̃^T_{i,i+1} (R_1 Π_{i,i+1})
      end.
```

The matrix $\Pi_{i,i+1}$ is a permutation in the $(i, i+1)$ plane, while $\tilde{P}_{i,i+1}$ is a rotation chosen to triangularize the product $R_1\Pi_{i,i+1}$. Recall that k denotes the number of "small" singular values of R_2. We invoke the procedure SVD-QRD (R_1, R_2, k) to complete the CSD algorithm:

```
Program CSD (Q_1, Q_2).
  begin
  QRD (Q_1, Q_2, R_1, R_2);
  SVD-QRD (R_2, R_1, 0);
  NSVD (R_2, R_1);
  SVD-QRD (R_1, R_2, k)
  end.
```

The overall algorithm requires $O(n+ps)$ time. A numerical example is given in [9].

Acknowledgments. The authors acknowledge valuable suggestions by A. Sameh and P. Eberlein.

REFERENCES

[1] R. P. BRENT AND F. T. LUK, *The solution of singular-value and symmetric eigenvalue problems on multiprocessor arrays*, this Journal, 6 (1985), pp. 69–84.

[2] R. P. BRENT, F. T. LUK AND C. VAN LOAN, *Computation of the singular value decomposition using mesh-connected processors*, J. VLSI Computer Systems, 1 (1985), pp. 242–270.

[3] C. DAVIS AND W. M. KAHAN, *The rotation of eigenvectors by a perturbation* III. SIAM J. Numer. Anal., 7 (1970), pp. 1–46.

[4] I. M. KAPLAN AND C. VAN LOAN, *On computing the CS-decomposition with systolic arrays*, Tech. Report TR84-647, Computer Science Dept., Cornell Univ., Ithaca, NY, 1984.

[5] D. E. KNUTH, *The Art of Computer Programming, Vol.* 3, *Sorting and Searching*, Addison-Wesley, Reading, MA, 1973.

[6] F. T. LUK, *A rotation method for computing the QR-decomposition*, this Journal, 7 (1986), pp. 452–459.

[7] ———, *A triangular processor array for computing singular values*, Linear Algebra Appl., to appear.

[8] ———, *A parallel method for computing the generalized singular value decomposition*, J. Parallel Distrib. Comput., 2 (1985), pp. 250–260.

[9] F. T. LUK AND S. QIAO, *A linear time method for computing the CS-decomposition*, Tech. Report EE-CEG-84-6, School of Electrical Engineering, Cornell Univ., Ithaca, NY, 1984.

[10] C. C. PAIGE, *Computing the generalized singular value decomposition*, this Journal, to appear.

[11] C. C. PAIGE AND M. A. SAUNDERS, *Toward a generalized singular value decomposition*, SIAM J. Numer. Anal., 18 (1981), pp. 398–405.

[12] G. W. STEWART, *On perturbation of pseudo-inverses, projections, and linear least squares problems*, SIAM Rev., 19 (1977), pp. 634–662.

[13] ———, *An algorithm for computing the CS decomposition of a partitioned orthonormal matrix*, Numer. Math., 40 (1983), pp. 297–306.

[14] ———, *A Jacobi-like algorithm for computing the Schur decomposition of a non-Hermitian matrix*, this Journal, 6 (1985), pp. 853–864.

[15] C. VAN LOAN, *Analysis of some matrix problems using the CS decomposition*, Tech. Report TR84-603, Computer Science Dept., Cornell Univ., Ithaca, NY, 1984.

[16] C. VAN LOAN, *Computing the CS and the generalized singular value decompositions*, Numer. Math., 46 (1985), pp. 479–491.

SIAM J. SCI. STAT. COMPUT.
Vol. 7, No. 4, October 1986

COMPUTING THE GENERALIZED SINGULAR VALUE DECOMPOSITION*

C. C. PAIGE†

Abstract. An algorithm is described for computing the generalized singular value decomposition of $A(m\times n)$ and $B(p\times n)$. Unitary matrices U, V and Q are developed so that U^HAQ and V^HBQ have as many nonzero parallel rows as possible, and these correspond to the common row space of the two matrices. The algorithm consists of an iterative sequence of cycles where each cycle is made up of the serial application of 2×2 generalized singular value decompositions. Convergence appears to be at least quadratic. With the correct choice of ordering the algorithm can be implemented using systolic array processors (Gentleman, personal communication). The algorithm can also be used to compute any CS decomposition of a unitary matrix.

Key words. generalized singular values, CS decomposition, unitary matrices, matrix decompositions, matrix parallel computations

AMS(MOS) subject classifications. Primary 65F30; secondary 15-04, 15A03, 15A18, 15A23, 65F25

1. Introduction.

The generalized singular value decomposition (GSVD) was introduced by van Loan [18]. Paige and Saunders [15] extended van Loan's idea in order to handle all possible cases, and presented a form of the decomposition which was more suitable for computation than that in [18]. This is the GSVD we will consider here. Van Loan [18] discussed several uses of the GSVD, for example the singular value pairs (α, β), see [15], of $A(m\times n)$ and $B(p\times n)$ solve $\det(\beta^2A^HA-\alpha^2B^HB)=0$. Paige [14] gave an expository introduction to the GSVD and its relation to the general Gauss-Markov linear model (y, Xb, σ^2FF^T) used in regression analysis, and showed how it not only reveals the structure and sensitivity of the model, but provides the best linear unbiased estimator and the structure of the covariance of the error of this estimator. For these and other problems there is a need for a good algorithm for computing the GSVD.

When B is square and nonsingular the GSVD of $A(m\times n)$ and $B(n\times n)$ corresponds to the singular value decomposition (SVD) of AB^{-1}. If A or B is ill conditioned with respect to solution of equations, then computing AB^{-1} would usually lead to unnecessarily large numerical errors, and so this approach is not recommended in general. When B is not square, or is singular, then the SVD of AB^+, where B^+ denotes the Moore-Penrose pseudoinverse of B, does not necessarily correspond to the GSVD of A and B, and a different approach is definitely needed.

If $[A^H, B^H]^H$ is a block of columns of a unitary matrix then the GSVD of A and B gives what is called the CS decomposition for this partitioning; see Stewart [17]. This decomposition is implicit in the work of Davis and Kahan [3]. It was stated explicitly in [16] for the slightly restricted case of $A(n\times n)$, and was derived in [15] for the general case, where it was seen to provide one approach to computing the GSVD of general $A(m\times n)$ and $B(p\times n)$. Briefly that approach computes the SVD or similar unitary factorization of general

$$\begin{pmatrix}A\\B\end{pmatrix}=P_1RQ_1^H,\qquad P_1=\begin{pmatrix}P_{11}\\P_{21}\end{pmatrix},\tag{1.1}$$

$R(k\times k)$ square and nonsingular, $P_1^HP_1=Q_1^HQ_1=I$; followed by the CS decomposition

* Received by the editors June 21, 1984, and in revised form August 15, 1985.

† Computer Science, McGill University, Montreal, Quebec, Canada H3A 2K6. This work was supported by the Canadian Natural Sciences and Engineering Research Council under grant A8652.

of $P_{11}(m \times k)$ and $P_{21}(p \times k)$. Methods for computing the CS decomposition are given by Stewart [17] and van Loan [19], and these can be combined with (1.1) to produce the GSVD of A and B, see [15].

The above two stage process for computing the GSVD has definite drawbacks. First A and B are combined in (1.1), an approach which ignores the good numerical practice of applying unitary transformations to A and B separately where possible. Distinctly different scaling of A and B could lead to difficulties. Next a rank decision must be made in (1.1), and so it is advisable to use the SVD; but again this rank decision could be affected by the relative sizes of A and B. This results in two separate iterative algorithms with an important and possibly difficult rank decision between them, and in difficult cases there will be no clear criteria on which to make this rank decision. Accordingly we have sought one unified iterative algorithm which applies unitary transformations to A and B separately.

One approach [1] implicitly applied the QR algorithm for the SVD in [5] to the equivalent of AB^{-1}, somewhat like the QZ algorithm of Moler and Stewart [13] does for the generalized eigenvalue problem. Although this appeared to work quite satisfactorily, it was so complicated we chose not to publish it or to pursue it further. To obtain some idea of the difficulty, it suffices to note that the QR algorithm for the SVD of a given matrix C in [5] has two levels of implicitness, one because it implicitly applies the QR algorithm to C^HC while working with C rather than forming C^HC, and a second because it carries out implicit shifts in these QR steps. If now we want the SVD of $C = AB^{-1}$ without forming AB^{-1}, this gives a third level of implicitness. In contrast the QZ algorithm only has two levels of implicitness.

The simplest unified approach we have found comes from an idea of Kogbetliantz [10], [11], for finding the singular value decomposition of a square matrix. Just as Jacobi's method [9] for computing the eigenvalues of a symmetric matrix solves a sequence of 2×2 symmetric eigenproblems, Kogbetliantz' method solves a sequence of 2×2 SVD problems. It is beautifully simple and surprisingly fast, and apparently has ultimate quadratic convergence. This has been proven when there are no pathologically close singular values [21]. However it is not usually as fast as the method of [5] and [6]. The fine analysis of Forsythe and Henrici [4] not only proved convergence of the serial-cyclic Jacobi method for the symmetric eigenproblem, but also of the serial-cyclic Kogbetliantz method, both under restrictions on the choice of the angles of rotation at each step. Kogbetliantz' method is discussed further in [8].

The method to be described here computes U^HAQ and V^HBQ, where unitary U, V and Q are built up in an iterative manner which when for example B is square and nonsingular corresponds to a refinement of Kogbetliantz' method of transforming $U^HAB^{-1}V$ to diagonal form. There is clearly only one level of implicitness in this. The method is similar in approach to that in [8], and owes much to the work on that paper. In fact reference will be made to that work where appropriate, instead of repeating some material here.

In § 2 we give a brief account of the serial-cyclic version of Kogbetliantz' algorithm applied to a general square matrix, and then to the special case of an upper triangular matrix, an improvement suggested in [8]. We then show how these ideas can be used to compute the GSVD of square A and B when one is nonsingular. This is the core of the algorithm, and in § 3 it is extended to give an algorithm for general A and B. Section 4 shows that this extension gives the correct result for 2×2 matrices, and § 5 uses this to show how the algorithm behaves for general A and B, a summary of the algorithm being included for ease in following the proofs. Section 6 contains numerical examples, and some comments are made in § 7. Finally as a result of ideas of Gentleman

[22], it is shown how the algorithm given here can be implemented for parallel computation in § 8. His approach is particularly suitable for systolic array processors.

The GSVD algorithm proposed here can be applied directly to the relevant submatrices of a unitary matrix to obtain any required CS decomposition.

2. The basic algorithm. Kogbetliantz [11] computes the SVD of a square matrix in an iterative sequence of cycles. In one cycle of the serial-cyclic method, $n(n-1)/2$ SVDs of 2×2 matrices centred on the diagonal are computed to eliminate elements in the following order ($n=4$),

$$\begin{pmatrix} 0 & 1 & 2 & 3 \\ 1 & 0 & 4 & 5 \\ 2 & 4 & 0 & 6 \\ 3 & 5 & 6 & 0 \end{pmatrix}. \tag{2.1}$$

We call this the row ordering; there is obviously a column version. The elements marked 1 are eliminated first, then those marked 2, in which case those marked 1 could reappear. Of course, the diagonal elements are not eliminated; in fact their sum of squares does not decrease, and this, together with a restriction we will discuss, ensures the convergence of the algorithm to diagonal form [4]. Convergence is found in practice to be at least quadratic, and 5 cycles are often sufficient to give full accuracy on for example the VAX 11/750 using double precision.

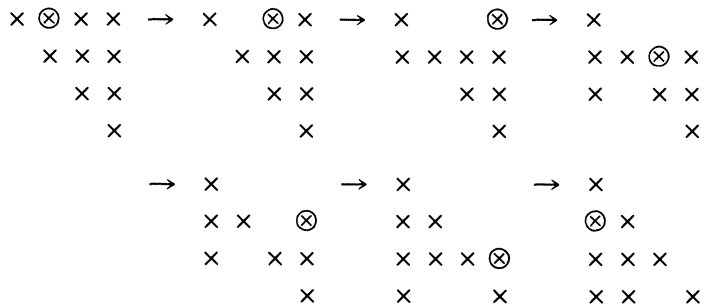
The unitary reduction of a 2×2 matrix to diagonal form requires a unitary rotation from the left through an angle ϕ say, and one of the right through ψ say. If the serial-cyclic approach (2.1), or the column equivalent, is used then [4] shows the $n\times n$ matrix converges to diagonal form if each

$$\phi \in J, \qquad \psi \in J, \tag{2.2}$$

where J is a fixed closed interval interior to the open interval $(-\pi/2, \pi/2)$. So a method which just requires the singular values of the 2×2 diagonal to be ordered in a fixed way does not necessarily satisfy this condition.

If the initial matrix is upper triangular then one cycle takes the following form, where now only one element is eliminated by each 2×2 SVD, and is circled immediately prior to elimination.

(2.3)



This leads to lower triangular form. The next cycle uses exactly the same ordering as (2.1) and produces an upper triangular matrix again This refinement leads to a saving

in computation, and since this is really the same method as for the full case, the same comments on convergence hold. A more complete description of these algorithms is given in [8].

Now we present a method which we will then show implicitly carries out one cycle of Kogbetliantz' algorithm on upper triangular $C = AB^{-1}$, when both A and B are initially upper triangular and B is nonsingular. We first describe what we will refer to as the (i, j) transformation. If A_{ij} and B_{ij} are 2×2 matrices, whose elements lie in rows i and j and columns i and j of A and B, and U and V are unitary so that

$$U^H A_{ij} B_{ij}^{-1} V = S \tag{2.4}$$

is diagonal, then

$$U^H A_{ij} = S V^H B_{ij}. \tag{2.5}$$

The dependence of these 2×2 U, V and S on i and j will not be stated explictly in these first four sections.

As a result, the first row of $U^H A_{ij}$ is parallel to the first row of $V^H B_{ij}$, and the second row of $U^H A_{ij}$ is parallel to that of $V^H B_{ij}$. Thus if Q is unitary so that $V^H B_{ij} Q$ is lower triangular, that is

$$(V^H B_{ij}) Q = \begin{pmatrix} \times & \otimes \\ \times & \times \end{pmatrix} = \begin{pmatrix} \times & \\ \times & \times \end{pmatrix}, \tag{2.6}$$

then $U^H A_{ij} Q$ is also lower triangular. For n by n upper triangular $C = AB^{-1}$ we carry out $n(n-1)/2$ such 2×2 transformations in the same order as in (2.1) and (2.3). The kth such transformation acting on its relevant 2×2 submatrices takes the form, with C_{ij} defined to be $A_{ij} B_{ij}^{-1}$,

$$\begin{array}{ccccc} A_{ij} & = & C_{ij} & \cdot & B_{ij} \\ & & \overset{k'}{\curvearrowleft} & & \\ k\begin{pmatrix} \times & \times \\ \square & \times \end{pmatrix} & = k\begin{pmatrix} \times & \otimes \\ & \times \end{pmatrix} & \cdot & k'\begin{pmatrix} \times & \times \\ \square & \times \end{pmatrix} \\ \overset{k''}{\curvearrowleft} & & & & \overset{k''}{\curvearrowleft} \\ \begin{pmatrix} \times & \otimes \\ \times & \times \end{pmatrix} & = & \begin{pmatrix} \times & \\ & \times \end{pmatrix} & \cdot & \begin{pmatrix} \times & \otimes \\ \times & \times \end{pmatrix} \end{array} \tag{2.7}$$

where k corresponds to U^H, k' corresponds to V^H, and k'' corresponds to Q.

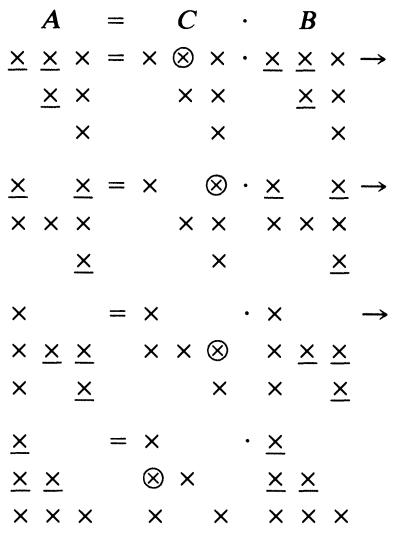
Note that we would design Q on the larger of the two possible rows in the figure. This is the obvious choice for numerical precision, but should be supported by a rounding error analysis. This kth transformation will be denoted

$$\begin{array}{ccccccccc} \underline{\times}\ \underline{\times} & = & \times\ \times & \cdot & \underline{\times}\ \underline{\times} & \rightarrow & \times & = \times & \cdot\ \times \\ \underline{\times} & & \times & & \underline{\times} & & \times\ \times & \times & \times\ \times \end{array} \tag{2.8}$$

Using this notation with that of (2.3), and taking $n = 3$ for illustration, one cycle starting

with upper triangular A and B takes the form

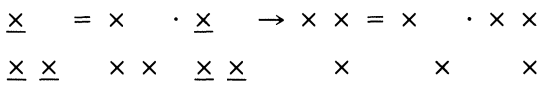
(2.9)



which results in lower triangular A and B. It can be seen from the above forms of A and B that the elements of the 2×2 $C_{ij}=A_{ij}B_{ij}^{-1}$ are just the elements in rows and columns i and j of $C=AB^{-1}$. The proof that this is true for larger n follows by induction, noting that after the off-diagonal elements in the first row of each of A and B have been eliminated, the remaining rotations are designed on the $n-1\times n-1$ upper triangular matrices in the bottom right-hand corners of A and B. When A and B are lower triangular the process is essentially the same and follows the ordering in (2.1), except now the equivalent of (2.6) is

$$(V^H B_{ij})Q=\begin{pmatrix}\times & \times\\ \otimes & \times\end{pmatrix}=\begin{pmatrix}\times & \times\\ & \times\end{pmatrix} \tag{2.10}$$

and the equivalent of (2.8) becomes



so that at the end of the cycle the A and B matrices are upper triangular again.

A comparison of C in (2.9) with (2.3) shows that this process is mathematically equivalent to applying Kogbetliantz' serial-cyclic method to upper triangular $C=AB^{-1}$, and all the previous comments on convergence hold. As a result we have given a method for finding the GSVD of two matrices, at least one of which is nonsingular. It will be seen in § 6 that this is very effective, giving as much accuracy as the problem and computer allows, in a reasonable time.

The computational cost per cycle is about the same as the method in [8] for finding the SVD of the product of two triangular matrices. That is, if we use r-multiplication rotations, one cycle as in (2.9) costs about rn^3 multiplications and $2n^3$ additions just to form the new A and B. In addition it would require $rn^3/2$ multiplications and n^3 additions for each of U, V, and Q that we chose to update. Since the method may sometimes take more than 5 cycles to converge, but rarely much more, it can be quite expensive for large n.

To counter-balance this expense it is clear that such methods are ideal for special architectures such as systolic arrays, and this can result in very efficient implementations.

Kogbetliantz' SVD method [11] for a full square matrix A has been suggested by Brent, Luk and van Loan [2] for systolic array computation. Their resulting method requires $O(n^2)$ processors and $O(n \log n)$ time to execute. On hearing the ideas presented here, Morven Gentleman [22] showed how the algorithms of the present paper can be implemented very effectively using systolic array processors.

3. The general case. For A and B square and B nonsingular we saw convergence of the method in § 2 gave $U^H A B^{-1} V = S$, a diagonal matrix, with $V^H B Q$ triangular. That is

$$U^H A Q = S V^H B Q, \qquad S = \operatorname{diag}(\sigma_1, \cdots, \sigma_n), \tag{3.1}$$

with the ith row of $U^H A Q$ parallel to the ith row of $V^H B Q$. If we ordered the singular values $\sigma_1 \geqq \sigma_2 \geqq \cdots \geqq \sigma_n \geqq 0$, and A had rank s, then $U^H A Q$ would be zero in all but the first s rows, and so would be upper trapezoidal if the algorithm took an even number of cycles.

For simplicity we will take A and B square, by initial unitary reduction to upper trapezoidal form and adding zero rows if necessary, and extend the algorithm to handle singular B. We will describe an algorithm which effectively produces on convergence after an even number of cycles, nonsingular upper triangular R and positive definite S_A and S_B such that

$$\underset{k\times k}{R} = \begin{pmatrix} R_{11} & R_{12} & R_{13} \\ 0 & R_{22} & R_{23} \\ 0 & 0 & R_{33} \end{pmatrix} \begin{matrix} \}r_1 \\ \}r_2, \\ \}s \end{matrix} \qquad r = r_1 + r_2, \tag{3.2}$$

$$S_A = \operatorname{diag}(\alpha_{r_1+1}, \cdots, \alpha_r), \qquad S_B = \operatorname{diag}(\beta_{r_1+1}, \cdots, \beta_r), \tag{3.3}$$

$$S_A^2 + S_B^2 = I, \tag{3.4}$$

$$U^H A Q = \begin{pmatrix} R_{11} & R_{12} & R_{13} & 0 \\ 0 & S_A R_{22} & S_A R_{23} & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}, \tag{3.5}$$

$$V^H B Q = \begin{pmatrix} 0_B & 0 & 0 & 0 \\ 0 & S_B R_{22} & S_B R_{23} & 0 \\ 0 & 0 & R_{33} & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}. \tag{3.6}$$

Here 0_B is an $r_1 \times r_1$ zero matrix so $[S_B R_{22}, S_B R_{23}]$ in (3.6) is in the same position as $[S_A R_{22}, S_A R_{23}]$ in (3.5). These then correspond to the nonzero parallel rows of $U^H A Q$ and $V^H B Q$, in analogy with (3.1). It follows from the theorem in [15, § 2] that this is the GSVD of A and B each with n columns.

To obtain (3.5) and (3.6) for any $A(m \times n)$ and $B(p \times n)$ we first transform with unitary matrices U and V and a permutation matrix Q so that $U^H A Q$ and $V^H B Q$ are upper trapezoidal; see for example [7]. To the resulting nonzero trapezoids we add zero rows if necessary to give square matrices. This is not essential but it simplifies the description and we can apply the algorithm very much as in § 2. We will assume the initial transformations we start with are identically partitioned

$$\underset{n\times n}{A} = \begin{pmatrix} A_{11} & A_{12} & A_{13} \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \begin{matrix} \}r \\ \\ \\ \end{matrix}, \qquad \underset{n\times n}{B} = \begin{pmatrix} B_{11} & B_{12} & B_{13} \\ 0 & B_{22} & B_{23} \\ 0 & 0 & 0 \end{pmatrix} \begin{matrix} \}r \\ \}q \\ \\ \end{matrix}, \tag{3.7a}$$

(3.7b) A_{11}, B_{11}, B_{22} are upper triangular, A_{11} is nonsingular, the nonzero part of B is at the top and has full row rank, and if $q>0$, B_{22} is nonsingular.

This can be done by designing the permutations in Q first during the reduction of A until A_{11} is produced, then from the part of the reduction of B that gives B_{22}.

The only reason the algorithm of § 2 cannot be applied directly here is that some of the 2×2 matrices B_{ij} in (2.4) will be singular. To circumvent this we merely need to define

$$A_{ij}=\begin{pmatrix}\alpha_{ii} & \alpha_{ij}\\ \alpha_{ji} & \alpha_{jj}\end{pmatrix},\quad B_{ij}=\begin{pmatrix}\beta_{ii} & \beta_{ij}\\ \beta_{ji} & \beta_{jj}\end{pmatrix},\quad C_{ij}=\begin{pmatrix}\gamma_{ii} & \gamma_{ij}\\ \gamma_{ji} & \gamma_{jj}\end{pmatrix},$$
$$C_{ij}=A_{ij}\,\mathrm{adj}\,(B_{ij})=A_{ij}\begin{pmatrix}\beta_{jj} & -\beta_{ij}\\ -\beta_{ji} & \beta_{ii}\end{pmatrix}, \tag{3.8}$$

where adj stands for *adjugate* (or *adjoint*, depending on which book you read last). So $A\,\mathrm{adj}\,(A)=\det(A)I$, α_{ij} are the elements of A, and β_{ij} are the elements of B. We then replace (2.4) by

$$U^HC_{ij}V=S=\mathrm{diag}\,(\sigma_{ii},\sigma_{jj}),\qquad \sigma_{ii},\sigma_{jj}\geqq 0, \tag{3.9}$$

taking care that

$$U=V=I\quad\text{when } C_{ij}=0. \tag{3.10}$$

With these changes the general algorithm then proceeds exactly as the algorithm in § 2, and somewhat surprisingly is found in practice to converge to the desired result (3.5) and (3.6) when we have an even number of cycles, or the corresponding lower triangular forms when we have an odd number of cycles. An outline of the algorithm is given in § 6; it includes a possible reordering of the elements to ensure (3.2) to (3.6) result.

This approach treats A and B essentially equally, for if we define

$$K=\begin{pmatrix}0 & 1\\ -1 & 0\end{pmatrix} \tag{3.11}$$

then it is easy to check that

$$B_{ij}\,\mathrm{adj}\,(A_{ij})=K^TC_{ij}^TK=\mathrm{adj}\,(C_{ij}), \tag{3.12}$$

so this has the same elements, apart from sign, as (3.8), and both have the same singular values and eigenvalues. That is, working with (3.8) is essentially the same as working with (3.12). At first glance (3.5) and (3.6) seem to treat A and B differently, but if W is unitary so that RW is lower triangular, see (3.2), then if W is applied to the right of (3.5) and (3.6) the roles of A and B will essentially be reversed. But this is just the form resulting from the odd-numbered steps of the algorithm, and so we can say A and B are essentially treated equally in the algorithm.

We have suggested a reasonable method, but now to see if it gives us what we want, we first show that (3.8) to (3.10) give U^HA_{ij} and V^HB_{ij} parallel rows, and so give the GSVD of 2×2 A_{ij} and B_{ij}. We will then use this result to justify the algorithm.

4. The GSVD of 2×2 A and B. We will show that (3.8) and (3.9) result in U^HA_{ij} and V^HB_{ij} having parallel rows, so that (2.10) results in the GSVD of A_{ij} and B_{ij}, see (3.2) to (3.6). This makes (3.8) and (3.9) a reasonable choice for the 2×2 computations in our algorithm in § 3. That is, we compute the GSVD of two general matrices by an

iterative sequence of 2×2 GSVDs, in analogy with the Jacobi and Kogbetliantz methods. We will need the following result.

LEMMA 1. *If A and B are nonzero 2×2 matrices, then*

$$A \operatorname{adj}(B) = 0 \Leftrightarrow A = fd^T \text{ and } B = hd^T, \tag{4.1}$$

that is, $\operatorname{rank}(A) = \operatorname{rank}(B) = 1$ *and A and B have the same row space.*

Proof. If $B = hd^T = [\eta_1, \eta_2]^T[\delta_1, \delta_2]$ then

$$\operatorname{adj}(B) = [\delta_2, -\delta_1]^T[\eta_2, -\eta_1]. \tag{4.2}$$

Thus if $A = fd^T$ then $A \operatorname{adj}(B) = 0$. On the other hand if $A \operatorname{adj}(B) = 0$ with A and B nonzero 2×2 matrices, then $\operatorname{rank}(A) = \operatorname{rank}(B) = 1$. Write $B = hd^T$, $A = ft^T = f[\tau_1, \tau_2]$, so that

$$0 = A \operatorname{adj}(B) = f\cdot(\tau_1\delta_2 - \tau_2\delta_1)\cdot[\eta_2, -\eta_1],$$

and since f and h are nonzero, $\det([t, d]) = 0$ so that t and d are parallel. The result follows.

THEOREM 1. *If A and B are 2×2 matrices and U and V are unitary so that*

$$U^H A \operatorname{adj}(B) V = S = \operatorname{diag}(\sigma_1, \sigma_2), \qquad \sigma_1 \geqq \sigma_2 \geqq 0, \tag{4.3}$$

then the ith row of $U^H A$ is parallel to the ith row of $V^H B$, $i = 1, 2$. If also

$$A \operatorname{adj}(B) \neq 0 \tag{4.4}$$

and $U = [u_1, u_2]$, $V = [v_1, v_2]$, then

$$u_1^H A \neq 0, \qquad v_2^H B \neq 0, \tag{4.5}$$

$$\operatorname{rank}(A) = 1 \Rightarrow u_2^H A = 0, \tag{4.6}$$

$$\operatorname{rank}(B) = 1 \Rightarrow v_1^H B = 0. \tag{4.7}$$

Proof. From (4.3) we see that

$$U^H A \det(B) = SV^H B. \tag{4.8}$$

We enumerate all the possibilities:

(i) If A or B is zero the results are trivial.

(ii) If A is nonsingular the results follow from (4.8) and Lemma 1.

(iii) If $\operatorname{rank}(B) = 1$ we write $B = hd^T$ for nonzero h and d. Together with (4.3) and (4.8) this gives

$$SV^H B = 0, \quad V^H B = ed^T, \quad e \neq 0, \quad S = \begin{pmatrix} \sigma_1 & \\ & 0 \end{pmatrix}, \tag{4.9}$$

so (4.2) and (4.3) give

$$u_2^H A \begin{pmatrix} \delta_2 \\ -\delta_1 \end{pmatrix} = 0. \tag{4.10}$$

With this case of $\operatorname{rank}(B) = 1$ we have the following subcases.

(a) If A is nonsingular then $\sigma_1 > 0$ in (4.3), and so in (4.9)

$$V^H B = \begin{pmatrix} 0 \\ \varepsilon_2 d^T \end{pmatrix}, \quad \varepsilon_2 d^T \neq 0 \quad \text{for some scalar } \varepsilon_2, \tag{4.11}$$

proving (4.5) and (4.7). But from (4.10) we have nonsingular

$$U^H A = \begin{pmatrix} * \\ \phi_2 d^T \end{pmatrix} \tag{4.12}$$

for some scalar ϕ_2, and so the results follow.

(b) If $A = fd^T$ then $U^H A = U^H f d^T$ and $V^H B = V^H h d^T$ automatically have parallel rows, but from Lemma 1 (4.4) does not hold.

(c) Finally if

$$A = ft^T, \quad B = hd^T, \quad \det([t, d]) \neq 0, \quad f \neq 0, \quad h \neq 0, \tag{4.13}$$

then from Lemma 1 $A \operatorname{adj}(B) \neq 0$, so $\sigma_1 > 0$ in (4.3) and again (4.11) holds. But now (4.10) and (4.13) imply $u_2^H f = 0$, and so

$$U^H A = \begin{pmatrix} \phi_1 t^T \\ 0 \end{pmatrix}, \qquad V^H B = \begin{pmatrix} 0 \\ \varepsilon_2 d^T \end{pmatrix}, \tag{4.14}$$

which are necessarily nonzero with parallel rows. Also (4.4) holds, and (4.14) shows that (4.5) to (4.7) hold, completing the theorem.

We see that (4.3) corresponds to (3.9) in the algorithm of § 3, and so these results hold for our algorithm. The use of (3.10) in the algorithm leads to another important result.

THEOREM 2. *If* (3.8) *to* (3.10) *hold with unitary* $U = [u_1, u_2]$, $V = [v_1, v_2]$, $I = [e_1, e_2]$, *then*

$$e_2^T A_{ij} = 0 \Rightarrow u_2^H A_{ij} = 0, \tag{4.15}$$

$$e_1^T B_{ij} = 0 \Rightarrow v_1^H B_{ij} = 0, \tag{4.16}$$

$$e_2^T B_{ij} = 0 \Rightarrow \begin{cases} U = V = I & \text{if } C_{ij} \text{ is zero,} \\ \text{or } V = [e_2, e_1] & \text{if } C_{ij} \text{ is nonzero.} \end{cases} \tag{4.17}$$

Proof. If C_{ij} in (3.8) is nonzero (4.15) and (4.16) follow from (4.6) and (4.7). If C_{ij} is zero (4.15) to (4.17) follow from (3.10). If C_{ij} is nonzero then zero $e_2^T B_{ij}$ implies B_{ij} has rank unity and then (4.17) follows from (4.7) and the fact that V is unitary.

5. Properties of the algorithm. One approach to finding the GSVD is to transform A and B to separate the row subspaces, and then solve the GSVD of two square nonsingular triangular matrices using the method of § 2. This would be an easy method to follow and a theoretically correct algorithm, but it would require two rank decisions which in practice could be difficult to make. To avoid this we carry out the QR factorization of each of A and B as the only preprocessing, giving square $A^{(0)}$ and $B^{(0)}$ of the form in (3.7). Here we can be somewhat casual on the rank decisions and just use tolerances. We then apply the general iterative algorithm of § 3 to these, resulting in square $A^{(k)}$ and $B^{(k)}$ after the kth cycle. It is far from trivial to show that this gives the correct result in theory, and so we will summarize the main results first before proving the details. We use $S(A)$ to denote the row space of A.

We will show that after the first cycle we will have

$$A^{(1)} = \begin{pmatrix} A_{1.}^{(1)} \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} A_{11}^{(1)} & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \begin{matrix} \}r \\ \}s, \\ \end{matrix} \qquad B^{(1)} = \begin{pmatrix} B_{1.}^{(1)} \\ B_{2.}^{(1)} \\ 0 \end{pmatrix} = \begin{pmatrix} B_{11}^{(1)} & 0 & 0 \\ B_{21}^{(1)} & B_{22}^{(1)} & 0 \\ 0 & 0 & 0 \end{pmatrix} \begin{matrix} \}r \\ \}s \\ \end{matrix} \tag{5.1}$$

where $A_{11}^{(1)}$ is nonsingular and lower triangular, and $B_{11}^{(1)}$, $B_{22}^{(1)}$ are square and lower

triangular. It follows that

$$S(B_{1.}^{(1)}) \subset S(A^{(1)}). \tag{5.2}$$

A reordering step in the algorithm also ensures

$$B_{2.}^{(1)} \text{ has full row rank} \quad \text{and} \quad B_{22}^{(1)} \text{ is nonsingular.} \tag{5.3}$$

After every subsequent step we will show we have identically partitioned matrices

$$A^{(k)} = \begin{pmatrix} A_{1.}^{(k)} \\ A_{2.}^{(k)} \\ 0 \\ 0 \end{pmatrix} \begin{matrix} \}r_1 \\ \}r_2 \end{matrix}, \qquad B^{(k)} = \begin{pmatrix} 0 \\ B_{2.}^{(k)} \\ B_{3.}^{(k)} \\ 0 \end{pmatrix} \begin{matrix} \}r_1 \\ \}r_2 \\ \}s \end{matrix}, \qquad k>1, \tag{5.4}$$

where each nonzero block has full row rank and the dimensions are related to (5.1) via

$$r_1 + r_2 = r. \tag{5.5}$$

The even numbered cycles will result in identically partitioned matrices with the row partitioning of (5.4)

$$A^{(k)} = \begin{pmatrix} A_{11}^{(k)} & A_{12}^{(k)} & A_{13}^{(k)} & 0 \\ 0 & A_{22}^{(k)} & A_{23}^{(k)} & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}, \qquad B^{(k)} = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & B_{22}^{(k)} & B_{23}^{(k)} & 0 \\ 0 & 0 & B_{33}^{(k)} & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}, \tag{5.6}$$

$$k \text{ even}, \quad k>0,$$

$$A_{11}^{(k)}, A_{22}^{(k)}, B_{22}^{(k)}, B_{33}^{(k)} \text{ nonsingular upper triangular,} \tag{5.7}$$

while the odd numbered cycles will result in matrices with identical partitioning to (5.6)

$$A^{(k)} = \begin{pmatrix} A_{11}^{(k)} & 0 & 0 & 0 \\ A_{21}^{(k)} & A_{22}^{(k)} & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}, \qquad B^{(k)} = \begin{pmatrix} 0 & 0 & 0 & 0 \\ B_{21}^{(k)} & B_{22}^{(k)} & 0 & 0 \\ B_{31}^{(k)} & B_{32}^{(k)} & B_{33}^{(k)} & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}, \qquad k \text{ odd}, k>1, \tag{5.8}$$

$$A_{11}^{(k)}, A_{22}^{(k)}, B_{22}^{(k)}, B_{33}^{(k)} \text{ nonsingular low triangular.} \tag{5.9}$$

In computing (5.6) from (5.8) or the identically partitioned $B^{(1)}$, $B_{3.}^{(k)}$ is not touched by the left transformations V^H, while in computing (5.8) from (5.6) $A_{1.}^{(k)}$ is not touched by the left transformation U^H.

When we have proven (5.1) to (5.9) we can prove the following key result. From (5.6) and (5.7)

$$S(A_{2.}^{(k-1)}) \subset S(B^{(k-1)}), \quad k-1 \text{ even}, \quad k>1, \tag{5.10}$$

so the comment following (5.9) ensures in (5.8)

$$S(A_{2.}^{(k)}) \subset S(B^{(k)}), \quad k \text{ odd}, \quad k>1. \tag{5.11}$$

But $B_{33}^{(k)}$ is nonsingular in (5.8), and so

$$A_{2.}^{(k)} = C_2^{(k)} B_{2.}^{(k)}, \quad k \text{ odd}, \quad k>1.$$

In a similar way we obtain a result for even k, and using (5.7) and (5.9) we have

$$A_{2\cdot}^{(k)} = C_2^{(k)} B_{2\cdot}^{(k)}, \quad C_2^{(k)} \text{ nonsingular}, \quad k = 2, 3, 4, \cdots \tag{5.12}$$

where we use (5.2) for the case $k = 2$.

We see that (5.12) gives the common row space of maximum dimension since $A_{11}^{(k)}$ is nonsingular in (5.6), and $B_{33}^{(k)}$ is nonsingular in (5.8). In theory this means that after the first two cycles of the algorithm only the rows of $A_{2\cdot}^{(k)}$ and $B_{2\cdot}^{(k)}$ are affected by transformations from the left, and so we have carried out a direct algorithm for separating subspaces. In practice we can still allow $B_{3\cdot}^{(k)}$ and $B_{2\cdot}^{(k)}$ to be combined in the transformation of (5.6), and $A_{1\cdot}^{(k)}$ and $A_{2\cdot}^{(k)}$ to be combined in the transformation of (5.8), giving possibly superior final results in the presence of rounding errors to those obtained by working with $A_{2\cdot}^{(k)}$ and $B_{2\cdot}^{(k)}$ alone in the iterative part of the algorithm.

In the third and subsequent cycles, an examination of the individual rotations shows that we are implicitly carrying out Kogbetliantz' algorithm to diagonalize the implicitly defined nonsingular trangular matrix $C_2^{(k)}$ in (5.12), and this necessarily converges if the angles of rotation obey (2.2). This means if (5.1) to (5.9) hold then we have the required proof for this algorithm in the absence of rounding errors.

We summarize the algorithm prior to showing the results of this section hold.

(5.13) ALGORITHM

begin {GSVD algorithm, A, B, n given as in (3.7)}
  $U := I_n$; $V := I_n$; $Q := I_n$; {if wanted}
  cycle := 0;
  while nonconvergence **and** (cycle < 10) **do**
  begin {general step}
    cycle := *cycle* + 1;
    for $i := 1$ **to** $n-1$ **do**
      for $j := i+1$ **to** n **do**
      begin {(i,j) transformation. Here α_{ij} and β_{ij} are the (i,j) elements of A and B respectively. U_{ij}, V_{ij}, Q_{ij} are 2×2 unitary matrices.}

$$A_{ij} := \begin{pmatrix} \alpha_{ii} & \alpha_{ij} \\ \alpha_{ji} & \alpha_{jj} \end{pmatrix} \qquad B_{ij} := \begin{pmatrix} \beta_{ii} & \beta_{ij} \\ \beta_{ji} & \beta_{jj} \end{pmatrix};$$

      $U_{ij}^H A_{ij} \operatorname{adj}(B_{ij}) V_{ij} = \operatorname{diag}(\sigma_1, \sigma_2)$, $\sigma_1, \sigma_2 \geqq 0$, {where $\sigma_1 \geqq \sigma_2$ if *cycle* $\leqq 2$, else (2.2) should hold}, with $U_{ij} = V_{ij} = I$ if $A_{ij} \operatorname{adj}(B_{ij}) = \operatorname{diag}(\sigma_1, \sigma_1)$;
      Choose Q_{ij} so $U_{ij}^H A_{ij} Q_{ij}$ and $V_{ij}^H B_{ij} Q_{ij}$ are lower triangular if *cycle* is odd, else upper triangular;
      Let U_n, V_n, Q_n be $n \times n$ unit matrices with (i,i), (i,j), (j,i), (j,j) elements replaced by the (1, 1), (1, 2), (2, 1), (2, 2) elements respectively of U_{ij}, V_{ij}, Q_{ij} respectively;
      $A := U_n^H A Q_n$; $B := V_n^H B Q_n$;
      $U := U U_n$; $V := V V_n$; $Q := Q Q_n$; {as required}
      end; {(i,j) transformation}
    if *cycle* = 1 **then do**
    begin {reorder}
      Let r and s be as in (5.1);
      for $i := r+1$ **to** $r+s$ **do**
        if the ith row of B is zero, move it and the ith column to the end of B, bringing all remaining rows and columns up one;
    end; {reorder}
  end; {general step}
end. {GSVD algorithm}

The remainder of this section is hard going. There is undoubtedly a more straightforward way of proving that this algorithm works, and I hope someone else can find it.

Theorem 1 proves that both $U_{ij}^H A_{ij}$ and $V_{ij}^H B_{ij}$ in the algorithm are triangularized by the one unitary matrix Q_{ij}, and as a result the progress of each odd numbered cycle is correctly described by A and B in (2.9), with the corresponding result for even numbered cycles. This means that $A^{(k)}$ and $B^{(k)}$ are lower triangular for odd k and upper triangular for even k, so (5.1) and (5.6) to (5.9) have the correct triangular form. But Theorem 2 shows that all the zero rows at the bottom of $A^{(k-1)}$ stay there, so from (3.7) the leading r rows of $A^{(k)}$ in (5.1) and (5.4) have full row rank, and (5.2) holds. This gives nonsingularity of $A_{11}^{(1)}$ in (5.1) and the A matrices in (5.9). Theorem 2 also shows that in an (i, j) transformation a zero row of $B^{(k)}$ can be exchanged with a nonzero row above it, but not with one below it, and a nonzero row cannot be combined with a zero row to give two nonzero rows. Combined with (3.7) this gives:

RESULT 1. *The collection of nonzero rows of any $B^{(k)}$ has full row rank, and apart from the reordering step in the algorithm nonzero rows of B can only move downward.*

The insistence that A_{11} and B_{22}, if it exists, be nonsingular upper triangular in (3.7) ensures in (5.1) that the diagonal elements beyond the rth corresponding to nonzero rows of $B^{(1)}$ are themselves nonzero. We prove this as a slightly more general result which also applies to the cycle which takes (5.6) to (5.8). We see (5.6) is a special case of (3.7a), and (5.8) is a special case of (5.1).

RESULT 2. *Let A_{11} and B_{22} be nonsingular upper triangular in* (3.7a), *and let* (5.1) *be the result of applying one cycle of the algorithm* (5.13) *to these, then $B^{(1)}$ has nonzero diagonal elements in the nonzero rows beyond the rth.*

Proof. Two classes of (i, j) transformations are relevant. When $i<j\leqq r$, A_{ij} is nonsingular and so the (i, i) and (j, j) elements of A remain nonzero. When $i\leqq r<j$ there is no U_{ij} transformation of A_{ij}, and the Q_{ij} transformation gives the (i, i) element of A size $\|[\alpha_{ii}, \alpha_{ij}]\|_2>0$, since α_{ii} is nonzero. We will show that if the jth row of B is zero and it is exchanged with the ith row this results in a nonzero (j, j) element of B, while if the (j, j) element of B is nonzero it stays so. Since initially B_{22} is nonsingular upper triangular in (3.7a) this will prove Result 2 by induction. When j corresponds to a zero row of B it will only be exchanged with the ith row if the top rows of A_{ij} and B_{ij} are not parallel; consequently the Q_{ij} which makes the $(1, 2)$ element of $A_{ij}Q_{ij}$ zero cannot also make the new (j, j) element of B zero. Next suppose before the (i, j) transformation that the (j, j) elements of B is nonzero. This implies that A_{ij} and $V_{ij}^H B_{ij}$ have parallel first rows, and since the $(1, 1)$ element of A_{ij} is nonzero the $(2, 2)$ element of $V_{ij}^H B_{ij}$ must be nonzero, since it could only be zero if V_{ij} were an exchange putting $[0, *]$ in the first row. Since $A_{ij}Q_{ij}$ has zero $(1, 2)$ element Q_{ij} does not exchange the columns, and $V_{ij}^H B_{ij}Q_{ij}$ cannot have zero $(2, 2)$ element, completing the proof of Result 2.

Before we can use Result 2 to say $B_{22}^{(1)}$ in (5.1) and $B_{33}^{(k)}$ in (5.8) are nonsingular we have to show that after the reordering step they are still lower triangular and $B_{2\cdot}^{(1)}$ in (5.1) and $B_{3\cdot}^{(k)}$ in (5.4) have no zero rows.

Let B' be the B matrix prior to the reordering step that results in $B^{(1)}$. Occasionally the nonzero rows after the rth in B' will not be adjacent, considerably complicating our analysis. Here we show that if $j>r$ and the jth row of B' is zero then so is the jth column, so the reordering step moves both these to the end. The jth column of $A^{(1)}$ is already zero, and the exchange has no effect on it. The remaining rows and columns of B' each move up one so the triangular form of $B_{22}^{(1)}$ is maintained and finally (5.3) follows from Result 2. Suppose the jth row of B' is zero, $j>r$, but B' has at least one nonzero row with higher index. In the (i, j) transformations in any cycle, zero rows of B can only be moved upwards, and since (j, t) transformations, $t>j$,

have $A_{jt}=0$, a zero row cannot be moved into the jth row from below. It follows that the jth row and all below it were zero at the start, and the jth row has remained zero through all the transformations. Now consider an (i,j) transformation in the first cycle, with $i<j$. We must have

$$A_{ij}=\begin{pmatrix}\times & \times\\ 0 & 0\end{pmatrix},\qquad B_{ij}=\begin{pmatrix}\times & \times\\ 0 & 0\end{pmatrix}. \tag{5.14}$$

The top rows of these are parallel, otherwise the nonzero ith row of B would be exchanged with the jth, which we have just shown did not happen. It follows from Lemma 1 that $U_{ij}=V_{ij}=I$, and then $A_{ij}Q_{ij}$ and $B_{ij}Q_{ij}$ have zero $(1,2)$ elements, so the (i,j) element of B is made zero. This ith row can now only be affected in (i,t) transformations with $t>j$. If the tth row of B were zero it could be made nonzero by exchanging it with this ith row, but the (i,j) and (t,j) elements would remain zero. If the tth row of B were nonzero it must have initially been the result of an earlier exchange like this giving it zero jth element. Clearly if the ith and tth rows of B are combined in the (i,t) transformation, the (i,j) and (t,j) elements of B are zero before and after this transformation. It follows that B' has zero jth column as stated. The reordering step then gives $B_{2\cdot}^{(1)}$ full row rank in (5.1) with $B_{22}^{(1)}$ lower triangular and nonsingular proving (5.3).

We have seen that all the rows of $B^{(k)}$ beyond the $(r+s)$th must henceforth remain zero, and since no zero rows can be moved downward or combined with nonzero rows, and the nonzero rows maintain full row rank, $B_{3\cdot}^{(k)}$ in (5.4) must have full row rank for $k>1$. Also if $j>r+s$, A_{ij} and B_{ij} from (5.1), (5.6) and (5.8) have zero second columns and rows, so these will not be transformed and the columns beyond the $(r+s)$th remain zero after all cycles. This with the full row rank of $B_{3\cdot}^{(k)}$ shows that $B_{33}^{(k)}$ in (5.6) is nonsingular.

In theory this reordering step can only have an effect immediately following the first cycle; however in practice with the use of tolerances it could also be useful in later cycles. We now want to show $B_{2\cdot}^{(k)}$ in (5.4) has full row rank.

An (i,j) transformation on (5.1) or (5.8) with $1\leqq i<j\leqq r$ transforms a nonsingular lower triangular A_{ij} into necessarily nonsingular upper triangular form, while from Theorem 2 if B_{ij} has nonzero first row and zero second these are exchanged. Thus if there are exactly r_1 zero rows in $B_{1\cdot}^{(1)}$ in (5.1), these are all put at the top of $B^{(2)}$ in (5.6) by the (i,j) transformations with $i=1,\cdots,r_1$, leading to full row rank $B_{2\cdot}^{(2)}$ in (5.4). But with Result 1 this shows $B_{2\cdot}^{(k)}$ has full row rank for all $k>1$, so $B_{22}^{(k)}$ in (5.7) is nonsingular.

Now we show that $A_{11}^{(k)}$ and $A_{22}^{(k)}$ in (5.6), and $B_{22}^{(k)}$ in (5.8), are nonsingular. An (i,j) transformation on (5.8) with $i<j\leqq r$ leaves the (i,i) and (j,j) elements of A nonzero, while with $i\leqq r<j$ the $(1,1)$ element of A_{ij} is nonzero and the $(2,2)$ element of B_{ij} is nonzero, so with an analogous argument to that in the proof of Result 2, the (i,i) element of A stays nonzero. This completes the proof of (5.7). A similar argument completes the proof of (5.9).

It remains for us to support the sentence following (5.9). We see that in computing (5.6) from (5.8) $B_{3\cdot}^{(k)}$ could only be affected by the left transformations V^H in (i,j) transformations with $r<j\leqq r+s$. But in these A_{ij} has zero last row so corresponding rows of the lower triangular A_{ij} and B_{ij} are parallel, and from (4.1) $A_{ij}\,\mathrm{adj}\,(B_{ij})=0$, and so $U_{ij}=V_{ij}=I$ in the algorithm. The same sort of argument holds for $A_{1\cdot}^{(k)}$ in going from (5.6) to (5.8). This completes all the results we required, and so the algorithm is theoretically correct.

6. Numerical examples. The examples here were generated using the MATLAB system of Moler [12] on a VAX 11/750. They were not chosen to be particularly testing of the numerical stability of the algorithm, but rather to show how it handled different ranks and common row spaces.

In each of the examples we took $A(n \times n)$ $B(n \times n)$, with possibly some zero rows at the bottom, and carried out the QR decomposition to give upper triangular A and B. We took $n = 6$ and built up our matrices as the products of random matrices of the ranks we chose to test. Since the algorithm is designed to produce parallel rows, we used this as a measure of convergence. For two vectors a and b we defined

$$\text{par}(a, b) = \begin{cases} 0 & \text{if } \|a\| < \text{tol or } \|b\| < \text{tol}, \\ \min\left\{\text{norm}\left(\dfrac{a}{\|a\|} \pm \dfrac{b}{\|b\|}\right)\right\}, & \text{otherwise;} \end{cases} \tag{6.1}$$

and if a_i^T and b_i^T were the ith rows of our matrices at the end of a cycle, we took as an indicator of convergence

$$\text{error} = \sum_{i=1}^{n} \text{par}(a_i, b_i). \tag{6.2}$$

Here $\|a\| = (a^H a)^{1/2}$ and we took $\text{tol} = 10^{-14}$ for these test problems. We stopped when

$$\text{error} \leqq \text{tol} \tag{6.3}$$

and then computed

$$\sigma_i = \begin{cases} 0 & \text{if } \|a_i\| = 0 \text{ or } \|b_i\| = 0, \\ \|a_i\| / \|b_i\| & \text{otherwise,} \end{cases} \tag{6.4}$$

to give the equivalent of $\sigma_i = \alpha_i / \beta_i$ in (3.3). We write $s = [\sigma_1, \cdots, \sigma_n]$. When B is nonsingular we give t as the vector of computed singular values of AB^{-1} and cond (A) as the ratio of the largest to smallest singular value of A. One cycle takes A and B from upper triangular to lower triangular form, or vice versa; see (2.9).

Example 1. *Nonsingular A and B*, cond $(A) = 24$, cond $(B) = 21$.

cycle	1	2	3	4	5
error	2.5	.52	.0018	3×10**(−10)	2×10**(−16)

$s =$	4.356373396988140	3.492138554949318	1.858034577501202	1.001892079494734
	.300018648851983	.197330532251905		
$t =$	4.356373396988142	3.492138554949318	1.858034577501202	1.001892079494734
	.300018648851983	.197330532251905		

The algorithm gives as much accuracy as possible, but it does take a while to start converging.

Example 2. rank $(A) = 4$, rank $(B) = 3$, *common row space dimension* 2.

cycle	1	2	3	4	5
error	.47	2×10**(−17)			

After the QR decomposition the original matrices were:

COLUMNS 1 THRU 4

$A =$	−1.483754207808792	−2.754569846168734	−1.901063717347766	−3.134725173442366
	.000000000000000	−.660367822613970	−.558467086690151	−.472306743711252
	.000000000000000	.000000000000000	−.190188592370571	−.444428846555170
	.000000000000000	.000000000000000	.000000000000000	−.047939573588338
	.000000000000000	.000000000000000	.000000000000000	.000000000000000
	.000000000000000	.000000000000000	.000000000000000	.000000000000000

COLUMNS 5 THRU 6

$A =$	−1.814262790205398	−2.047678638962671
	−.263343099599686	−.264594139637406
	−.220470422637046	−.284218707808010
	−.149658467473861	−.071714348021438
	.000000000000000	.000000000000000
	.000000000000000	.000000000000000

COLUMNS 1 THRU 4

$B =$	−2.152554420079003	−3.894294758406499	−2.368033003532614	−4.429539352490643
	.000000000000000	.074280121258419	.133400322312574	.209165610430773
	.000000000000000	.000000000000000	.073602580523686	−.176428622793232
	.000000000000000	.000000000000000	.000000000000000	.000000000000000
	.000000000000000	.000000000000000	.000000000000000	.000000000000000
	.000000000000000	.000000000000000	.000000000000000	.000000000000000

COLUMNS 5 THRU 6

$B =$	−3.459002749639446	−3.411337868703009
	.079934376871628	.121676462551819
	−.387079235508506	−.368994080158650
	−.000000000000001	−.000000000000001
	.000000000000000	.000000000000000
	.000000000000000	.000000000000000

Note the small nonzero elements in positions (4, 5) and (4, 6) of B. These have been introduced by rounding errors.

After 2 cycles of the GSVD algorithm the matrices were:

COLUMNS 1 THRU 4

$A =$	−.200073697315396	−.216325788985202	−.138100394035475	−.593297250589336
	.000000000000000	−.222361539578086	.118655281512143	.580017836681617
	.000000000000000	.000000000000000	.389711441881607	.218509802415124
	.000000000000000	.000000000000000	.000000000000000	.199123441354269
	.000000000000000	.000000000000000	.000000000000000	.000000000000000
	.000000000000000	.000000000000000	.000000000000000	.000000000000000

COLUMNS 5 THRU 6

$A =$	2.269980552374192	.000000000000000
	−4.039759246493302	.000000000000000
	−2.527788530781247	.000000000000000
	−1.805072397942648	.000000000000000
	.000000000000000	.000000000000000
	.000000000000000	.000000000000000

COLUMNS 1 THRU 4

$B =$	.000000000000000	.000000000000000	.000000000000000	.000000000000000
	.000000000000000	.000000000000000	.000000000000000	.000000000000000
	.000000000000000	.000000000000000	.128833790821756	.072236642683449
	.000000000000000	.000000000000000	.000000000000000	.489752152327848
	.000000000000000	.000000000000000	.000000000000000	.000000000000000
	.000000000000000	.000000000000000	.000000000000000	.000000000000000

```
COLUMNS 5 THRU 6
B = -.000000000000001     .000000000000000
     .000000000000001     .000000000000000
    -.835655677041274     .000000000000000
   -4.439648521477558     .000000000000000
    6.954830268687245     .000000000000000
     .000000000000000     .000000000000000

s = .000000000000000     .000000000000000    3.024916362360086    .406580022992879
     .000000000000000     .000000000000000
```

Only two cycles were required, since the first separates out the common row space of dimension 2, and then the second produces parallel rows in this row space. This is because a 2×2 GSVD is solved exactly in one step. Note the small nonzeros in positions (1, 5) and (2, 5) of B. These were ignored by the use of the tolerance.

Example 3. rank (A) = rank $(B) = 4$, *common row space dimension* = 3.

cycle	1	2	3	4	5
error	.33	.003	2×10**(−9)	9×10**(−17)	

After the QR decomposition, the original matrices were:

```
COLUMNS 1 THRU 4
A = -1.483754207808792   -2.754569846168734   -1.901063717347766   -3.134725173442366
     .000000000000000    -.660367822613970    -.558467086690151    -.472306743711252
     .000000000000000     .000000000000000    -.190188592370571    -.444428846555170
     .000000000000000     .000000000000000     .000000000000000    -.047939573588338
     .000000000000000     .000000000000000     .000000000000000     .000000000000000
     .000000000000000     .000000000000000     .000000000000000     .000000000000000

COLUMNS 5 THRU 6
A = -1.814262790205398   -2.047678638962671
    -.263343099599686    -.264594139637406
    -.220470422637046    -.284218707808010
    -.149658467473861    -.071714348021438
     .000000000000000     .000000000000000
     .000000000000000     .000000000000000

COLUMNS 1 THRU 4
B = -3.123355429329773   -5.020366343436982   -2.972191466297970   -5.812154899974686
     .000000000000000     .207884167869281     .158033933963608     .182365676077579
     .000000000000000     .000000000000000     .107328952460706    -.021768478702607
     .000000000000000     .000000000000000     .000000000000000     .219692246210581
     .000000000000000     .000000000000000     .000000000000000     .000000000000000
     .000000000000000     .000000000000000     .000000000000000     .000000000000000

COLUMNS 5 THRU 6
B = -4.217468529709810   -4.339624584281506
     .227500717178881     .164471791036765
    -.275289576213577    -.216390069125860
     .269744345263956     .300087815368396
     .000000000000000     .000000000000000
     .000000000000000     .000000000000000
```

After 4 steps of the GSVD algorithm the matrices were:

COLUMNS 1 THRU 4

$A =$	−.152535367220157	.215310202586866	−.277325505745866	−.659393620451283
	.000000000000000	.427810977395549	−.029228351378672	−.206360377382993
	.000000000000000	.000000000000000	−.212975875737842	.175581097281234
	.000000000000000	.000000000000000	.000000000000000	.194134471721972
	.000000000000000	.000000000000000	.000000000000000	.000000000000000
	.000000000000000	.000000000000000	.000000000000000	.000000000000000

COLUMNS 5 THRU 6

$A =$	−3.535137973645832	.000000000000000
	−3.172167996024390	.000000000000000
	1.906135608940945	.000000000000000
	2.258917563058958	.000000000000000
	.000000000000000	.000000000000000
	.000000000000000	.000000000000000

COLUMNS 1 THRU 4

$B =$	.000000000000000	.000000000000000	.000000000000000	.000000000000000
	.000000000000000	.121957526020080	−.008332225239963	−.058827852542294
	.000000000000000	.000000000000000	.144065810553351	−.118770414771316
	.000000000000000	.000000000000000	.000000000000000	.491824577188100
	.000000000000000	.000000000000000	.000000000000000	.000000000000000
	.000000000000000	.000000000000000	.000000000000000	.000000000000000

COLUMNS 5 THRU 6

$B =$	.000000000000000	.000000000000000
	−.904300687351262	.000000000000000
	−1.289390033379685	.000000000000000
	5.722791864318372	.000000000000000
	8.899102416565753	.000000000000000
	.000000000000000	.000000000000000

$s =$	.000000000000000	3.507868610954851	1.478323517008020	.394722998252534
	.000000000000000	.000000000000000		

Again the algorithm behaved as well as could be hoped.

These examples show that the theoretical behaviour described in § 5 also occurs in practice. This is not too surprising since each of A and B is subject to unitary transformations only. There is one caveat here, and that is that the nonzero singular values of A and B are all in a reasonable range, so none of the final nonzero rows is very small. If they were, we would not be able to use (6.3) as our test for convergence, because rounding errors would stop (6.2) being that small. It is clear that improvements will be needed in a production code.

7. Comments. We have suggested a straightforward algorithm for computing the GSVD of given matrices $A(m \times n)$ and $B(p \times n)$, or the CSD of a partitioned unitary matrix. The approach is a unified one whereby the A and B matrices are transformed separately with unitary transformations (orthogonal transformations when A and B are real) and no difficult rank decisions need be made during the computation. Because of its simplicity the algorithm is fairly easy to program, however the proof of correctness here was long and involved, and it is hoped that someone else can derive a more brief and straightforward proof.

For most problems the criterion (2.2) need not be enforced, however it would appear to be theoretically necessary as the following example suggests. Let

$$C = \begin{pmatrix} \alpha & 0 & \beta \\ 0 & \alpha & 0 \\ 0 & 0 & \alpha \end{pmatrix}, \tag{7.1}$$

and apply Kogbetliantz' algorithm to this, exchanging rows i and j, and columns i and j, each (i,j) transformation. After 6 such transformations the matrix will be identical to its starting value (7.1), so there is no convergence.

The numerical behaviour of the algorithm has so far been exemplary. By following the work of Wilkinson [20] we know that with correct use of unitary rotations our computed matrices $A^{(k)}$ and $B^{(k)}$ will be the exact result of applying unitary matrices $U^{(k)}$, $V^{(k)}$ and $Q^{(k)}$ to slightly perturbed initial data $A+\delta A^{(k)}$ and $B+\delta B^{(k)}$, so that if we obtain convergence we can be confident about our results. We have not proven convergence in the presence of rounding errors, but all the tests run so far have converged pleasingly swiftly, rarely taking more than 6 cycles to give full precision using MATLAB on a VAX 11/750.

The ultimate convergence appears to be at least quadratic. The method implicitly applies Kogbetliantz' method, for which quadratic convergence has recently been proven when there are no pathologically close singular values [21], and work on the general case is in progress. One delightful observation that may contribute to the understanding of convergence is the following.

RESULT 3. Kogbetliantz' serial-cyclic algorithm applied to nonsingular triangular C also implicitly applies the algorithm to C^{-1}.

This means the algorithm is diagonalizing C and C^{-1} in exactly the same way at the same time, and this could help convergence. The result can be proven by examining the form of C and C^{-1} during the elimination of the offdiagonal elements of the first row of C as in (2.3), and using induction. Note however that the rate of convergence is similar for triangular and full C.

For our algorithm the above result means that if A and B are nonsingular the algorithm is implicitly diagonalizing AB^{-1} and BA^{-1} simultaneously, so that A and B are treated completely equally. However the iterative algorithm also works with singular square A and B, and in this case we have the following.

RESULT 4. The algorithm (5.13) implicitly applies Kogbetliantz' algorithm to both A adj (B) and B adj (A).

This is not a very significant result if for example the rank of B is less than $n-1$, for then adj $(B)=0$. However it does indicate that this is an equal opportunity algorithm.

8. Systolic array implementation. A brief sketch will be given here describing how this GSVD algorithm can be implemented for systolic array computations. This was first worked out by W. M. Gentleman [22], but since he did not have time to write it up, this outline is included to round out the presentation. The term "rotation" will be used loosely to mean either a 2×2 unitary rotation or a 2×2 elementary unitary hermitian.

The important step is to implement Kogbetliantz' basic algorithm in a way that can be transferred effectively to systolic array processors. The physical ordering presented in (2.1) and (2.3) cannot be used because it combines rows and columns i and j where the pair (i,j) cycles through $(1,2)$, $(1,3),\cdots,(1,n)$; $(2,3),\cdots,(2,n);\cdots;(n-1,n)$, which does not allow a lot of concurrency. The ideal ordering will only apply rotations to adjacent rows and to adjacent columns. The ordering suggested by Gentleman starts with an upper triangular matrix as in (2.3), and gives theoretically identical results to the one in (2.3), but one cycle leaves the matrix in upper triangular form rather than lower triangular form as in (2.3). In fact if

$$\text{(8.1)}\qquad \begin{pmatrix} -c_1 & s_1 \\ s_1 & c_1 \end{pmatrix} \quad \text{and} \quad \begin{pmatrix} -c_2 & s_2 \\ s_2 & c_2 \end{pmatrix}$$

are the nontrivial elements of the left and right rotations respectively for one 2×2 SVD in (2.3), then in Gentleman's ordering the left and right rotations have the diagonal blocks

$$\begin{pmatrix} s_1 & c_1 \\ -c_1 & s_1 \end{pmatrix} \quad \text{and} \quad \begin{pmatrix} s_2 & -c_2 \\ c_2 & s_2 \end{pmatrix} \tag{8.2}$$

respectively.

Thus the ordering corresponds to the same rotations on the same elements as in (2.3), with each rotation followed by a permutation. We illustrate this with the equivalent diagram to (2.3), but include indices which indicate the position of each corresponding element in (2.3). The element to be eliminated is underlined immediately prior to elimination.

(8.3)

$$\begin{array}{ccccccccccccccccccc}
11 & & \underline{12} & & 13 & & 14 & \to & 22 & & & 23 & & 24 & \to & 22 & & 23 & & 24 & & 21 \\
 & & 22 & & 23 & & 24 & & & & 11 & & \underline{13} & & 14 & & & 33 & & 34 & & 31 \\
 & & & & 33 & & 34 & & & & & & 33 & & 34 & & & & 11 & & \underline{14} & & \\
 & & & & & & 44 & & & & & & & & 44 & & & & & & 44 & & \\
 & & & & & & & & & & & & & & & & & & & & & & 11 \\
 & & & & & & & \to & 33 & & & 34 & & 31 & \to & 33 & & \underline{34} & & 32 & & 31 & \to & 44 \\
 & & & & & & & & & & 22 & & \underline{24} & & 21 & & & 44 & & & & 41 & & \\
 & & & & & & & & & & & & 44 & & & & & & & 22 & & 21 & & \\
 & & & & & & & & & & & & & & 11 & & & & & & & 11 & &
\end{array}$$

Thus the rotations are in the (i, j) plane where (i, j) goes through $(1, 2), (2, 3), (3, 4), \cdots, (n-1, n)$; $(1, 2), (2, 3), \cdots, (n-2, n-1); \cdots; (1, 2), (2, 3); (1, 2)$. Only adjacent rows or columns are combined, so this is suitable for implementation using systolic array processors. However, if sequential computations are used, this approach can be made to give numerically identical results to that in (2.3). This parallels the ordering that Stewart [23] uses for the unsymmetric eigenvalue problem.

If we refer to (8.3) as the forward cycle, then the next cycle is referred to as the reverse cycle, and has rotations in the planes $(n-1, n)$; $(n-2, n-1), \cdots, (1, 2)$; $(n-1, n), (n-2, n-1), \cdots, (2, 3); \cdots; (n-1, n), (n-2, n-1); (n-1, n)$. Choosing the angles correctly will mean that this gives exactly the same upper triangular matrix as the two cycles described by (2.3) and its following paragraph. Sweep would perhaps be a more satisfactory term than cycle here.

The way this method can be applied to obtaining the GSVD of A and B, or the SVD of AB as in [8], is now easy to see. If A, B and C are upper triangular, and A_{ij}, B_{ij} and C_{ij} are 2×2 matrices with the (i, i), (i, j), (j, i) and (j, j) elements of A, B and C respectively, and $j = i+1$, then

$$A_{ij} = C_{ij}B_{ij} \quad \text{if } A = CB \quad \text{and} \tag{8.4}$$

$$C_{ij} = A_{ij}B_{ij} \quad \text{if } C = AB. \tag{8.5}$$

Thus if we keep A and B upper triangular throughout the computation, the three implicitly defined elements of C required for each 2×2 SVD in (8.3) will be available just as they were in (2.7) for example. If we now concentrate on the GSVD of A and B, the computation corresponding to (2.7) will have the rotation k in (2.7) followed by a permutation, and the rotation k' followed by a permutation, but k'' will now be chosen to eliminate the $(2, 1)$ elements of A_{ij} and B_{ij}, rather than the $(1, 2)$ element as in (2.7). Since with $j = i+1$ this gives the correct transformation for C in (8.3), and keeps both A and B upper triangular, the computations for C in (8.3) can be carried

through implicitly by only applying rotations to adjacent rows and to adjacent columns of both A and B. And it can be seen from (2.9) that these occur in exactly the same order as for C in (8.3). What is more with sequential computations this can be made numerically identical to the earlier GSVD, since the permutations after k and k' in (2.7) ensure the new k'' is acting on the same elements as k'' in (2.7).

We will leave the details of the systolic array implementation to others. It suffices here to point out in (8.3) for example, that while the element marked $\underline{12}$ is being eliminated in this forward cycle, the element in the position marked $\underline{34}$ in that first matrix can be eliminated as part of the previous reverse cycle. Similarly $\underline{14}$ and $\underline{23}$ in the 3rd and 4th matrices can be eliminated simultaneously, as can $\underline{34}$ in the 6th and $\underline{21}$ (corresponding to the reverse cycle) in the 7th.

In conclusion this very nice ordering initially proposed by Stewart for the eigenproblem [23], and now by Gentleman [22] for the SVD and GSVD, is a very natural and easy one to follow and implement, and appears to be the correct approach to use in implementing such algorithms using systolic array processors.

Acknowledgments. I am deeply indebted to Binay Bhattacharya and Mike Saunders for the many insights that arose from the hours of discussion and computing that went into our earlier attempts at this problem. I am also grateful to Mike Heath for his steadfast insistence on treating both matrices with complete equality in problems like this. It was after working with Mike Heath, Alan Laub and Bob Ward on [8] that I realized the correct approach to the algorithm here. Gene Golub contributed very positively during the difficult proofs of § 5. Morven Gentleman rounded out the work very nicely by showing how to implement the algorithm using systolic array processors.

REFERENCES

[1] B. K. BHATTACHARYA, C. C. PAIGE AND M. A. SAUNDERS, unpublished code for the generalized singular value decomposition, 1980.

[2] R. P. BRENT, F. T. LUK AND C. VAN LOAN, *Computation of the singular value decomposition using mesh-connected processors*, Technical Report CS-52, Dept Computer Science, Cornell Univ., Ithaca, NY, 1983.

[3] C. DAVIS AND W. M. KAHAN, *The rotation of eigenvectors by a perturbation.* III, SIAM J. Numer. Anal., 7 (1970), pp. 1-46.

[4] G. E. FORSYTHE AND P. HENRICI, *The cyclic Jacobi method for computing the principal values of a complex matrix*, Trans. Amer. Math. Soc., 94 (1960), pp. 1-23.

[5] G. GOLUB AND W. KAHAN, *Calculating the singular values and pseudo-inverse of a matrix*, SIAM J. Numer. Anal., 2 (1965), pp. 205-224.

[6] G. H. GOLUB AND C. REINSCH, *Singular value decomposition and least squares solutions*, Numer. Math., 14 (1970), pp. 403-420.

[7] G. H. GOLUB AND C. F. VAN LOAN, *Matrix Computations*, The Johns Hopkins Univ. Press, Baltimore, MD, 1983.

[8] M. T. HEATH, A. J. LAUB, C. C. PAIGE AND R. C. WARD, *Computing the* SVD *of product of two matrices*, this Journal, 7 (1986), pp. 1147-1159.

[9] C. G. J. JACOBI, *Über eine neue Auflösungsart der bei der Methode der kleinsten Quadrate vorkommenden linearen Gleichungen, (On a new way of solving linear equations occurring in the method of least squares)*, Astronom. Nachr., 22 (1845), pp. 297-306. (Werke, Vol III, 1884, pp. 468-478.)

[10] E. G. KOGBETLIANTZ, *Diagonalization of general complex matrices as a new method for solution of linear equtions*, Proc. Internat. Congr. Math., Amsterdam, 2 (1954), pp. 356-357.

[11] ———, *Solution of linear equations by diagonalization of coefficients matrix*, Quart. Appl. Math., 13 (1955), pp. 123-132.

[12] C. B. MOLER, MATLAB *Users' Guide*, Technical Report CS81-1, Dept. of Computer Science, Univ. New Mexico, Albuquerque, NM, 1981.

[13] C. B. MOLER AND G. W. STEWART, *An algorithm for generalized matrix eigenvalue problems*, SIAM J. Numer. Anal., 10 (1973), pp. 241-256.

[14] C. C. PAIGE, *The general linear model and the generalized singular value decomposition*, Linear Algebra Appl., 70 (1985), pp. 269-284.

[15] C. C. PAIGE AND M. A. SAUNDERS, *Towards a generalized singular value decomposition*, SIAM, J. Numer. Anal., 18 (1981), pp. 398-405.

[16] G. W. STEWART, *On the perturbation of pseudo-inverses, projections and linear least squares problems*, SIAM Rev., 19 (1977), pp. 634-662.

[17] ———, *Computing the CS decompostion of a partitioned orthonormal matrix*, Numer. Math., 40 (1982), pp. 297-306.

[18] C. F. VAN LOAN, *Generalizing the singular value decomposition*, SIAM J. Numer. Anal., 13 (1976), pp. 76-83.

[19] ———, *Computing the CS and generalized singular value decompositions*, Technical Report CS-604, Dept. Computer Science, Cornell Univ., Ithaca, NY, 1984.

[20] J. H. WILKINSON, *The Algebraic Eigenvalue Problem*, Clarendon Press, Oxford, 1965.

[21] C. C. PAIGE AND P. VAN DOOREN, *On the quadratic convergence of Kogbetliantz's algorithm for computing the singular value decomposition*, Linear Algebra Appl., to appear.

[22] W. M. GENTLEMAN, personal communication at the IXth Gatlinburg meeting on Numerical Linear Algebra, University of Waterloo, Ontario, Canada, July 8-14, 1984.

[23] G. W. STEWART, Computer Science Technical Report 1321, Univ. Maryland, College Park, MD, 1983.

SIAM J. SCI. STAT. COMPUT.
Vol. 7, No. 4, October 1986

006

COMPUTING THE SINGULAR VALUE DECOMPOSITION OF A PRODUCT OF TWO MATRICES*

M. T. HEATH†, A. J. LAUB‡, C. C. PAIGE§ AND R. C. WARD†

Abstract. An algorithm is developed for computing the singular value decomposition of a product of two general matrices without explicitly forming the product. The algorithm is based on an earlier Jacobi-like method due to Kogbetliantz and uses plane rotations applied to the two matrices separately. A triangular variant of the basic algorithm is developed that reduces the amount of work required.

Key words. singular value decomposition, plane rotations, matrix product, Jacobi, Kogbetliantz

AMS(MOS) subject classification. 65F

1. Introduction. The singular value decomposition is one of the most useful and powerful tools of numerical linear algebra and arises in many application areas such as statistical analysis, image processing, and control theory. The singular value decomposition (SVD) of an $m \times n$ matrix A has the form

$$A = U\Sigma V^T, \tag{1.1}$$

where U and V are orthogonal matrices of order m and n, respectively, and Σ is an $m \times n$ nonnegative diagonal matrix. (For convenience we assume $m \geqq n$; if this is not the case, we work with the transpose. We also assume that all matrices are real, but all of our developments also apply to complex matrices with the obvious changes, e.g., unitary replaces orthogonal, conjugate transpose replaces transpose, etc.) The diagonal elements σ_i of Σ are called the singular values of A and by convention are ordered so that $\sigma_1 \geqq \sigma_2 \geqq \cdots \sigma_n \geqq 0$. The singular values of A are the nonnegative square roots of the eigenvalues of the symmetric matrix A^TA, and the columns of U and V are orthonormal eigenvectors of AA^T and A^TA, respectively. For purposes of numerical computation, however, explicit formation of either of these product matrices is inadvisable because of a potential loss of information in finite precision arithmetic, as shown in the following simple example [11].

Let

$$A = \begin{bmatrix} 1 & 1 \\ \delta & 0 \\ 0 & \delta \end{bmatrix}, \tag{1.2}$$

with $\delta^2 < \varepsilon \ll \delta$, where ε is the relative floating-point machine precision. We then obtain

* This work was supported by the Applied Mathematical Sciences subprogram of the Office of Energy Research, U.S. Department of Energy under contract DE-AC05-84OR21400 with Martin Marietta Energy Systems, Inc., by the U.S. Air Force Office of Scientific Research and the National Science Foundation under grant ECS84-06152, and by the Canadian Natural Sciences and Engineering Research Council under grant A8652.

† Mathematical Sciences Section, Engineering Physics and Mathematics Division, Oak Ridge National Laboratory, P.O. Box Y, Oak Ridge, Tennessee 37831.

‡ Department of Electrical and Computer Engineering, University of California, Santa Barbara, California 93106.

§ Basser Department of Computer Science, Madsen Building F09, University of Sydney, N.S.W., Sydney, Australia, 2006.

the computed floating-point result

$$fl(A^TA) = \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}. \tag{1.3}$$

The matrix of (1.3) has eigenvalues 0 and 2, whereas the true eigenvalues of A^TA are δ^2 and $2+\delta^2$. Computing the singular values of A given by (1.2) with a numerically stable algorithm [11] can be expected to give values at least as good as $\delta + O(\varepsilon)$ and $(2+\delta^2)^{1/2} + O(\varepsilon)$, which is a much more satisfactory result than that obtained by taking square roots of the eigenvalues of (1.3).

Here we are concerned with the more difficult problem of computing the SVD of a product

$$A = B^TC$$

as accurately as possible for given B and C using a given precision. The preceding discussion indicates that this should be done without first forming $fl(B^TC)$. This problem arises in a number of applications, including the orthogonal Procrustes problem in statistics [12, pp. 425-426], which determines how closely two subspaces match under orthogonal transformations. Our interest in computing the SVD of a product was originally motivated by a problem in control theory that is sketched in an appendix to this paper; full details will be found in [14]. The problem arises more generally in the following form:

Problem. For a given $p \times m$ matrix B and $p \times n$ matrix C, find orthogonal matrices U and V of order m and n, respectively, such that the columns of

$$\tilde{B} = BU \text{ and } \tilde{C} = CV \tag{1.4}$$

form biorthogonal sets.

To solve this problem theoretically, consider the SVD

$$A = B^TC = U\Sigma V^T, \tag{1.5}$$

where U, V, and Σ are as in (1.1). Then

$$\tilde{B}^T\tilde{C} = U^TB^TCV = U^TAV = \Sigma,$$

and the problem is solved. If we consider the angle ψ_i between the ith columns of $\tilde{B}$ and $\tilde{C}$, we have

$$\tilde{b}_i^T\,\tilde{c}_i = \sigma_i = \|\tilde{b}_i\|_2\,\|\tilde{c}_i\|_2 \cos\psi_i, \tag{1.6}$$

so a small σ_i could result from a small $\|\tilde{b}_i\|$ or $\|\tilde{c}_i\|$, or from a ψ_i close to $\pi/2$. The $\cos\psi_i$ often provide important information about the problem. For example, they determine the condition of the problem considered in [14].

A special case of the biorthogonalization problem (1.4) is when each of B and C has orthonormal columns (such B and C may have resulted, for example, from orthogonalization of the columns of more general matrices), and then in (1.6) we have $\sigma_i = \cos\psi_i$, and the ψ_i are the canonical (principal) angles between the range spaces of B and C [12, pp. 428-429].

In order to develop an algorithm for computing the SVD in (1.5) accurately, we first discuss in §2 an algorithm originally due to Kogbetliantz [16] for computing the SVD of any given matrix A. Although the computational cost of this algorithm on traditional digital computers appears to be of the same order as that of the standard SVD algorithm [11], in practice it usually requires greater execution time. The algorithm of Kogbetliantz has the virtue of simplicity, however, so that it is more readily adaptable,

for example, to parallel computation [1] and, most importantly for our problem, it is ideally suited to computing the SVD in (1.5) without initially forming B^TC. In §3 we discuss the important subproblem of computing the SVD of a 2×2 matrix. In §4 we develop a triangular variant of the basic algorithm which reduces the amount of work required. In §5 we apply the basic algorithm and triangular variant to compute the SVD in (1.5). The paper concludes with a discussion of implementation details and test results and an appendix in which we sketch the application that motivated our work.

We will use the notation that a matrix A_k has columns $a_j^{(k)}$ and elements $\alpha_{ij}^{(k)}$.

2. A Jacobi-like SVD algorithm. A plane rotation matrix $J(i, j, \theta)$ of order n is equal to the identity matrix of order n except for the four elements in the intersections of rows and columns i and j, and this 2×2 submatrix has the form

$$\begin{bmatrix} c & s \\ -s & c \end{bmatrix},$$

where $c = \cos(\theta)$ and $s = \sin(\theta)$. These orthogonal matrices are useful tools in numerical linear algebra because they can be used to introduce zeros selectively into a matrix in a numerically stable manner, which is an important step toward reducing the matrix to some simpler form (e.g., triangular or diagonal).

Jacobi's method is an iterative algorithm for diagonalizing a symmetric matrix $A = A_1$ in which at iteration k a plane rotation $J_k = J(i_k, j_k, \theta_k)$ is chosen so as to annihilate a symmetric pair of off-diagonal elements:

$$\begin{bmatrix} c & -s \\ s & c \end{bmatrix}\begin{bmatrix} \alpha_{ii}^{(k)} & \alpha_{ij}^{(k)} \\ \alpha_{ji}^{(k)} & \alpha_{jj}^{(k)} \end{bmatrix}\begin{bmatrix} c & s \\ -s & c \end{bmatrix} = \begin{bmatrix} \alpha_{ii}^{(k+1)} & 0 \\ 0 & \alpha_{jj}^{(k+1)} \end{bmatrix}. \tag{2.1}$$

The off-diagonal mass of the matrix is thereby reduced at each iteration, the A_k converge to a diagonal matrix, and the product of the J_k converges to a matrix of orthogonal eigenvectors. If the off-diagonal elements are annihilated in a reasonable, systematic order (numerous strategies have been used, the most popular being of cyclic or threshold type), then the convergence rate is ultimately quadratic. Jacobi first published his method in 1845, and it was the standard algorithm for symmetric eigenproblems for about a century until being superseded by the somewhat more efficient symmetric QR algorithm. See [12, pp. 303-305] for references to Jacobi's original work and to the many convergence analyses, numerical refinements, and generalizations of this fascinating algorithm. There has been renewed interest in Jacobi's algorithm in recent years because its simplicity and compactness lend themselves to implementation on very small computers or to parallel computation.

Jacobi's algorithm has been generalized in various ways for nonsymmetric matrices. These include annihilating only one of the two off-diagonal elements at each iteration in computing the Schur decomposition and the use of nonorthogonal elementary matrices instead of rotations in diagonalizing nonnormal matrices. The most interesting generalization from our point of view is the use of different rotations on each side of A_k in order to compute the SVD of a (not necessarily square) matrix, in effect reducing the SVD computation to a sequence of 2×2 SVD's. This approach appears to have been suggested first by Kogbetliantz [16], has been analyzed by Forsythe and Henrici [6], has now become a textbook exercise [12, p. 301], and has been implemented for parallel computation [1]. We will refer to it as the Kogbetliantz algorithm.

We now describe the algorithm in more detail. For $i < j$ we define an (i, j) reduction of A_k to be a rotation $J(i, j, \theta_1)^T$ applied to rows i and j, and another rotation $J(i, j, \theta_2)$

applied to columns i and j, so that the resulting A_{k+1} has zeros in positions (i,j) and (j,i). These rotations are determined from the SVD of the 2×2 submatrix

$$\begin{bmatrix} c_1 & -s_1 \\ s_1 & c_1 \end{bmatrix}\begin{bmatrix} \alpha_{ii}^{(k)} & \alpha_{ij}^{(k)} \\ \alpha_{ji}^{(k)} & \alpha_{jj}^{(k)} \end{bmatrix}\begin{bmatrix} c_2 & s_2 \\ -s_2 & c_2 \end{bmatrix}=\begin{bmatrix} \alpha_{ii}^{(k+1)} & 0 \\ 0 & \alpha_{jj}^{(k+1)} \end{bmatrix}, \tag{2.2}$$

where $c_1=\cos(\theta_1)$, etc. (Strictly speaking, one must generalize either the notion of rotation or of SVD in order for this statement to be true; we return to this point in the next section.) For a square matrix $A=A_1$, Kogbetliantz [16] suggested applying such reductions in cycles of $n(n-1)/2$ steps in row- or column-major order. Thus, one sweep of the serial version of the cyclic algorithm is given by

```
for i := 1, ···, n−1
  for j := i+1, ···, n
    begin
    A_{k+1} := (i,j) reduction of A_k;
    k := k+1;
    end;
```
(2.3)

If A is symmetric, then this is just one sweep of the cyclic Jacobi algorithm for the symmetric eigenproblem. As with Jacobi's method, it appears experimentally that this algorithm has ultimate quadratic convergence, and usually requires fewer than 10 sweeps to converge to within machine precision. Clearly we could use other strategies for ordering the reduction steps, analogous to those suggested for the Jacobi algorithm. The algorithm is also applicable to a nonsquare $m\times n$ matrix A if we change the limits on the for-loops appropriately (e.g., if $m>n$, then $i:=1,\cdots,n$ and $j:=i+1,\cdots,m$) and, in the (i,j) reduction, omit the rotation on the right whenever $j>n$.

Kogbetliantz introduced his iterative diagonalization not specifically to compute the SVD, but in order to solve square systems of linear equations involving general and special complex matrices. He appears to have had a limited awareness of the importance of the SVD for other purposes. Kogbetliantz tested the algorithm experimentally on a new IBM 701, which diagonalized a 32×32 matrix with "extreme rapidity" in 23 minutes, including 4 minutes to print out the answers! He checked the accuracy of the method by reconstructing the original matrix from the diagonal matrix and accumulated orthogonal factors, and he found the results quite satisfactory.

3. SVD of a 2×2 matrix. Accurate and efficient computation of the SVD for 2×2 submatrices is the most important subproblem in the general SVD algorithm described in § 2. Several factors make it less than obvious how best to solve this subproblem. First, the angles θ_1 and θ_2 necessary to annihilate the off-diagonals in (2.2) are not unique. This ambiguity is usually resolved by restricting the domain of possible angles, and most rigorous convergence analyses of Jacobi-like methods have been based on such a restriction. Second, if we follow the usual straightforward recipe for computing the rotations, the resulting diagonal elements in (2.2) will not necessarily be ordered by magnitude, nor will they necessarily be nonnegative, and thus we do not have the true SVD in the strict sense of (1.1). The diagonal elements can always be reordered by permutations (which are special cases of rotations), but adjusting their signs may require a reflection. Thus we must either be content with this "unnormalized" SVD or allow 2×2 reflections as well as rotations. We will choose the former approach. A third factor to consider is that if the 2×2 input matrix is badly scaled, the 2×2 SVD computation can encounter numerical difficulties analogous to those for the (logically equivalent) problem of solving quadratic equations [5]. Thus, we must guard against overflow, cancellation error, etc.

Kogbetliantz [16] derived straightforward formulas for the necessary rotations in (2.2), but gave little attention to possible numerical difficulties. Forsythe and Henrici [6] suggested an approach to computing the rotations that leads to a numerically robust but fairly complicated algorithm that is stated in detail as algorithm FHSVD in [1]. A simpler but numerically quite effective approach is suggested by Golub and Van Loan (see [12, p. 301]), who propose first symmetrizing the 2×2 input matrix with a rotation, then diagonalizing the resulting symmetric matrix with a single rotation applied on both sides (as in (2.1)). The product of the diagonalizing and symmetrizing rotations then gives one of the two rotations sought, while the diagonalizing rotation alone gives the other. A detailed implementation of this idea is stated as algorithm USVD in [1], and this is essentially the algorithm we use. Brent, Luk and Van Loan [1] empirically found USVD to be as effective as FHSVD, and also found the "unnormalized" 2×2 SVD to be just as effective as the standard 2×2 SVD in the context of the Jacobi SVD algorithm. If we want the standard SVD as final result, then after the Kogbetliantz algorithm has converged we will need to reorder the diagonal elements of Σ and alter their signs, adjusting the orthogonal factors $U$and V accordingly.

4. SVD of a triangular matrix. Efficiency can often be gained in computing the SVD if the input matrix is first orthogonally transformed into triangular form [2]. This raises the question of how the Kogbetliantz algorithm behaves when applied to triangular matrices. The somewhat surprising answer is that one serial sweep of the form (2.3) transforms an upper triangular matrix into a lower triangular one, or a lower triangular matrix into an upper triangular one. Thus, as iterations proceed, the matrices A_k alternate between upper and lower triangular forms. We illustrate this phenomenon for a 4×4 upper triangular matrix in Fig. 1, which shows one complete sweep of (2.3) with original nonzeros indicated by $\times$, zeros by blank, newly created nonzeros (fill) by $+$, and the nonzero to be eliminated next is circled (there is only one rather than two because the matrix is triangular).

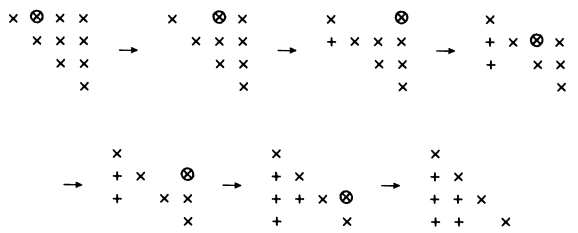


FIG. 1. *Transformation from upper to lower triangular form by one sweep.*

The proof that the same transformation occurs for general n is by induction. Suppose one serial sweep on an $(n-1)\times(n-1)$ upper triangular matrix results in a lower triangular matrix. For an $n\times n$ upper triangular matrix the $(1,2)$ reduction eliminates the $(1,2)$ element, while for $j=3,\cdots,n$ the $(1,j)$ reduction eliminates the $(1,j)$ element and introduces a $(j-1,1)$ element. Thus the trailing $(n-1)\times(n-1)$block is still upper triangular after these reductions. The remaining reductions serially transform this trailing block to lower triangular form, and, since these combine only columns $2,\cdots,n$ and rows $2,\cdots,n$, the off-diagonal zeros in the first row remain. Thus, the resulting $n\times n$ matrix is lower triangular. Applying the reductions to a lower triangular matrix is equivalent to applying the above process to its transpose, and our result follows.

Using r-multiplication rotations, where $r = 2$, 3, or 4 (see [8], [13]), the method on a full $n \times n$ matrix A requires $n(n-1)/2$ SVD's of full 2×2 matrices, and about rn^3 multiplications and n^3 additions per sweep to apply the rotations to produce the new A. For triangular A the method requires $n(n-1)/2$ SVDs of triangular 2×2 matrices, and about $rn^3/2$ multiplications and n^3 additions per sweep to apply the rotations to produce the new triangular A. In either case, we will need an additional $rn^3/2$ multiplications and $n^3/2$ additions per sweep to accumulate each of U and V in (1.1), if desired. Since in practice we have found that about six to eight sweeps are required for full accuracy, even this more efficient triangular variant of the Kogbetliantz algorithm is not competitive with the standard algorithm [11] for large n on a general purpose computer. However, initial tests suggest that it is just as accurate and reliable, and therefore its compactness and simplicity may give it some utility for small n on small computers. An overriding advantage of the Kogbetliantz algorithm for our purposes is its applicability to computing the SVD of a product of matrices.

Although the triangular variant of the Kogbetliantz algorithm reduces the work required, it seems to restrict the order of eliminations to the serial ordering of (2.3), and thus rules out an ordering suited to parallel computation, such as those in [22] and [1]. However, if the eliminations are accompanied by pairwise interchanges of rows and columns as in [24], then a parallel implementation is possible, as was pointed out to the authors by Gentleman [9]. Another interesting possibility suggested by the triangular variant is to store the matrix in a packed triangular form in a one-dimensional array (rather than a rectangular array), performing the algorithm in place, and thereby conserving storage. Unfortunately, such an approach lessens the simplicity of coding that is one of the nicest features of the algorithm, makes a parallel implementation more difficult, and is of questionable value since the triangular matrix will usually have resulted from preliminary factorization of a general matrix that would require rectangular storage anyway.

5. SVD of a product of matrices. We now turn to the application of the Kogbetliantz algorithm to compute the SVD of a product of two matrices. We first describe the algorithm as applied to $A = B^T C$ without any initial transformations of B or C. At step k we have B_k and C_k, and the four elements needed in (2.2) are computed as in

$$\alpha_{ij}^{(k)} := b_i^{(k)T} c_j^{(k)}.$$

The resulting rotations are then applied separately to B_k and C_k to obtain

$$B_{k+1} := B_k J(i, j, \theta_1) \text{ and } C_{k+1} := C_k J(i, j, \theta_2).$$

We make several observations about this approach.

1. Since this algorithm is theoretically equivalent to the original algorithm for A_k, the reduction of off-diagonal mass at each step and consequent convergence properties are theoretically the same.
2. Only two columns of B_k and two columns of C_k are touched (read or modified) at each step.
3. Only four elements of A_k are computed at each step, and even this can be improved, since two of the four are always diagonal elements and these may be known from previous steps. In particular, if we make use of the knowledge of $\alpha_{ii}^{(k+1)}$ and $\alpha_{jj}^{(k+1)}$, we require four vector inner products in designing the (1,2) reduction, but only three inner products for each of the $(1,3), \cdots, (1,n)$ reductions, and only two inner products for each of the remaining reductions in a sweep.

4. A seemingly unsatisfactory aspect is that we actually form part of $B_k^T C_k$ at each step, and our original aim was to avoid forming $B^T C$. This is not too dangerous, however, since the few elements we form are used only in designing the rotations, and the rotations are then applied to the individual B_k and C_k. Thus, any information lost in computing these $\alpha_{ij}^{(k)}$ may have a small effect on convergence, but should not affect the ultimate accuracy of the computed results. A similar situation is encountered in [11] for computing the SVD of general A.
5. When $B = C$, this method effectively specializes to the one-sided method of Hestenes [15] for computing the SVD of B. See also [3] and [19].

As with the original algorithm, the cost per sweep can be reduced for the product algorithm by using the triangular variant described in § 4. This requires a preliminary orthogonal transformation of B and C into triangular form, which we now describe. Recall that we are assuming that B is $p \times m$ and C is $p \times n$, with $m \geqq n$. Thus there are two cases:

i). $p \geqq n$: Choose an orthogonal matrix Q of order p such that

$$Q^T C = \begin{bmatrix} C_1 \\ 0 \end{bmatrix},$$

with C_1 upper triangular and $n \times n$. If we partition

$$Q^T B = \begin{bmatrix} \hat{B} \\ \bar{B} \end{bmatrix},$$

similarly, then

$$B^T Q Q^T C = [\hat{B}^T, \bar{B}^T] \begin{bmatrix} C_1 \\ 0 \end{bmatrix} = \hat{B}^T C_1.$$

ii). $n > p$: Choose an orthogonal matrix Q of order n such that

$$CQ = [C_1, 0],$$

with C_1 upper triangular and $p \times p$. Then

$$B^T C Q = [\hat{B}^T C_1, 0],$$

where in this case we merely take $\hat{B} = B$.

In either case we now require only the SVD of $\hat{B}^T C_1$, so we choose an orthogonal matrix P of order m such that

$$P^T \hat{B}^T = \begin{bmatrix} B_1^T \\ 0 \end{bmatrix},$$

with B_1^T upper triangular and either $n \times n$ (case i) or $p \times p$ (case ii). Thus, all cases reduce to the problem of computing the SVD of the triangular matrix

$$A_1 = B_1^T C_1, \tag{5.1}$$

where B_1^T and C_1 are upper triangular, and all three matrices are either $n \times n$ or $p \times p$, depending on whether n or p is smaller. For definiteness, we henceforth assume that B_1 and C_1 in (5.1) are $n \times n$. Note that in computing these orthogonal transformations of the original problem we need make no assumptions or decisions regarding rank. We simply carry through with standard orthogonal factorization procedures, and any zero or small singular values will be revealed in the subsequent iterative phase.

The triangular variant of the Kogbetliantz algorithm is now applicable to A_1, but unfortunately the preservation of triangularity upon which the triangular variant depends does not hold for the triangular factors B_1^T and C_1. We therefore use an additional rotation at each step to restore both factors to triangular form, as we now show. At a given step k, after the (i,j) reduction has implicitly annihilated $\alpha_{ij}^{(k)}$ and $\alpha_{ji}^{(k)}$using rotations $J(i,j,\theta_1)$ and $J(i,j,\theta_2)$, we choose a third rotation $J(i,j,\theta_3)$ which annihilates the (i,j) elements of both $J(i,j,\theta_1)^T B_k^T$and $J(i,j,\theta_2)C_k$. The net result is that we implicitly have

$$A_{k+1}=\{J(i,j,\theta_1)^T B_k^T J(i,j,\theta_3)^T\}\{J(i,j,\theta_3)C_k J(i,j,\theta_2)\}=B_{k+1}^T C_{k+1}, \tag{5.2}$$

with B_{k+1} and C_{k+1} having a nonzero pattern which, like that of A_{k+1}, preserves triangularity over a full sweep. We illustrate the details of this procedure by means of a 3×3 example shown in Fig. 2, beginning a sweep with upper triangular matrices and finishing with lower triangular ones. The first two columns of matrices in the figure show operations explicitly performed on B_k^T and C_k, while the third column shows the implicit operations for A_k. Original nonzeros are indicated by $\times$, zeros by blank, newly created nonzeros (fill) by $+$, and nonzeros to be eliminated next are circled.

To see why a single rotation $J(i,j,\theta_3)$ suffices to annihilate elements of both B_1 and C_1(after $J(i,j,\theta_1)$ and $J(i,j,\theta_2)$ have been applied), consider the first step in Fig. 2. Although the (1, 2) reduction annihilates the (1, 2) element of A_1, the (1, 2) elements of B_1^T and C_1 remain nonzero in general, and their (2,1) elements become nonzero

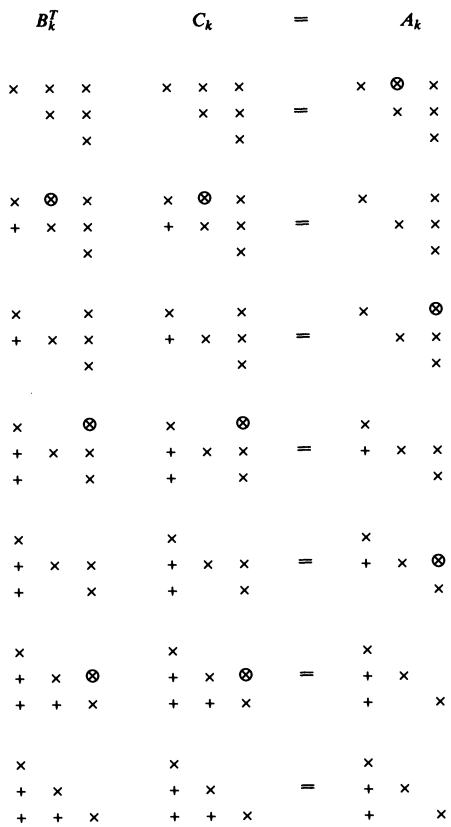


FIG. 2. *Transformation from upper to lower triangular form.*

as well. If we now design a rotation $J(1, 2, \theta_3)$ based on C_1 to make the (1, 2) element $\gamma_{12}^{(2)}$ of C_2 zero, and the resulting (2, 2) element $\gamma_{22}^{(2)}$ of C_2 is nonzero, then the (1, 2) element $\beta_{21}^{(2)}$ of B_2^T must necessarily be zero, since

$$\begin{bmatrix} \alpha_{11}^{(2)} & 0 \\ 0 & \alpha_{22}^{(2)} \end{bmatrix} = \begin{bmatrix} \beta_{11}^{(2)} & \beta_{21}^{(2)} \\ \beta_{12}^{(2)} & \beta_{22}^{(2)} \end{bmatrix} \begin{bmatrix} \gamma_{11}^{(2)} & 0 \\ \gamma_{21}^{(2)} & \gamma_{22}^{(2)} \end{bmatrix}.$$

If instead the rotation is based on B_1 and designed to make the (1, 2) element of B_2^T zero, and the resulting (1, 1) element of B_2 is nonzero, then the (1, 2) element of C_2 must necessarily be zero. We can choose which of the two matrices to use in designing the rotation by comparing magnitudes of the β_{11} and γ_{22} that would result. Thus we can design the rotation on B_1 if, for example,

$$|\beta_{11}^{(1)}|^2 + |\beta_{21}^{(1)}|^2 \geqq |\gamma_{12}^{(1)}|^2 + |\gamma_{22}^{(1)}|^2,$$

and otherwise on C_1. In this way we can guarantee that $J(i, j, \theta_3)$ annihilates the (i, j) elements of both $J(i, j, \theta_1)^T B_k^T$ and $C_k J(i, j, \theta_2)$. The proof that this approach works for higher dimensional matrices is similar to the induction proof in § 4.

Because of the form of the A_k, B_k and C_k, we have for each (i, j) reduction

$$\begin{bmatrix} \alpha_{ii}^{(k)} & \alpha_{ij}^{(k)} \\ 0 & \alpha_{jj}^{(k)} \end{bmatrix} = \begin{bmatrix} \beta_{ii}^{(k)} & \beta_{ji}^{(k)} \\ 0 & \beta_{jj}^{(k)} \end{bmatrix} \begin{bmatrix} \gamma_{ii}^{(k)} & \gamma_{ij}^{(k)} \\ 0 & \gamma_{jj}^{(k)} \end{bmatrix}, \tag{5.3}$$

which is superior to forming three inner products between vectors of p elements each. In fact, the three rotations $J(i, j, \theta_1)$, $J(i, j, \theta_2)$, and $J(i, j, \theta_3)$ can all be designed from the six elements in the right side of (5.3). Together they transform (5.3) into

$$\begin{bmatrix} \alpha_{ii}^{(k+1)} & 0 \\ 0 & \alpha_{jj}^{(k+1)} \end{bmatrix} = \begin{bmatrix} \beta_{ii}^{(k+1)} & 0 \\ \beta_{ij}^{(k+1)} & \beta_{jj}^{(k+1)} \end{bmatrix} \begin{bmatrix} \gamma_{ii}^{(k+1)} & 0 \\ \gamma_{ji}^{(k+1)} & \gamma_{jj}^{(k+1)} \end{bmatrix}.$$

Thus the triangular variant of the Kogbetliantz algorithm is applicable to a product of triangular matrices, and the triangular form of the factors can be preserved. Note that the matrices B_k^T and C_k do not converge to diagonal matrices, in general, even though their product does.

As was remarked earlier, the algorithm we have just presented is theoretically equivalent to the Kogbetliantz SVD algorithm, the convergence of which is shown in [6]. In [20] an ultimately quadratic convergence rate for this algorithm is established, provided there are no pathologically close singular values. Since the matrices B and C are transformed separately using only unitary matrices, standard error analysis shows that the computed triangular factors are exact for initial matrices very close to B and C. Thus, the product algorithm is a numerically stable implementation of a convergent iteration. A simple bound on the error in the singular values computed by this algorithm is no better than that for the conventional SVD algorithm applied to the explicitly computed product, and indeed we expect no greater accuracy for the larger singular values. If there is substantial variation in magnitude of the singular values, however, then the product algorithm usually has greater accuracy for computing the smaller singular values since the error in the computed elements on which the rotations depend is not necessarily dominated by the errors in the largest elements of the two matrices. The numerical test results given in the next section bear out these expectations.

6. Test results and conclusions. We have written a Fortran program implementing the triangular variant of the product algorithm developed in §5. In our code, the preliminary orthogonal transformations of B^T and C to upper triangular form are

performed using the orthogonal factorization package of Wright and Glassman [26]. After the preliminary triangularization, the algorithm proceeds in a sequence of double sweeps, each of which transforms both matrices from upper triangular form to lower triangular form and then back to upper triangular form. To solve the 2×2 SVD subproblems we explicitly form the triangular 2×2 product in (5.3), then apply algorithm USVD of [1]. Note that the latter takes no advantage of triangularity; indeed, the symmetrization step destroys triangularity. We have found no numerically stable approach to the 2×2 subproblem which avoids explicitly forming the product (5.3) and takes significant advantage of triangularity.

An individual (i, j) reduction step is skipped if the 2×2 matrix to be diagonalized is already diagonal to within some relative tolerance, which we always take to be the relative floating-point machine precision. Convergence is declared if all reductions are skipped for an entire sweep. By maintaining the two matrices B_1^T and C_1 separately throughout, at convergence we have produced two triangular matrices (not necessarily diagonal) whose product gives the diagonal matrix Σ of (1.5). This product need not necessarily be formed unless we are interested in the singular values themselves; for example, we might want merely to compute the matrices in (1.4). The orthogonal matrices U and V of (1.5) are formed by accumulating the rotations $J(i, j, \theta_1)$ and $J(i, j, \theta_2)$ from each (i, j) reduction step on the orthogonal matrices P and Q that initially triangularized $\hat{B}^T$ and C, respectively.

To test the algorithm we generated pairs of random matrices whose products have known singular values. Specifically, let Σ_1 and Σ_2 be known nonnegative diagonal matrices of order $p\times m$ and $p\times n$, respectively. Using the technique of Stewart [23], generate random orthogonal matrices W_1, W_2, and W_3 of order p, m, and n, respectively. If we now take

$$B = W_1\Sigma_1 W_2^T \quad \text{and} \quad C = W_1\Sigma_2 W_3^T, \tag{6.1}$$

then the singular values of B^TC are given by the diagonal elements of $\Sigma_1^T\Sigma_2$. In this way we can generate random test problems having whatever distribution of singular values and whatever relative weighting of the two matrices we may desire.

All computations were carried out on a DEC Vax 11/780, which has a relative floating-point precision of about 6.0×10^{-8} in single precision and about 1.4×10^{-17} in double precision. In order to emphasize rounding effects, we tested our algorithm primarily in single precision. For comparison, we explicitly computed in single precision the product B^TC and then computed its SVD using the single precision LINPACK [4] routine SSVDC, which uses an algorithm based on [11]. As an accuracy benchmark, we also computed in double precision the product B^TC and then computed its SVD using the analogous double precision LINPACK routine DSVDC.

In order to make the input matrices B and C as accurate as possible, the computations in (6.1) were carried out in double precision, and the resulting B and C rounded to single precision. Even so, the singular values of the product as computed by any algorithm cannot be expected to agree exactly with those of the input $\Sigma_1^T\Sigma_2$ due to the inexact representation of the input matrices B and C. Thus, the double precision results using DSVDC for the same input matrix should be used in judging the accuracy of the single precision results.

For problems having well separated singular values, our algorithm usually converges to within machine precision in about three or four double sweeps. For problems having multiple or nearly multiple singular values, however, convergence can be slightly degraded as the order and multiplicity become larger. These conclusions are illustrated in Table 1, which gives the average number of double sweeps required for convergence

over an ensemble of 20 random problems for each of several orders of matrices (with $p = m = n$) and distributions of singular values ($\sigma_i \sim U[0,1]$ means that the singular values are random numbers from a uniform distribution on the interval [0,1]). The high multiplicity case, in which all of the singular values are about 1 or 2, requires approximately one extra double sweep for the larger problems to attain the same level of convergence. See [25] for a discussion of the effect of multiple eigenvalues on Jacobi type algorithms.

TABLE 1
Average number of double sweeps for convergence.

Singular values	Order of matrices			
	5	10	20	40
$\sigma_i \sim U[0,1]$	2.20	2.95	3.10	4.00
$\sigma_i = i^{-2}$	2.10	2.95	3.00	3.00
$\sigma_i = i^{-1}$	2.10	3.00	3.00	3.60
$\sigma_i = i$	2.05	3.00	3.05	4.00
$\sigma_i = 1 + mod(i,2)$	1.65	2.80	3.85	4.70

It is somewhat difficult to monitor the detailed convergence behavior, since in the normal course of the algorithm we never explicitly form the matrix whose off-diagonal mass is being annihilated (recall that the factors themselves do not become diagonal). As a check in our experimental code, however, we computed the off-diagonal mass of the product after each sweep (i.e., each half of a double sweep) and observed a quadratic asymptotic convergence rate in all cases, including those having multiple singular values. By a quadratic asymptotic convergence rate we mean that ultimately the magnitude of the off-diagonal mass of the product, as measured by whatever norm, is squared by each successive sweep.

For problems having all singular values well above machine precision, our algorithm produces results which are similar in accuracy to those obtained by explicitly forming the product B^TC and applying the standard algorithm. Our algorithm shows the expected superiority, however, in computing accurately very small singular values of the product. A typical example, with $p = m = n = 6$ and $\sigma_i = 10^{-2(i-1)}$, is shown in Table 2, in which the input Σ_1 and Σ_2 are shown, along with the computed singular values resulting from our product algorithm, SSVDC, and DSVDC. The singular values computed by SSVDC from the explicitly formed single precision product matrix show a steady decline in accuracy, the smallest computed singular value having no significant

TABLE 2
Computed singular values.

Σ_1	Σ_2	Prod. Alg.	SSVDC	DSVDC
1.0	1.0	1.000001	1.000000	1.00000002514
1.0e-1	1.0e-1	1.000002e-2	1.000000e-2	1.00000003954e-2
1.0e-2	1.0e-2	1.000001e-4	1.000082e-4	.999999359846e-4
1.0e-3	1.0e-3	1.000003e-6	.9988159e-6	.999999805948e-6
1.0e-4	1.0e-4	1.000048e-8	2.843042e-8	1.00002308186e-8
1.0e-5	1.0e-5	1.000767e-10	7.477041e-10	1.00005751845e-10

digits of accuracy. The product algorithm, though suffering some loss of accuracy, still provides about three digits of accuracy in the smallest singular value despite the use of single precision computations throughout.

We conclude that the algorithm we have presented is a useful alternative for computing the SVD of a product of two matrices, and is to be preferred when explicit formation of that product would cause a serious loss of information in finite precision arithmetic.

Appendix. The need to compute the SVD of a product of two matrices arises naturally from a problem in linear systems theory involving the computation of system (or state) balancing transformations. That problem and related aspects are discussed in detail in, for example, [10], [18], and [21]. Specific details of the application of our algorithm to system balancing are described in [14].

Briefly, the basic mathematical computation involved is that of computing a contragredient transformation T. More specifically, given two symmetric positive definite matrices F and G, we seek a contragredient transformation T such that

$$T^{-1}FT^{-T} = T^TGT = \Lambda,$$

with Λ diagonal. One approach to this problem is to let F have the Cholesky decomposition

$$F = LL^T,$$

then compute the symmetric eigendecomposition

$$U^T(L^TGL)U = \Lambda^2,$$

where U is orthogonal and Λ is diagonal with positive diagonal elements. Then

$$T = LU\Lambda^{-1/2}$$

is the desired contragredient balancing transformation. This method was proposed in [17] for system balancing and is also the basis for EISPACK [7] routines such as REDUC2 and REBAKB for the generalized eigenproblems $ABx = \lambda x$ and $BAx = \lambda x$.

Alternatively, and this is the approach taken in [14], one may start with the Cholesky factorizations

$$F = LL^T \text{ and } G = MM^T$$

for both F and G (or these may already be available directly; cf. the application described in [14]). Now compute the SVD of the product

$$L^TM = U\Lambda V^T$$

using the algorithm developed in the present paper. Then the contragradient balancing transformation is given by

$$T = LU\Lambda^{-1/2},$$

and note also that

$$T^{-1} = \Lambda^{-1/2}V^TM^T.$$

If the Cholesky factors L and M are available, we have, in effect, solved the generalized eigenproblem $FGx = \lambda x$ without forming any matrix products explicitly, for it may be verified that

$$T^{-1}FGT = \Lambda^2,$$

where T and its inverse are as given above.

REFERENCES

[1] R. P. BRENT, F. T. LUK AND C. F. VAN LOAN, *Computation of the singular value decomposition using mesh-connected processors*, Report No. TR 82-528, Dept. of Computer Science, Cornell University, Ithaca, NY, March 1983.

[2] T. F. CHAN, *An improved algorithm for computing the singular value decomposition*, ACM Trans. Math. Software, 8 (1982), pp. 72-83.

[3] B. A. CHARTRES, *Adaptation of the Jacobi method for a computer with magnetic tape backing store*, Comput. J., 5 (1962), pp. 51-60.

[4] J. J. DONGARRA, J. R. BUNCH, C. B. MOLER AND G. W. STEWART, LINPACK *Users' Guide*, Society for Industrial and Applied Mathematics, Philadelphia, 1979.

[5] G. E. FORSYTHE, *What is a satisfactory quadratic equation solver*, in Constructive Aspects of the Fundamental Theorem of Algebra, B. Dejon and P. Henrici, eds., Wiley-Interscience, New York, 1969, pp. 53-61.

[6] G. E. FORSYTHE AND P. HENRICI, *The cyclic Jacobi method for computing the principal values of a complex matrix*, Trans. Amer. Math. Soc., 94 (1960), pp. 1-23.

[7] B. S. GARBOW, J. M. BOYLE, J. J. DONGARRA AND C. B. MOLER, *Matrix Eigensystem Routines—* EISPACK *Guide Extension*, Springer-Verlag, New York, 1977.

[8] W. M. GENTLEMAN, *Least squares computations by Givens transformations without square roots*, J. Inst. Math. Appl., 12 (1973), pp. 329-336.

[9] W. M. GENTLEMAN, personal communication, Waterloo, Ontario, Canada, July 1984.

[10] K. GLOVER, *All optimal Hankel-norm approximations of linear multivariable systems and their L^∞-error bounds*, Internat. J. Control, 39 (1984), pp. 1115-1193.

[11] G. H. GOLUB AND C. REINSCH, *Singular value decomposition and least squares solutions*, Numer. Math., 14 (1970), pp. 403-420.

[12] G. H. GOLUB AND C. F. VAN LOAN, *Matrix Computations*, Johns Hopkins University Press, Baltimore, 1983.

[13] S. HAMMARLING, *A note on modifications to the Givens plane rotation*, J. Inst. Math. Appl., 13 (1974), pp. 215-218.

[14] M. T. HEATH, A. J. LAUB, C. C. PAIGE AND R. C. WARD, *Computation of system balancing transformations*, IEEE Trans. Automat. Control, to appear.

[15] M. R. HESTENES, *Inversion of matrices by biorthogonalization and related results*, J. Soc. Indust. Appl. Math., 6 (1958), pp. 51-90.

[16] E. G. KOGBETLIANTZ, *Solution of linear equations by diagonalization of coefficients matrix*, Quart. Appl. Math., 13 (1955), pp. 123-132.

[17] A. J. LAUB, *Computation of "balancing" transformations*, Proc. Joint Automat. Control Conf., San Francisco, June 1980, pp. FA8-E.

[18] B. C. MOORE, *Principal component analysis in linear systems: controllability, observability, and model reduction*, IEEE Trans. Automat. Control, AC-26 (1981), pp. 17-32.

[19] J. C. NASH, *A one-sided transformation method for the singular value decomposition and algebraic eigenproblem*, Comput. J., 18 (1975), pp. 74-76.

[20] C. C. PAIGE AND P. VAN DOOREN, *On the quadratic convergence of Kogbetliantz's algorithm for computing the singular value decomposition*, Linear Algebra Appl., 77 (1986), pp. 301-313.

[21] L. PERNEBO AND L. M. SILVERMAN, *Model reduction via balanced state space representations*, IEEE Trans. Automat. Control, AC-27 (1982), pp. 382-387.

[22] A. SAMEH, *On Jacobi and Jacobi-like algorithms for a parallel computer*, Math. Comp. 25 (1971), pp. 579-590.

[23] G. W. STEWART, *The efficient generation of random orthogonal matrices with an application to condition estimators*, SIAM J. Numer. Anal., 17 (1980), pp. 403-409.

[24] G. W. STEWART, *A Jacobi-like algorithm for computing the Schur decomposition of a non-Hermitian matrix*, this Journal, 6 (1985), pp. 853-864.

[25] J. H. WILKINSON, *Almost diagonal matrices with multiple or close eigenvalues*, Linear Algebra Appl., 1 (1968), pp. 1-12.

[26] M. H. WRIGHT AND S. C. GLASSMAN, *Fortran subroutines to solve the linear least-squares problem and compute the complete orthogonal factorization*, Report No. SOL-78-8, Dept. of Operations Research, Stanford University, Stanford, CA, April 1978.

SIAM J. SCI. STAT. COMPUT.
Vol. 7, No. 4, October 1986

COMPUTING THE POLAR DECOMPOSITION—WITH APPLICATIONS*

NICHOLAS J. HIGHAM†

Abstract. A quadratically convergent Newton method for computing the polar decomposition of a full-rank matrix is presented and analysed. Acceleration parameters are introduced so as to enhance the initial rate of convergence and it is shown how reliable estimates of the optimal parameters may be computed in practice.

To add to the known best approximation property of the unitary polar factor, the Hermitian polar factor H of a nonsingular Hermitian matrix A is shown to be a good positive definite approximation to A and $\frac{1}{2}(A+H)$ is shown to be a best Hermitian positive semi-definite approximation to A. Perturbation bounds for the polar factors are derived.

Applications of the polar decomposition to factor analysis, aerospace computations and optimisation are outlined; and a new method is derived for computing the square root of a symmetric positive definite matrix.

Key words. polar decomposition, singular value decomposition, Newton's method, matrix square root

AMS(MOS) subject classifications. 65F25, 65F30, 65F35

1. Introduction.

The polar decomposition is a generalisation to matrices of the familiar complex number representation $z = r e^{i\theta}$, $r \geqq 0$.

THEOREM 1.1. *Polar Decomposition. Let $A \in \mathbb{C}^{m\times n}$, $m \geqq n$. Then there exists a matrix $U \in \mathbb{C}^{m\times n}$ and a unique Hermitian positive semi-definite matrix $H \in \mathbb{C}^{n\times n}$ such that*

$$A = UH, \qquad U^*U = I_n.$$

If rank $(A) = n$ *then H is positive definite and U is uniquely determined.*

The decomposition is well known and can be found in many textbooks, for example, [13], [16], [27]. An early reference is [1].

It is well known that the polar factor U possesses a best approximation property (see § 2.2). Less attention has been paid in the literature to the Hermitian polar factor H. We derive some interesting properties of H which show that when A is nonsingular and Hermitian, H is a good Hermitian positive definite approximation to A and $\frac{1}{2}(A+H)$ is a best Hermitian positive semi-definite approximation to A.

In view of the properties possessed by the polar factors of a matrix, techniques for computing the polar decomposition are of interest. While U and H can be obtained via the singular value decomposition (see § 3.1), this approach is not always the most efficient (if $A \approx U$, as explained in § 6.2) or the most convenient (a library routine for computing the singular value decomposition might not be available, on a microcomputer, for example).

In § 3 we present and analyse a Newton method for computing the polar decomposition which involves only matrix additions and matrix inversions. The method is shown to be quadratically convergent. Acceleration parameters are introduced so as to enhance the initial rate of convergence and it is shown how reliable estimates of the optimal parameters may be computed in practice. The stability of the method is considered in § 4. In § 5 the relationship of the method to two well-known iterations is described.

* Received by the editors March 13, 1985. This work was carried out with the support of a SERC Research Studentship.

† Department of Mathematics, University of Manchester, Manchester M13 9PL, England.

In § 6 we describe applications of the polar decomposition to factor analysis, aerospace computations and optimisation. We show how our algorithm may be employed in these applications and compare it with other methods in use currently. A new method for computing the square root of a symmetric positive definite matrix is derived.

2. Properties of the polar decomposition.

2.1. Elementary properties. We begin by noting the close relationship of the polar decomposition to the singular value decomposition. Let $A \in \mathbb{C}^{m\times n}$, $m \geqq n$, have the singular value decomposition [16, p. 16]

$$A = P\begin{pmatrix}\Sigma\\ 0\end{pmatrix}Q^*, \tag{2.1}$$

where $P \in \mathbb{C}^{m\times m}$ and $Q \in \mathbb{C}^{n\times n}$ are unitary and

$$\Sigma = \operatorname{diag}(\sigma_1, \sigma_2, \cdots, \sigma_n), \qquad \sigma_1 \geqq \sigma_2 \geqq \cdots \geqq \sigma_n \geqq 0.$$

Partitioning

$$P = [\underset{n}{P_1}, \underset{m-n}{P_2}], \qquad P_1^* P_1 = I_n,$$

it follows that A has the polar decomposition $A = UH$, where

$$U = P_1 Q^*, \tag{2.2}$$

$$H = Q\Sigma Q^*. \tag{2.3}$$

Conversely, given the polar decomposition $A = UH \in \mathbb{C}^{n\times n}$, from a spectral decomposition $H = Q\Sigma Q^*$ ($Q^*Q = I$) one can construct the singular value decomposition $A = (UQ)\Sigma Q^*$.

Several interesting properties of the polar decomposition are displayed in Lemma 2.1. Our notation is as follows. For $A \in \mathbb{C}^{n\times n}\lambda(A)$ and $\sigma(A)$ denote, respectively, the set of eigenvalues and the set of singular values of A, and $\kappa_2(A) = \sigma_1/\sigma_n$ is the 2-norm condition number. $C^{1/2}$ denotes the unique Hermitian positive semi-definite square root of the Hermitian positive semi-definite matrix C [20], [27]. A is normal if $A^*A = AA^*$ [16, p. 193].

LEMMA 2.1. *Let $A \in \mathbb{C}^{n\times n}$ have the polar decomposition $A = UH$. Then*

(i) $H = (A^*A)^{1/2}$.

(ii) $\lambda(H) = \sigma(H) = \sigma(A)$.

(iii) $\kappa_2(H) = \kappa_2(A)$.

(iv) *A is normal if and only if $UH = HU$.*

Proof. (i)-(iii) are immediate. For (iv) see [13, p. 276].

2.2. The unitary polar factor. Our interest in the polar decomposition stems from a best approximation property possessed by the unitary factor. The following result displays this property; it is the generalisation to complex matrices of a result in [16, p. 425] (see also [9], [15], [17], [26], [31]). The Frobenius matrix norm is defined by

$$\|A\|_F = (\operatorname{trace}(A^*A))^{1/2}.$$

THEOREM 2.2. *Let $A, B \in \mathbb{C}^{m\times n}$ and let $B^*A \in \mathbb{C}^{n\times n}$ have the polar decomposition*

$$B^*A = UH.$$

Then for any unitary $Z \in \mathbb{C}^{n\times n}$,

$$\|A - BU\|_F \leqq \|A - BZ\|_F \leqq \|A + BU\|_F. \tag{2.4}$$

$$\|A \pm BU\|_F^2 = \sum_{i=1}^{n} (\sigma_i(A)^2 \pm 2\sigma_i(B^*A) + \sigma_i(B)^2). \tag{2.5}$$

An important special case of Theorem 2.2 is obtained by taking $m = n$ and $B = I$.

COROLLARY 2.3. *Let $A \in \mathbb{C}^{n\times n}$ have the polar decomposition*

$$A = UH.$$

Then for any unitary $Z \in \mathbb{C}^{n\times n}$,

$$\left(\sum_{i=1}^{n} (\sigma_i(A)-1)^2\right)^{1/2} = \|A-U\|_F \leqq \|A-Z\|_F \leqq \|A+U\|_F = \left(\sum_{i=1}^{n} (\sigma_i(A)+1)^2\right)^{1/2}.$$

Thus if distance is measured in the Frobenius norm, the nearest unitary matrix to $A \in \mathbb{C}^{n\times n}$ is the unitary factor in the polar decomposition of A; and the furthest unitary matrix from A is minus this unitary factor. This result was established by Fan and Hoffman [12] for any unitarily invariant norm (thus it is valid for the 2-norm). It is not hard to show that Corollary 2.3 remains true when $A \in \mathbb{C}^{m\times n}$ with $m > n$ (this does not follow immediately from Theorem 2.2).

2.3. The Hermitian polar factor. As well as yielding a closest unitary matrix, the polar decomposition provides information about nearby Hermitian positive (semi-) definite matrices.

Let $A \in \mathbb{C}^{n\times n}$ be Hermitian with at least one negative eigenvalue and consider the problem of finding a small-normed perturbation $E = E^*$ such that $A+E$ is positive semi-definite. Define, for any Hermitian B,

$$\delta(B) = \min\{\|E\|_2 : B+E \text{ is Hermitian positive semi-definite}\}. \tag{2.6}$$

From the Courant–Fischer minimax theory [16, p. 269], any admissible E in the definition of $\delta(A)$ must satisfy

$$0 \leqq \lambda_n(A+E) \leqq \lambda_n(A) + \lambda_1(E),$$

where $\lambda_n(\cdot) \leqq \cdots \leqq \lambda_1(\cdot)$. Thus

$$\|E\|_2 \geqq |\lambda_1(E)| \geqq \lambda_1(E) \geqq -\lambda_n(A). \tag{2.7}$$

We now find a perturbation E for which this lower bound is attained. Let A have the spectral decomposition

$$A = Z\Lambda Z^* = \sum_{i=1}^{n} \lambda_i z_i z_i^*, \qquad Z^*Z = I. \tag{2.8}$$

For

$$E_p = -\sum_{i:\,\lambda_i<0} \lambda_i z_i z_i^*$$

(or $E = -\lambda_n I$) it is easily seen that $A+E_p$ is singular and Hermitian positive semi-definite, with $\|E_p\|_2 = -\lambda_n$. It follows from (2.6) and (2.7) that

$$\delta(A) = -\lambda_n(A) \quad (\lambda_n(A) < 0). \tag{2.9}$$

Now observe, from (2.8), that A has the polar decomposition $A = UH$, where

$$U = Z \operatorname{diag}(\operatorname{sign}(\lambda_i))Z^*, \qquad H = Z \operatorname{diag}(|\lambda_i|)Z^*. \tag{2.10}$$

It follows that

$$E_p = \tfrac{1}{2}(H - A). \tag{2.11}$$

Thus $\frac{1}{2}(A+H) = A + E_p$ is a nearest Hermitian positive semi-definite matrix to A in the 2-norm. This is a special case of a result obtained by Halmos [18] that applies to general non-Hermitian linear operators A.

We summarise our findings in the following lemma.

LEMMA 2.4. *Let $A \in \mathbb{C}^{n\times n}$ be Hermitian, with the polar decomposition $A = UH$. Then*

(i) $\delta(A) = \max\{0, -\lambda_n(A)\} = \frac{1}{2}\|A - H\|_2$.

(ii) $\frac{1}{2}(A+H)$ *is a best Hermitian positive semi-definite approximation to A in the 2-norm.*

(iii) *For any Hermitian positive (semi-) definite $X \in \mathbb{C}^{n\times n}$,*

$$\|A - H\|_2 \leqq 2\|A - X\|_2.$$

(iv) *H and A have a common set of eigenvectors.*

The lemma shows that from the polar decomposition of a Hermitian matrix A we can obtain not only a best Hermitian positive semi-definite approximation to A, $\frac{1}{2}(A+H)$, but also, if A is nonsingular, a good Hermitian positive definite approximation to A, H itself. In § 6.3 we give an example of how the positive definite approximation may be utilised.

2.4. Perturbation bounds for the polar factors. It is of interest both for theoretical and for practical purposes (see § 4) to determine bounds for the changes induced in the polar factors of a matrix by perturbations in the matrix. The following theorem provides such bounds.

THEOREM 2.5. *Let $A \in \mathbb{C}^{n\times n}$ be nonsingular, with the polar decomposition $A = UH$. If $\varepsilon = \|\Delta A\|_F / \|A\|_F$ satisfies $\kappa_F(A)\varepsilon < 1$ then $A + \Delta A$ has the polar decomposition*

$$A + \Delta A = (U + \Delta U)(H + \Delta H),$$

where

$$\frac{\|\Delta H\|_F}{\|H\|_F} \leqq \sqrt{2}\,\varepsilon + O(\varepsilon^2),$$

$$\frac{\|\Delta U\|_F}{\|U\|_F} \leqq (1+\sqrt{2})\kappa_F(A)\varepsilon + O(\varepsilon^2).$$

Proof. Let $E = (1/\varepsilon)\Delta A$. Then $A + tE$ is nonsingular for $0 \leqq t \leqq \varepsilon$. Thus $A + tE$ has the polar decomposition

$$A + tE = U(t)H(t), \qquad 0 \leqq t \leqq \varepsilon, \tag{2.12}$$

where $H(t)$ is positive definite. We prove the theorem under the assumption that $U(t)$ and $H(t)$ are twice continuously differentiable functions of t; a rather similar but longer proof which does not require this assumption may be found in [22].

From (2.12),

$$H(t)^2 = (A + tE)^*(A + tE),$$

which gives, on differentiating [16, p. 4] and setting $t = 0$,

$$H\dot{H}(0) + \dot{H}(0)H = A^*E + E^*A.$$

Since $A = UH$, this can be written as

$$H\dot{H}(0) + \dot{H}(0)H = HF + F^*H, \tag{2.13}$$

where

$$F = U^*E.$$

Let H have the spectral decomposition

$$H = Z\Lambda Z^*, \qquad Z^*Z = I.$$

Performing a similarity transformation on (2.13) using Z gives

$$\Lambda\tilde{\dot{H}} + \tilde{\dot{H}}\Lambda = \Lambda\tilde{F} + \tilde{F}^*\Lambda,$$

where

$$\tilde{\dot{H}} = Z^*\dot{H}(0)Z = (\tilde{h}_{ij}), \qquad \tilde{F} = Z^*FZ = (\tilde{f}_{ij}).$$

This equation has the solution

$$\tilde{h}_{ij} = \frac{\lambda_i\tilde{f}_{ij} + \tilde{f}_{ji}^*\lambda_j}{\lambda_i + \lambda_j}, \qquad 1 \leqq i, j \leqq n.$$

Using the Cauchy-Schwarz inequality,

$$|\tilde{h}_{ij}|^2 \leqq \frac{\lambda_i^2 + \lambda_j^2}{(\lambda_i + \lambda_j)^2}(|\tilde{f}_{ij}|^2 + |\tilde{f}_{ji}|^2) \leqq |\tilde{f}_{ij}|^2 + |\tilde{f}_{ji}|^2,$$

from which it follows that

$$\|\tilde{\dot{H}}\|_F \leqq \sqrt{2}\|\tilde{F}\|_F.$$

Thus

$$\|\dot{H}(0)\|_F = \|\tilde{\dot{H}}\|_F \leqq \sqrt{2}\|\tilde{F}\|_F = \sqrt{2}\|F\|_F = \sqrt{2}\|E\|_F. \tag{2.14}$$

A Taylor expansion gives

$$H + \Delta H \equiv H(\varepsilon) = H(0) + \varepsilon\dot{H}(0) + O(\varepsilon^2) = H + \varepsilon\dot{H}(0) + O(\varepsilon^2),$$

so that

$$\|\Delta H\|_F \leqq \varepsilon\|\dot{H}(0)\|_F + O(\varepsilon^2) \leqq \sqrt{2}\,\varepsilon\|E\|_F + O(\varepsilon^2).$$

The required bound is obtained by dividing throughout by $\|H\|_F = \|A\|_F$ and using $\|E\|_F = \|A\|_F$.

Now write (2.12) in the form $U(t) = (A + tE)H(t)^{-1}$ and differentiate, to obtain

$$\dot{U}(t) = EH(t)^{-1} - (A + tE)H(t)^{-1}\dot{H}(t)H(t)^{-1}.$$

Setting $t = 0$ gives

$$\dot{U}(0) = EH^{-1} - AH^{-1}\dot{H}(0)H^{-1} = (E - U\dot{H}(0))H^{-1},$$

and so, using (2.14),

$$\|\dot{U}(0)\|_F \leqq (1+\sqrt{2})\|E\|_F\|H^{-1}\|_F = (1+\sqrt{2})\|E\|_F\|A^{-1}\|_F.$$

From the Taylor series for $U(t)$,

$$\begin{aligned}\|\Delta U\|_F = \|U(\varepsilon) - U(0)\|_F &\leqq \varepsilon\|\dot{U}(0)\|_F + O(\varepsilon^2)\\ &\leqq (1+\sqrt{2})\varepsilon\|E\|_F\|A^{-1}\|_F + O(\varepsilon^2)\\ &= (1+\sqrt{2})\varepsilon\kappa_F(A) + O(\varepsilon^2),\end{aligned}$$

which gives the required bound, since $\|\Delta U\|_F/\|U\|_F = \|\Delta U\|_F/\sqrt{n} \leqq \|\Delta U\|_F$. □

3. Computing the polar decomposition.

3.1. Using the singular value decomposition. Our constructive derivation of the polar decomposition in § 2 suggests the following computational procedure:

(1) compute the singular value decomposition (2.1), forming only the first n columns P_1 of P;

(2) form U and H according to (2.2) and (2.3).

This method requires (when A is real) approximately $7mn^2+11/3n^3$ flops to compute P_1, Σ and Q, if we use the Golub–Reinsch SVD algorithm [16, p. 175], plus mn^2 flops to form U and $n^3/2$ flops to form H (see [16, p. 32] for a discussion of the term "flop"). Since the SVD algorithm is numerically stable and is readily available in library routines such as LINPACK [11] this SVD approach has much to recommend it.

We now develop an alternative method for computing the polar decomposition which does not require the use of sophisticated library routines and which, in certain circumstances (see § 6.2), is computationally much less expensive than the SVD technique. The method applies to nonsingular square matrices. If $A\in\mathbb{C}^{m\times n}$ with $m>n$ and rank $(A)=n$ then we can first compute a QR factorisation [16, p. 146] $A=QR$ (where $Q\in\mathbb{C}^{m\times n}$ has orthonormal columns and R is upper triangular and nonsingular) and then apply the method to R. The polar decomposition of A is given in terms of that of R by

$$A=QR=Q(U_RH_R)=(QU_R)H_R\equiv UH.$$

3.2. A Newton method. Consider the iteration (the real matrix version of which is discussed in [3], [4], [5], [6], [28])

$$X_0=A\in\mathbb{C}^{n\times n},\qquad \text{nonsingular,} \tag{3.1a}$$

$$X_{k+1}=\tfrac{1}{2}(X_k+X_k^{-*}),\qquad k=0,1,2,\cdots, \tag{3.1b}$$

where X_k^{-*} denotes $(X_k^{-1})^*$. We claim that the sequence $\{X_k\}$ converges quadratically to the unitary polar factor in A's polar decomposition. To prove this we make use of the singular value decomposition

$$\begin{aligned} A&=P\Sigma Q^* \qquad (P^*P=Q^*Q=I_n)\\ &\equiv UH, \end{aligned}$$

where

$$U=PQ^*,\qquad H=Q\Sigma Q^*. \tag{3.2}$$

Define

$$D_k=P^*X_kQ. \tag{3.3}$$

Then from (3.1) we obtain

$$D_0=\Sigma, \tag{3.4a}$$

$$D_{k+1}=\tfrac{1}{2}(D_k+D_k^{-*}). \tag{3.4b}$$

Since $D_0\in\mathbb{R}^{n\times n}$ is diagonal with positive diagonal elements it follows by induction that the sequence $\{D_k\}$ is defined and that

$$D_k=\operatorname{diag}(d_i^{(k)})\in\mathbb{R}^{n\times n},\qquad d_i^{(k)}>0. \tag{3.5}$$

Accordingly, (3.4) represents n uncoupled scalar iterations

$$\begin{aligned} &d_i^{(0)}=\sigma_i,\\ &d_i^{(k+1)}=\frac{1}{2}\left(d_i^{(k)}+\frac{1}{d_i^{(k)}}\right), \end{aligned}\qquad 1\leqq i\leqq n,$$

which we recognise as Newton iterations for the square root of 1 with starting values the singular values of A.

Simple manipulations yield the relations (cf. [19, p. 84], [21])

$$d_i^{(k+1)} - 1 = \frac{1}{2d_i^{(k)}}(d_i^{(k)} - 1)^2, \qquad 1 \leqq i \leqq n, \tag{3.6}$$

$$\frac{d_i^{(k+1)} - 1}{d_i^{(k+1)} + 1} = \left(\frac{d_i^{(k)} - 1}{d_i^{(k)} + 1}\right)^2 = \cdots = \left(\frac{\sigma_i - 1}{\sigma_i + 1}\right)^{2^{k+1}} \equiv \eta_i^{2^{k+1}}, \qquad 1 \leqq i \leqq n. \tag{3.7}$$

Since A is nonsingular $|\eta_i| < 1$ for each i. It follows that $d_i^{(k)} - 1$ as $k \to \infty$ for each i, that is, $D_k \to I$, or equivalently, from (3.3) and (3.2)

$$\lim_{k\to\infty} X_k = U.$$

To analyse the rate of convergence we write (3.6) in the form

$$D_{k+1} - I = \tfrac{1}{2}(D_k - I)D_k^{-1}(D_k - I)$$

and pre- and post-multiply by P and Q^*, respectively, to obtain, from (3.2) and (3.3)

$$X_{k+1} - U = \tfrac{1}{2}(X_k - U)X_k^{-1}(X_k - U).$$

Furthermore, using (3.2), (3.3) and (3.7),

$$\begin{aligned} \|(X_{k+1} + U)^{-1}(X_{k+1} - U)\|_2 &= \|Q(D_{k+1} + I)^{-1}P^*P(D_{k+1} - I)Q^*\|_2 \\ &= \|(D_{k+1} + I)^{-1}(D_{k+1} - I)\|_2 \\ &= \max_{1\leqq i\leqq n}\left(\frac{d_i^{(k)} - 1}{d_i^{(k)} + 1}\right)^2 = \max_{1\leqq i\leqq n} \eta_i^{2^{k+1}}. \end{aligned}$$

Note from (3.3) and (3.5) that $d_1^{(k)}, \cdots, d_n^{(k)}$ are the singular values of X_k. We have proved the following.

THEOREM 3.1. *Let $A \in \mathbb{C}^{n\times n}$ be nonsingular and consider iteration* (3.1). *Each iterate X_k is nonsingular,*

$$\lim_{k\to\infty} X_k = U$$

where U is the unitary factor in the polar decomposition of A, and

$$\|X_{k+1} - U\|_2 \leqq \begin{cases} \dfrac{1}{2}\|X_k^{-1}\|_2\|X_k - U\|_2^2, & (3.8) \\[2ex] \|X_{k+1} + U\|_2\left(\max\limits_{1\leqq i\leqq n}\left|\dfrac{\sigma_i(X_k) - 1}{\sigma_i(X_k) + 1}\right|\right)^2, & (3.9) \\[2ex] \|X_{k+1} + U\|_2\left(\max\limits_{1\leqq i\leqq n}\left|\dfrac{\sigma_i(A) - 1}{\sigma_i(A) + 1}\right|\right)^{2^{k+1}}. & (3.10) \end{cases}$$

3.3. Accelerating convergence. The quadratic convergence of iteration (3.1) ensures rapid convergence in the final stages of the iteration: (3.8) implies that the number of correct significant figures will approximately be doubled on each step. Initially, however, the speed of convergence can be inordinately slow, as can be seen by considering $A = \alpha I$, for large $|\alpha|$, in (3.1) and (3.10).

We are led to the idea of scaling the matrix A, or more generally, scaling the current iterate at the start of each step, with the aim of hastening the onset of the ultimate phase of rapid convergence.

Consider the scaling $X_k \to \gamma_k X_k$, $\gamma_k > 0$. From (3.1b) we have

$$X_{k+1} = X_{k+1}(\gamma_k) = \frac{1}{2}\left(\gamma_k X_k + \frac{1}{\gamma_k} X_k^{-*}\right)$$

(thus γ_k can be regarded as an acceleration parameter), and from (3.9)

$$\|X_{k+1}(\gamma_k) - U\|_2 \leqq \|X_{k+1}(\gamma_k) + U\|_2 \theta_k(\gamma_k)^2, \tag{3.11}$$

where

$$\theta_k(\gamma_k) = \max_{1 \leqq i \leqq n} \left| \frac{\gamma_k \sigma_i(X_k) - 1}{\gamma_k \sigma_i(X_k) + 1} \right|.$$

A natural choice for γ_k is the value $\gamma_{\text{opt}}^{(k)}$ which minimises $\theta_k(\gamma)$. A straightforward argument shows that

$$\gamma_{\text{opt}}^{(k)} = (\sigma_1(X_k)\sigma_n(X_k))^{-1/2}, \tag{3.12}$$

$$\theta_k(\gamma_{\text{opt}}^{(k)}) = \frac{\kappa_2(X_k)^{1/2} - 1}{\kappa_2(X_k)^{1/2} + 1}. \tag{3.13}$$

One can show that for $X_{k+1} = X_{k+1}(\gamma_{\text{opt}}^{(k)})$,

$$\begin{aligned} \kappa_2(X_{k+1}) &\leqq \frac{1}{2}\left(\kappa_2(X_k)^{1/2} + \frac{1}{\kappa_2(X_k)^{1/2}}\right) \\ &\leqq \kappa_2(X_k)^{1/2}. \end{aligned} \tag{3.14}$$

If this acceleration technique is used at each stage of iteration (3.1) then from (3.11), (3.13) and (3.14) we have, by induction (cf. (3.10))

$$\|X_{k+1} - U\|_2 \leqq \|X_{k+1} + U\|_2 \left(\frac{\kappa_2(A)^{1/2^{k+1}} - 1}{\kappa_2(A)^{1/2^{k+1}} + 1}\right)^2. \tag{3.15}$$

The effectiveness of the acceleration procedure is illustrated by the example $A = \text{diag}\,(1, 2^4, 3^4, \cdots, 25^4)$; with the convergence criterion $\|X_k - U\|_2 \leqq 10^{-9}$ the unaccelerated iteration requires twenty-two iterations, while the accelerated version requires only seven.

3.4. The practical algorithm. It is not feasible to compute $\gamma_{\text{opt}}^{(k)}$ exactly at each stage, since this would require computation of the extremal singular values of X_k, but a good approximation to $\gamma_{\text{opt}}^{(k)}$ can be computed at negligible cost.

Taking $A = X_k$, X_k^{-1} in the inequalities [16, p. 15]

$$\sigma_1(A) = \|A\|_2 \leqq \sqrt{\|A\|_1 \|A\|_\infty} \leqq \sqrt{n}\,\|A\|_2,$$

and defining

$$\alpha_k = \sqrt{\|X_k\|_1 \|X_k\|_\infty}, \qquad \beta_k = \sqrt{\|X_k^{-1}\|_1 \|X_k^{-1}\|_\infty},$$

$$\gamma_{\text{est}}^{(k)} = \sqrt{\frac{\beta_k}{\alpha_k}},$$

we find, from (3.12), that

$$\frac{1}{n^{1/4}} \gamma_{\text{opt}}^{(k)} \leqq \gamma_{\text{est}}^{(k)} \leqq n^{1/4} \gamma_{\text{opt}}^{(k)}.$$

Making suitable modifications to the derivation of (3.15) one can show that if the acceleration parameter estimates $\gamma_{\text{est}}^{(k)}$ are used in the first k stages of iteration (3.1) then (cf. (3.15))

$$\|X_{k+1}-U\|_2 \leq \|X_{k+1}+U\|_2\left(\frac{\sqrt{n}\,k_2(A)^{1/2^{k+1}}-1}{\sqrt{n}\,k_2(A)^{1/2^{k+1}}+1}\right)^2. \tag{3.16}$$

This bound suggests that in the initial stages of iteration (3.1) the estimates $\gamma_{\text{est}}^{(k)}$ will be almost as effective as the exact values $\gamma_{\text{opt}}^{(k)}$.

We have found empirically that once the error $\|X_k - U\|_2$ is sufficiently small—less than 10^{-2}, say—it is advantageous to revert to the original, unaccelerated form of iteration (3.1) so as to secure the desirable quadratic convergence.

Incorporating the acceleration parameter estimates $\gamma_{\text{est}}^{(k)}$ into iteration (3.1) we have the following.

ALGORITHM POLAR. Given a nonsingular matrix $A \in \mathbb{C}^{n\times n}$ this algorithm computes the polar decomposition $A = UH$.

(1) $X_0 := A$; $k := -1$.

(2) Repeat

$k := k+1$

$Y_k := X_k^{-1}$

If "close to convergence" then

$\gamma_k := 1$

else

$\alpha_k := \sqrt{\|X_k\|_1 \|X_k\|_\infty}$; $\quad \beta_k := \sqrt{\|Y_k\|_1 \|Y_k\|_\infty}$

$\gamma_k := \sqrt{\beta_k/\alpha_k}$

$$X_{k+1} := \frac{1}{2}\left(\gamma_k X_k + \frac{1}{\gamma_k} Y_k^*\right)$$

Until converged.

(3) $U := X_{k+1}$

$H_1 := U^*A$

$H := \frac{1}{2}(H_1 + H_1^*)$ (to ensure that the computed H is Hermitian).

Cost: (for real A) $(s+1)n^3$ flops, where s iterations are required for convergence.

In step (3) of the algorithm we could implicitly force H to be Hermitian by computing only the upper triangular part of U^*A; the given technique is preferred for reasons discussed in § 4.

A suitable convergence test to apply in step (2) of Algorithm Polar is

$$\|X_{k+1} - X_k\|_1 \leq \delta_n \|X_k\|_1, \tag{3.17}$$

where δ_n, depending on n, is a small multiple of the machine unit roundoff u [16, p. 33].

4. Backward error analysis. Consider the SVD approach to computing the polar decomposition, described in § 3.1. Using the backward error analysis for the Golub–Reinsch SVD algorithm [16, p. 174] one can show that the computed polar factors of A, $\hat{U}$ and $\hat{H}$, satisfy

$$\begin{aligned}
&\hat{U} = V + \Delta U, && \|\Delta U\|_2 \leq \varepsilon,\\
&\hat{H} = K + \Delta H, && \hat{H}^* = \hat{H}, \quad \|\Delta H\|_2 \leq \varepsilon \|K\|_2,\\
&VK = A + \Delta A, && \|\Delta A\|_2 \leq \varepsilon \|A\|_2,
\end{aligned}$$

where V is unitary, K is Hermitian positive semi-definite (certainly positive definite

if $\kappa_2(A) < 1/\varepsilon$) and ε is a small multiple of the machine precision u. Thus $\hat{U}$ and $\hat{H}$ are relatively close to the true polar factors of a matrix "near" to A. This result is the best that can be expected of any method for computing the polar decomposition in finite precision arithmetic.

We have been unable to prove a corresponding stability result for Algorithm Polar. Instead we derive an a posteriori test for stability of the computed polar factors $\hat{U}$ and $\hat{H}$.

Under mild assumptions one can show that with the convergence test (3.17) $\hat{U}$ satisfies

$$\hat{U} = V + \Delta U, \quad V^*V = I, \quad \|\Delta U\|_2 \leqq \delta_n + O(\delta_n^2).$$

Algorithm Polar computes

$$\hat{H}_1 = \hat{U}^*A, \qquad \hat{H} = \tfrac{1}{2}(\hat{H}_1 + \hat{H}_1^*),$$

where, for simplicity, we ignore the rounding errors incurred in the computation of $\hat{H}_1$ and $\hat{H}$ (these lead to extra terms of order $\varepsilon\|A\|_2$, which do not affect the conclusion below). Defining

$$G = \tfrac{1}{2}(\hat{H}_1 - \hat{H}_1^*)$$

we have

$$V\hat{H} = V(\hat{H}_1 - G) = V(V^* + \Delta U^*)A - VG = A + \Delta A,$$

where

$$\|\Delta A\|_2 \leqq \delta_n\|A\|_2 + \|G\|_2 + O(\delta_n^2).$$

This result is comparable with the result for the SVD method if (changing to the one-norm)

(4.1a) $$\delta_n \approx \varepsilon,$$

(4.1b) $$\|G\|_1 \approx \delta_n\|A\|_1,$$

(4.1c) $$\hat{H} \text{ is positive definite.}$$

Thus, in particular, $\|G\|_1$ must be sufficiently small, that is, $\hat{H}_1$ must be sufficiently close to being Hermitian. These conditions are easily tested; one can test (4.1c) by attempting to compute a Choleski decomposition of $\hat{H}$. Note that evaluation of (4.1b) is computationally much less expensive than the alternative of comparing $\|A - \hat{U}\hat{H}\|_1$ with $\delta_n\|A\|_1$.

Once the above tests have been performed, the accuracy of the computed polar factors (that is, the forward error) can be estimated with the aid of Theorem 2.5. The condition numbers $\kappa_1(A)$, $\kappa_\infty(A)$ can be formed at no extra cost during the first step of Algorithm Polar.

5. Relation to matrix sign and square root iterations. In this section we show how iteration (3.1) is related to iterations for the matrix sign function and the matrix square root.

For a diagonalisable matrix $A = ZDZ^{-1}$, $D = \text{diag}\,(d_i)$, $\text{Re}\,d_i \neq 0$, the sign function is given by [10], [30]

$$\text{sign}\,(A) = Z\,\text{diag}\,(\text{sign}\,(\text{Re}\,d_i))Z^{-1}.$$

An iterative method for computing sign (A) is [10], [30]

(5.1) $$S_{k+1} = \tfrac{1}{2}(S_k + S_k^{-1}), \qquad S_0 = A.$$

This iteration is essentially Newton's method for a square root of I, with starting matrix A (see [21]). We observe that iteration (3.1) implicitly performs this "sign iteration" on the matrix Σ of singular values: see (3.4) and (3.5). In fact, iteration (3.1) may be derived by applying the sign iteration to the Hermitian matrix

$$W = \begin{pmatrix} 0 & A^* \\ A & 0 \end{pmatrix},$$

whose eigenvalues are plus and minus the singular values of A.

Our analysis of the convergence of iteration (3.1), and of the acceleration parameters $\{\gamma_k\}$, applies with suitable modifications to the sign iteration (5.1); cf. [23], [24], [25], [30].

Consider now the iteration

$$Y_{k+1} = \tfrac{1}{2}(Y_k + Y_k^{-1}B), \qquad Y_0 = B, \tag{5.2}$$

for a square root of $B \in \mathbb{C}^{n\times n}$. In [21] this iteration is shown to be numerically unstable in the sense that a small perturbation in the kth iterate can lead to perturbations in succeeding iterates which grow unboundedly.

It can be shown that the sequence $\{X_k\}$ from iteration (3.1) is related to the sequence $\{Y_k\}$ generated by (5.2) with $B = A^*A$ according to

$$X_k^* A \equiv Y_k.$$

Thus iteration (3.1) implicitly carries out iteration (5.2) on $B = A^*A$, without ever forming A^*A. The techniques of [21] can be used to show that iteration (3.1) does not suffer from the numerical instability which impairs iteration (5.2).

6. Applications.

6.1. Factor analysis [17], [31]. In psychometrics the "Orthogonal Procrustes" problem consists of finding an orthogonal matrix $Q \in \mathbb{R}^{n\times n}$ which most nearly transforms a given matrix $B \in \mathbb{R}^{m\times n}$ into a given matrix $A \in \mathbb{R}^{m\times n}$, according to the criterion that the sum of squares of the residual matrix $A - BQ$ is minimised [17], [31] (see also [9], [32]). Theorem 2.2 shows that a solution to this problem is $Q = U$ where $B^TA = UH$ is a polar decomposition. If A and B have full rank then B^TA is nonsingular and U may be computed by Algorithm Polar; if either A or B is rank-deficient then U may be computed via a singular value decomposition of B^TA, as described in § 3.1 (see also [16, p. 426]).

6.2. Aerospace computations [2]–[7], [9], [28], [32]. In aerospace systems an important role is played by the direction cosine matrix (DCM)—an orthogonal matrix $D \in \mathbb{R}^{3\times 3}$ which transforms vectors from one coordinate system to another. Errors incurred in computation of the DCM result in a loss of orthogonality; an intuitively appealing way in which to restore orthogonality is to replace the computed DCM $\hat{D}$ by the nearest orthogonal matrix, that is, by the orthogonal polar factor of $\hat{D}$ (see Corollary 2.3).

A key feature of this application is that $\hat{D}$ is relatively close to being orthogonal: typically $\|\hat{D} - U\|_F < .1$ [2], [3], [4]. From (3.8) we can expect iteration (3.1) to converge within four iterations, for a tolerance $\delta_3 \geqq 10^{-16}$ in (3.17). Of course if U is not required to full machine accuracy then there is no need to iterate to convergence—just one or two iterations may yield a sufficiently accurate approximation to U.

For matrices that are as close to orthogonality as $\hat{D}$ above, computation of U from Algorithm Polar will require at most $4n^3$ flops, making this method particularly

attractive, since the singular value decomposition approach described in § 3.1 still requires approximately $12n^3$ flops.

We now compare Algorithm Polar with two other iterative techniques which have been proposed for computing the orthogonal polar factor of a nearly-orthogonal matrix.

Bjorck and Bowie [7] derive a family of iterative methods with orders of convergence $2, 3, \cdots$ by employing a binomial expansion for the matrix square root in the expression $U = AH^{-1} = A(A^*A)^{-1/2}$ (see Lemma 2.1(i)). Their quadratically convergent method is

$$X_0 = A, \tag{6.1a}$$

$$Q_k = I - X_k^* X_k, \tag{6.1b}$$

$$X_{k+1} = X_k(I + \tfrac{1}{2}Q_k), \tag{6.1c}$$

$$k = 0, 1, 2, \cdots.$$

One step of this iteration costs $3n^3/2$ flops (for $A \in \mathbb{R}^{n\times n}$); in comparison iteration (3.1) requires only n^3 flops per step. Also, while iteration (3.1) converges for any nonsingular A, a practical condition for the convergence of iteration (6.1) is [7]

$$0 < \sigma_i(A) < \sqrt{3}, \qquad 1 \leqq i \leqq n.$$

The following iteration is proposed in [2]:

$$X_0 = A \in \mathbb{R}^{n\times n}, \tag{6.2a}$$

$$X_{k+1} = X_k - \tfrac{1}{2}(X_k A^T X_k - A), \qquad k = 0, 1, 2, \cdots. \tag{6.2b}$$

It is shown in [3], [5], [28] that iteration (6.2) is locally, linearly convergent to the orthogonal polar factor of A. Evaluation of iteration (6.2) requires $2n^3$ flops per step. Because of its linear convergence and its computational cost, this iteration is decidedly unattractive in comparison with iteration (3.1).

We appreciate that flop counts are not necessarily a fair means for comparing competing algorithms. In fact, because iterations (3.1), (6.1) and (6.2) use only matrix multiplications, inversions and additions, the flops involved in these iterations may be significantly cheaper than those in the Golub-Reinsch SVD algorithm, particularly on the new special computer architectures.

6.3. Optimisation. Newton's method for the minimisation of $F(x)$, $F: \mathbb{R}^n \to \mathbb{R}$, requires at each stage computation of a search direction p_k from

$$G_k p_k = -g_k,$$

where $g_k = \nabla F(x_k)$ is the gradient vector and

$$G_k = \left(\frac{\partial^2 F}{\partial x_i\, \partial x_j}(x_k)\right)$$

is the (symmetric) Hessian matrix. Difficulties occur when G_k is not positive definite since p_k, if defined, need not be a descent direction [14, p. 107]. We suggest that in this situation one replaces G_k by its polar factor H. H is positive definite (assuming G_k is nonsingular) and it has the properties listed in Lemmas 2.1 and 2.4. H may be computed using Algorithm Polar at a cost of $(s/2+1)n^3$ flops, if advantage is taken of the symmetry of the iterates (for example the LINPACK routine SSIDI [11] may be used to compute the matrix inverses). The equation $Hp_k = -g_k$ may be solved in $n^3/6$ flops by use of the Choleski decomposition.

In [14] several techniques are described for modifying G_k to give a related positive definite matrix. One of these consists of computing a spectral decomposition

$G_k = Z\Lambda Z^*$ and replacing G_k by $\hat{G}_k = Z|\Lambda|Z^*$; from (2.10) we recognize $\hat{G}_k$ as the polar factor H of G_k. This approach yields the same matrix as our suggestion, at a cost of about $6n^3$ flops [16, p. 282].

6.4. Matrix square root [8], [10], [20], [21], [25]. A new method for computing the symmetric positive definite square root $A^{1/2}$ of a symmetric positive definite matrix A is obtained from the observation that if

$$A = LL^T, \qquad L^T = UH$$

are Choleski and polar decompositions respectively, then (see Lemma 2.1(i)) $H = A^{1/2}$.

ALGORITHM ROOT. Given a symmetric positive definite matrix $A \in \mathbb{R}^{n\times n}$ this algorithm computes $A^{1/2}$.

(1) Compute the Choleski decomposition $A = LL^T$ [16, p. 89].

(2) Compute the Hermitian polar factor $H = A^{1/2}$ of L^T using Algorithm Polar.

Cost: $(s - \frac{1}{6})n^3$ flops, where s iterations of Algorithm Polar are required for convergence (taking into account the triangularity of L).

Note that since we are applying Algorithm Polar to L^T, the quantity $\kappa_2(A)$ in the bound (3.16) is replaced by $\kappa_2(L^T) = \kappa_2(A)^{1/2}$.

Algorithm Root is an attractive, numerically stable alternative (see § 5) to the iterations in [10], [21], [25] for the case where A is symmetric positive definite.

7. Numerical examples. In this section we present some test results which illustrate the performance of Algorithm Polar. The computations were performed using MATLAB [29] in double precision on a VAX 11/780 computer; the unit roundoff $u = 2^{-56} \approx 1.39 \times 10^{-17}$.

We used the convergence test (3.17) with $\delta_n = 4u$ for $n \leqq 25$ and $\delta_{50} = 8u$. Once the criterion $\|X_k - X_{k-1}\|_1 \leqq .01$ was satisfied $X_{k+1}, X_{k+2}, \cdots$ were computed using the unaccelerated iteration ($\gamma_j = 1, j > k$).

In the first test real matrices A of order $n = 5, 10, 25, 50$ were generated according to $A = U\Sigma V^T$, where $\Sigma = \operatorname{diag}(\sigma_i)$ is a matrix of singular values ($\sigma_i = i$, i^2, i^4 or 2^i) and U, V are random orthogonal matrices (different for each A), obtained from the QR decomposition of a matrix with elements from the uniform distribution on $[0, 1]$. The results are summarised in Table 7.1. The quantity

$$\text{BERR}_n = \frac{\|H_1 - H_1^*\|_1}{2\delta_n \|A\|_1}$$

is the backward error measure derived in § 4 (see (4.1b)) and must be of order one for the algorithm to have performed in a stable manner. For every matrix in this test the computed Hermitian polar factor $\hat{H}$ was positive definite.

TABLE 7.1
Number of iterations.

	$n = 5$	10	25	50
$\sigma_i = i$	6	7	8	8
$\sigma_i = i^2$	7	7	10	9
$\sigma_i = i^4$	8	8	10	10
$\sigma_i = 2^i$	7	8	9	10
max BERR_n	.38	.55	2.1	2.8

The second test compares Algorithm Polar with iterations (6.1) and (6.2) (using the same convergence test, (3.17), for each iteration). The parametrised matrix

$$A(\alpha)=\begin{pmatrix} \alpha & 0 & -1 \\ 0 & 1 & 0 \\ -1 & 0 & 0 \end{pmatrix}$$

is orthogonal for $\alpha = 0$. The results are displayed in Table 7.2.

TABLE 7.2
Number of iterations.

α	Algorithm Polar	Iteration (6.1)	Iteration (6.2)
.001	4	4	5
.01	4	4	8
.1	5	5	13
1	6	10	76
2	7	diverged	diverged

8. Conclusions. From the test results of § 7 and the theory of § 3 we draw several conclusions about Algorithm Polar.

The acceleration parameter estimates are very effective. Convergence to a tolerance $\delta_n \leqq 10^{-17}$ (see (3.17)) is usually obtained within ten iterations, the computational cost of one iteration being approximately n^3 flops.

In applications where A is nearly orthogonal (see § 6.2) Algorithm Polar is an attractive alternative to iterations (6.1) and (6.2)—it is guaranteed to converge (within four or five iterations, typically) and it will usually be computationally the least expensive of the three methods.

We have not proved that Algorithm Polar is stable, that is, that the computed polar factors are relatively close to the true polar factors of a matrix near to A. The tests (4.1) provide an inexpensive means of monitoring the stability of Algorithm Polar. Algorithm Polar has performed stably in all our numerical tests, producing, in every case, computed polar factors which are just as acceptable as those furnished by the SVD approach.

Acknowledgments. I have benefited from stimulating discussions on this work with Professors A. Bjorck, R. Byers, G. H. Golub and C. B. Moler. This work was completed during a visit to Stanford University. I thank Professor G. H. Golub for making my stay so pleasant and Stanford University for the use of their computer facilities. I also thank Dr. I. Gladwell and Dr. G. Hall for their valuable comments on the manuscript. The helpful comments of the referee and editor are appreciated.

REFERENCES

[1] L. AUTONNE, *Sur les groupes linéaires, réels et orthogonaux*, Bull. Soc. Math. France, 30 (1902), pp. 121-134.

[2] I. Y. BAR-ITZHACK, *Iterative optimal orthogonalization of the strapdown matrix*, IEEE Trans. Aerospace Electron. Systems, 11 (1975), pp. 30-37.

[3] ———, *A unidimensional convergence test for matrix iterative processes applied to strapdown navigation*, Internat. J. Numer. Methods Engrg., 11 (1977), pp. 115-130.

[4] I. Y. BAR-ITZHACK AND K. A. FEGLEY, *Orthogonalization techniques of a direction cosine matrix*, IEEE Trans. Aerospace Electron. Systems, 5 (1969), pp. 798-804.

[5] I. Y. BAR-ITZHACK AND J. MEYER, *On the convergence of iterative orthogonalization processes*, IEEE Trans. Aerospace Electron. Systems, 12 (1976), pp. 146-151.
[6] I. Y. BAR-ITZHACK, J. MEYER AND P. A. FUHRMANN, *Strapdown matrix orthogonalization: the dual iterative algorithm*, IEEE Trans. Aerospace Electron. Systems, 12 (1976), pp. 32-37.
[7] A. BJORCK AND C. BOWIE, *An iterative algorithm for computing the best estimate of an orthogonal matrix*, SIAM J. Numer. Anal., 8 (1971), pp. 358-364.
[8] A. BJORCK AND S. HAMMARLING, *A Schur method for the square root of a matrix*, Linear Algebra Appl., 52/53 (1983), pp. 127-140.
[9] J. E. BROCK, *Optimal matrices describing linear systems*, AIAA J., 6 (1968), pp. 1292-1296.
[10] E. D. DENMAN AND A. N. BEAVERS, *The matrix sign function and computations in systems*, Appl. Math. Comput., 2 (1976), pp. 63-94.
[11] J. J. DONGARRA, J. R. BUNCH, C. B. MOLER AND G. W. STEWART, LINPACK *Users' Guide*, Society for Industrial and Applied Mathematics, Philadelphia, PA, 1979.
[12] K. FAN AND A. J. HOFFMAN, *Some metric inequalities in the space of matrices*, Proc. Amer. Math. Soc., 6 (1955), pp. 111-116.
[13] F. R. GANTMACHER, *The Theory of Matrices, Volume One*, Chelsea, New York, 1959.
[14] P. E. GILL, W. MURRAY AND M. H. WRIGHT, *Practical Optimization*, Academic Press, London, 1981.
[15] G. H. GOLUB, *Least squares, singular values and matrix approximations*, Appl. Math., 13 (1968), pp. 44-51.
[16] G. H. GOLUB AND C. F. VAN LOAN, *Matrix Computations*, Johns Hopkins University Press, Baltimore, MD, 1983.
[17] B. F. GREEN, *The orthogonal approximation of an oblique structure in factor analysis*, Psychometrika, 17 (1952), pp. 429-440.
[18] P. R. HALMOS, *Positive approximants of operators*, Indiana Univ. Math. J., 21 (1972), pp. 951-960.
[19] P. HENRICI, *Elements of Numerical Analysis*, John Wiley, New York, 1964.
[20] N. J. HIGHAM, *Computing real square roots of a real matrix*, Numerical Analysis Report No. 89, University of Manchester, England, 1984; to appear in Linear Algebra Appl.
[21] ———, *Newton's method for the matrix square root*, Numerical Analysis Report No. 91, University of Manchester, England, 1984; to appear in Math. Comp.
[22] ———, *Computing the polar decomposition—with applications*, Numerical Analysis Report No. 94, University of Manchester, England, 1984.
[23] W. D. HOSKINS, D. S. MEEK AND D. J. WALTON, *The numerical solution of the matrix equation* $XA+AY=F$, BIT, 17 (1977), pp. 184-190.
[24] ———, *The numerical solution of* $A'Q+QA=-C$, IEEE Trans. Automat. Control, AC-22 (1977), pp. 882-883.
[25] W. D. HOSKINS AND D. J. WALTON, *A faster, more stable method for computing the pth roots of positive definite matrices*, Linear Algebra Appl., 26 (1979), pp. 139-163.
[26] J. B. KELLER, *Closest unitary, orthogonal and Hermitian operators to a given operator*, Math. Mag., 48 (1975), pp. 192-197.
[27] M. MARCUS AND H. MINC, *Introduction to Linear Algebra*, Macmillan, New York, 1965.
[28] J. MEYER AND I. Y. BAR-ITZHACK, *Practical comparison of iterative matrix orthogonalisation algorithms*, IEEE Trans. Aerospace Electron. Systems, 13 (1977), pp. 230-235.
[29] C. B. MOLER, MATLAB *Users' Guide*, Technical Report CS81-1 (revised), Department of Computer Science, University of New Mexico, Albuquerque, NM, 1982.
[30] J. D. ROBERTS, *Linear model reduction and solution of the algebraic Riccati equation by use of the sign function*, Internat. J. Control, 32 (1980), pp. 677-687.
[31] P. H. SCHONEMANN, *A generalized solution of the orthogonal Procrustes problem*, Psychometrika, 31 (1966), pp. 1-10.
[32] G. WAHBA, *Problem* 65-1: *A least squares estimate of satellite attitude*, SIAM Rev., 7 (1965), p. 409; solutions in, 8 (1966), pp. 384-386.

SIAM J. SCI. STAT. COMPUT.
Vol. 7, No. 4, October 1986

008

CONDITION NUMBER ESTIMATORS IN A SPARSE MATRIX SOFTWARE*

ZAHARI ZLATEV†, JERZY WASNIEWSKI‡ AND KJELD SCHAUMBURG§

Abstract. The stability of the computational process in the solution of systems of linear algebraic equations $Ax = b$ depends on the condition number of matrix A. Reliable and efficient algorithms for calculating estimates of the condition number of a matrix are given in *An estimate for the condition number of a matrix*, SIAM J. Numer. Anal., 16 (1979), pp. 368–375. The application of these algorithms in a sparse matrix software, package Y12M [Y12M-*solution of large and sparse systems of linear algebraic equations*, Lecture Notes in Computer Science, Vol. 121, Springer, Berlin, 1981], is discussed. Three algorithms have been implemented in package Y12M and tested on a very large set of problems. The pivotal strategies for sparse matrices, which are used instead of the partial pivoting for dense matrices, usually provide reliable estimates for the condition number when the value of the stability factor u is small, say $u \in [4, 16]$ ($u = 1$ for the partial pivoting). It is shown that for the subroutines of package Y12M this is true even if the stability factor u is rather large. This phenomenon is explained by a careful analysis of the main pivotal strategy in package Y12M. Very often a special nonnegative parameter, a drop-tolerance, is used during the factorization of matrix A so that each element, which in the course of the calculations becomes smaller in absolute value than the drop-tolerance, is considered as a zero element. Both storage and computing time may be saved in this way for some classes of matrices. The influence of the use of a large drop-tolerance on the reliability of the condition number estimators is discussed. Some conclusions, concerning the three condition number estimators, are given.

Key words. condition number of a matrix, Gaussian elimination, sparse matrices, sparse matrix technique, pivotal strategies for sparse matrices, stability factor, drop-tolerance

1. Statement of the problem. Let $A \in \mathbb{R}^{n \times n}$ be a given matrix. Assume that the Gaussian elimination (GE) is used to obtain a factorization $LU = A + E$, where $L \in \mathbb{R}^{n \times n}$ is a unit lower triangular matrix, $U \in \mathbb{R}^{n \times n}$ is an upper triangular matrix and $E \in \mathbb{R}^{n \times n}$ is a perturbation matrix [15]–[19]. The GE is normally carried out by the use of some pivoting. Any pivoting can be considered as a multiplication of matrix A by permutation matrices. It is assumed that such a multiplication is performed before the beginning of the GE and the notation A is also used for the matrix obtained after the multiplication. It is clear that this is not a restriction.

Sometimes it is necessary to calculate an estimate κ^* of the condition number $\kappa(A) = \|A\| \|A^{-1}\|$ of matrix A. Here it is assumed that A^{-1} exists and that $\|\cdot\|$ is some norm of the matrix under consideration. Algorithms for calculating κ^* in a cheap and reliable way are discussed in [2]–[4], [8], [9], [11], [12]. The two algorithms described in [3] together with the algorithm from [9] will be used in this paper. These algorithms are based on the following rules: (i) *compute* $\|A\|$, (ii) *calculate L and U*, (iii) *solve* $U^T w = e$ *and* $L^T y = w$ (*e being a given vector*), (iv) *solve* $Lv = y$ *and* $Uz = v$, (v) *calculate* $A^* = \|z\|/\|y\|$ (*and consider* A^* *as an approximation to* $\|A^{-1}\|$) *and* (vi) *set* $\kappa^* = \|A\| A^*$.

The approximation A^* is good when $\|y\|/\|e\|$ is as large as possible [3, p. 371]. Therefore an attempt to choose the components of e so that $\|y\|/\|e\|$ is large has to be carried out.

* Received by the editor July 28, 1983, and in revised form August 1, 1985.

† Air Pollution Laboratory, National Agency of Environmental Protection, Risø National Laboratory, DK-4000 Roskilde, Denmark.

‡ Regional Computing Centre at the University of Copenhagen, Vermundsgade 5, DK-2100 Copenhagen, Denmark.

§ Department of Chemical Physics, The H. C. Ørsted Institute, Universitetsparken 5, DK-2100 Copenhagen, Denmark.

In the first algorithm from [3] (see also [8]) the components of e are determined as follows. Let e_1 be either 1 or -1. Consider

$$u_{ii}w_i = e_i - \sum_{j=1}^{i-1} u_{ji}w_j \qquad (i=2(1)n) \tag{1.1}$$

and set $e_i = 1$ if the sum in (1.1) is negative and $e_i = -1$ otherwise. This algorithm is implemented in subroutine DECOMP in [8]. This is probably the first subroutine in which a device for estimating $\kappa(A)$ is implemented.

The second algorithm in [3] is not so simple. Set again e_1 equal either to 1 or to -1. Consider the quantities:

$$t_i = \sum_{j=1}^{k-1} u_{ji}w_i \quad (i=k(1)n), \qquad e_k^+ = \operatorname{sign}(-t_k), \qquad e_k^- = -e_k^+ \quad (k=2(1)n); \tag{1.2}$$

$$w_k^+ = (e_k^+ - t_k)/u_{kk}, \quad w_k^- = (e_k^- - t_k)/u_{kk} \quad (k=2(1)n); \tag{1.3}$$

$$t_i^+ = t_i + u_{ki}w_k^+, \quad t_i^- = t_i + u_{ki}w_k^- \quad (i=k+1(1)n,\ k=2(1)n); \tag{1.4}$$

$$T_n^+ = \sum_{i=k}^{n} t_i^+, \quad T_k^- = \sum_{i=k}^{n} t_i^- \quad (k=2(1)n). \tag{1.5}$$

Set $e_k = e_k^+$ if $T_k^+ > T_k^-$ and $e_k = e_k^-$ otherwise. This algorithm is implemented in many subroutines of LINPACK [6]. An attempt to avoid overflows and divisions by zero is carried out in LINPACK by scaling the appropriate vectors during the calculations. The number of scalings could be reduced for some matrices by a simple modification of the LINPACK condition number estimator [9].

Three questions should be answered in connection with the implementation of the above algorithms in sparse matrix software:

(i) *How can the sparsity scheme used in the software under consideration be applied in the calculations within the condition number estimator*?

(ii) *What is the effect of the pivotal strategy used in the software on the accuracy of the condition number estimator*?

(iii) *What is the effect of neglecting some "small" elements in the* GE *process on the condition number estimator*?

The first question concerns the efficiency of the sparse condition number estimators with regard to the computing time and storage used. It should be stressed here that it *may* be very inefficient just to take a code for dense matrices and to modify it for sparse matrices. The fact that the matrix is sparse must be exploited and special techniques have to be applied.

The next two questions are related to the accuracy of κ^*. It should be emphasized here "*that at best we can obtain* $\kappa(A+E)$" [3, p. 373]. Therefore in the calculation of κ^* by the use of L and U an assumption that the GE is stable is made. If matrix A is dense, then it is commonly accepted that "*in practice Gaussian elimination with partial pivoting must be considered a stable algorithm*", [15, p. 152], though examples where this is not true can be constructed, [18]. Thus, in the condition number estimators it is implicitly assumed that L and U are accurate and that the norm of E is small in some sense. Experience indicates that this is a realistic assumption for dense matrices. However, the situation may change when the matrix is sparse, because the accuracy requirements are often relaxed in the choice of a pivotal strategy and/or because some small elements may be neglected during the GE (both actions being taken in order to preserve the sparsity of L and U). Thus some difficulties connected with the accuracy of the GE process for sparse matrices *may* arise and a careful examination of the

influence of the relaxation of the accuracy requirements during the factorization of sparse matrices on the accuracy of the condition number estimators seems to be useful.

The development of several condition number estimators for package Y12M will be described in this paper. Package Y12M is fully documented in [28]. The different algorithms and strategies implemented in this package are discussed in [20], [23], [24], [26], [30]. The sparse structure needed in the condition number subroutines will be outlined in § 2. The implementation of three algorithms for estimating the condition number of a matrix in package Y12M will be described in § 3. Some results concerning the dependence of the accuracy of the calculated condition numbers on the stability requirements in the pivotal strategy will be given in § 4. The influence of the introduction of a drop-tolerance (leading to neglecting some elements of A) on the accuracy of the calculated condition numbers will be discussed in § 5. Some concluding remarks will be made in § 6.

2. Application of the storage scheme used in package Y12M. Let ANORM1 $= \|A\|_1$. The main steps in the calculation of κ^* are the computation of ANORM1 and the solution of $U^T w = e$, $L^T y = w$, $Lv = y$, $Uz = v$. The application of the storage scheme used in package Y12M in the performance of these steps will be described in this section. The codes by which the calculations are carried out will also be given.

Let N and NZ be the order of A and the number of nonzero elements in A. Before the beginning of the GE the nonzero elements are stored in the first NZ locations of a REAL array $A(NN)$; $NN \geqq 2NZ$. The order of the nonzero elements is arbitrary but if $a_{ij} \neq 0$ is stored in $A(K)$, $1 \leqq K \leqq NZ$, then $RNR(K) = i$ and $SNR(K) = j$. $RNR(NN1)$ and $SNR(NN)$ are INTEGER arrays; $NN1 \geqq NZ$. The code for calculating ANORM1 by the use of this structure and by the use of a REAL array W, of length at least equal to N, is given in Fig. 1.

```
      ANORM1 = 0.0
      DO 10 I = 1, N
10    W(I) = 0.0
      DO 20 I = 1, NZ
20    W(SNR(I)) = W(SNR(I)) + ABS(A(I))
      DO 30 I = 1, N
30    ANORM1 = AMAX1(W(I), ANORM1)
```

FIG. 1. *Calculation of* ANORM1 *by the use of the storage scheme in package* Y12M.

At the end of the GE the nonzero elements of both L and U (with the exception of its diagonal elements) are stored in array A. An INTEGER array, $HA(N, 11)$, is used to store pointers for starts and ends of rows and columns, information about the pivotal interchanges etc. Only the first 3 columns of HA are of interest in this paper. The nonzero elements of row i in L are stored from location $HA(i, 1)$ to location $HA(i, 2) - 1$ in array A. The nonzero elements of row i in U (without u_{ii}) are between $HA(i, 2)$ and $HA(i, 3)$. There are no free locations between $HA(i, 1)$ and $HA(i, 3)$. If a nonzero element, either from L or from U, is stored in $A(K)$, then its column number is stored in $SNR(K)$; $1 \leqq K \leqq NN$. PIVOT(I) contains the pivotal element u_{ii}; PIVOT being a REAL array of length at least equal to N. The codes for solving the 4 systems involved in the calculation of κ^* are given in Figs. 2–5.

Let YNORM1 and ZNORM1 be the 1-norms of vectors y and z. The calculation of these norms is the same as in the dense case. Codes for calculating YNORM1 and

```
      W(1) = W(1)/PIVOT(1)
      DO 50 I = 2, N
      DO 40 J = HA(I-1,2), HA(I-1,3)
   40 W(SNR(J)) = W(SNR(J)) - W(I-1) * A(J)
   50 W(I) = W(I)/PIVOT(I)
```

FIG. 2. *Solving $U^T w = e$ using the storage scheme in package* Y12M. *It is assumed that e is stored in array W on entry. The solution vector w is in W on exit.*

```
      DO 60 I = N, 2, -1
      DO 60 J = HA(I,1), HA(I,2)-1
   60 W(SNR(J)) = W(SNR(J)) - W(I) * A(J)
```

FIG. 3. *Solving $L^T y = w$ using the storage scheme in package* Y12M. *It is assumed that w is stored in W on entry. The solution y is in W on exit.*

```
      DO 80 I = 2, N
      DO 80 J = HA(I,1), HA(I,2)-1
   80 W(I) = W(I) - A(J) * W(SNR(J))
```

FIG. 4. *Solving $Lv = y$ using the storage scheme in package* Y12M. *It is assumed that y is stored in W on entry. The solution v is in W on exit.*

```
      W(N) = W(N)/PIVOT(N)
      DO 100 I = N-1, 1, -1
      DO 90 J = HA(I,2), HA(I,3)
   90 W(I) = W(I) - A(J) * W(SNR(J))
  100 W(I) = W(I)/PIVOT(I)
```

FIG. 5. *Solving $Uz = v$ using the storage scheme in package* Y12M. *It is assumed that v is stored in W on entry. The solution z is in W on exit.*

ZNORM1 are given in Figs. 6 and 7. The code in Fig. 6 should be used after the solution of $L^T y = w$, while that in Fig. 7 should be used after the solution of $Uz = v$. LINPACK subroutines may be attached to the condition number estimators for sparse matrices and used instead of codes given in Figs. 6 and 7.

The pieces of codes in the above figures can be connected in an obvious way and the number ACOND1 = ANORM1 * ZNORM1/YNORM1 will often be a good estimate κ_1^* of the magnitude of $\kappa_1(A) = \|A\|_1 \|A^{-1}\|_1$ even if an arbitrary vector e is chosen in the beginning of the process [3, p. 371], [11, pp. 7-8]. The probability of achieving a good κ_1^* is enhanced when an attempt to choose e so that $\|y\|_1/\|e\|_1$ is as large as

```
      YNORM1 = 0.0
      DO 70 I = 1, N
   70 YNORM1 = YNORM1 + ABS(W(I))
```

FIG. 6. *Calculating the* 1*-norm of vector y.*

```
      ZNORM1 = 0.0
      DO 110 I = 1, N
  110 ZNORM1 = ZNORM1 + ABS(W(I))
```

FIG. 7. *Calculating the* 1*-norm of vector z.*

possible is made [3, p. 371], [11, p. 8]. The implementation of some algorithms with special choices of e will be discussed in § 3.

3. Algorithms for choosing a starting vector: implementation and comparison. The implementation of the algorithms from [3] and [9] in package Y12M is discussed below. In all these algorithms the starting vector e is chosen in a special way. The difficulties arise because the components of e must be determined dynamically in the process of the solution of system $U^T w = e$.

Let X be any subroutine for calculating κ_1^* of a *dense* matrix A. Then the following strategy seems to be very straightforward in the efforts to obtain a version of subroutine X for sparse matrices.

Strategy 1. 1). *Replace all loops involving the two-dimensional array, in which the elements of matrix A are stored, by loops involving one-dimensional arrays. The loops corresponding to the storage scheme in package* Y12M *are given in Figs.* 1–5.

2). *Leave the other parts of X unchanged. For example, loops like these given in Figs.* 6 *and* 7 *will not be changed according to this rule*; *the same is true for the loops carried out to scale vectors in* [6].

Strategy 1 has successfully been applied to obtain a version for sparse matrices from the algorithm in DECOMP, [8]. This algorithm is called Algorithm 1 below.

Strategy 1 has also been applied to obtain a version for sparse matrices from the algorithm used in SGECO [6]. The version of the algorithm in SGECO found by Strategy 1 is referred to as Algorithm 2*. This version is not very efficient. This is so because the sums (1.5) require too many, $2(n-k+1)$, arithmetic operations at stage k, $k=2(1)n$. For dense matrices the relations $t_i^+ \neq t_i$ and $t_i^- \neq t_i$ are normally satisfied $\forall i \in \{k+1, k+2, \cdots, n\}$ at stage k, $k=2(1)n$. Therefore the sums (1.5) *must* be calculated at each stage when A is dense. For A sparse, however, $t_i^+ = t_i$ and $t_i^- = t_i$ for many i, $i \in \{k+1, k+2, \cdots, n\}$, at any stage k, $k=2(1)n$, because many elements u_{ki} are equal to zero. Therefore the calculation of t_i^+ and t_i^- by (1.4) is not justified if A is sparse. A REAL variable SUM is introduced. SUM = 0.0 is set before the beginning of the solution of $U^T w = e$. At stage k, $k=2(1)n$, if $t_i^+ \neq t_i$, then SUM is updated by $\text{SUM} = \text{SUM} - ABS(t_i) + ABS(t_i^+)$. The question whether $t_i^+ \neq t_i$ is not asked because by the use of the pointers stored in the first three columns of array *HA* only the indices for which $t_i^+ \neq t_i$ are considered. The operations involving t_i^- are considered in a similar way. The version so found is called *Algorithm* 2. It must be emphasized here that *Algorithm* 2* *and Algorithm* 2 *are two different implementations, in a sparse matrix code* (*not necessarily package* Y12M), *of the* LINPACK *algorithm applied in* SGECO.

The modifications advocated in [9] can easily be inserted in Algorithm 2 (following the instructions given in [9, p. 385]. This has been done and the algorithm so found is called Algorithm 3. An attempt to reduce the number of scalings (performed in the LINPACK subroutines and, therefore, also in Algorithm 2 and in Algorithm 2*) is carried out in Algorithm 3.

The subroutines in which the above algorithms are applied were run on a wide range of test-matrices. The number of test-matrices was several thousands. The order of the matrices varies from 32 to 10,000. Both matrices that arise in practical problems and matrices that are produced by subroutines that generate sparse test-matrices, [20], [22]–[24], [28]–[31], were used in the experiments. Matrices arising in fluid dynamics [5], in thermodynamics [26], in nuclear spectroscopic theory [13], [14], [21], in different chemical processes [1] as well as the Harwell set of test-matrices [7] were used in the runs.

The matrix-generators used produce matrices depending on some or all of the following parameters: *the number of rows m, the number of columns n, the sparsity*

pattern modifier c, the modifier of the number NZ of nonzero elements r and the modifier of the magnitude of nonzero elements α. Two such generators, MATRE and MATRF2, were used. MATRE generates matrices of class $E(n, c)$, [20], that depend on n and c only. These matrices are symmetric and positive definite (containing 4's on the main diagonal, -1's on the two adjacent diagonals, -1's on the two diagonals located c locations away from the main diagonal; all other elements being equal to zero). MATRF2 generates matrices of class F2(m, n, c, r, α), [29], [30]. This generator will be used with $m = n$ only. If $m = n$, then the nonzero elements of a matrix of this class are: $a_{ii} = 1$ $(i = 1(1)n)$, $a_{i,i+c+s} = (-1)^s.s.i$ $(i = 1(1)n-c-s)$ and $a_{i,i-n+c+s} = (-1)^s.s.i$ $(i = n-c-s+1(1)n)$ for $s = 1(1)r-1$, $a_{i,n-11+i+j} = j\alpha$ $(i = 1(1)11-j,\ j = 1(1)10)$, $a_{n-11+i+j,j} = i/\alpha$ $(j = 1(1)11-i,\ i = 1(1)10)$. It should be mentioned that (i) *the number of nonzero elements of a matrix generated by* MATRF2 *is* $NZ = mr+110$ *and* (ii) $\max(|a_{ij}|)/\min(|a_{ij}|) = 10\alpha^2$ (*considering nonzero elements* a_{ij} *only*).

The matrix-generators can conveniently be used to carry out some systematic investigations by fixing some of the parameters and varying the others. This is especially true for MATRF2; all parameters in this generator were varied in quite large intervals. The dependence of the reliability of the condition number estimates obtained by different algorithms on the number of nonzero elements in matrix A and on the magnitude of the nonzero elements of A is very important. Therefore many tests with matrices of class F2(m, n, c, r, α) were carried out with different values of r and α.

All experiments were run on a UNIVAC 1100/82 computer at RECKU (Regional Computing Centre at the University of Copenhagen). All computing times are given in seconds. The subroutine Y12MFE from package Y12M was used in all runs. This subroutine carries out the factorization of A and solves $Ax = b$ (in our tests b was always generated so that all components of x are equal to 1) by iterative refinement, IR; the IR process is discussed in [10], [15]–[19]. Some of the results obtained in our runs will be given in this and in the following sections. However, it should be pointed out that our conclusions are drawn by the use of the results obtained in runs of several thousands of matrices.

TABLE 1

Comparison of computing times obtained by 4 *algorithms for estimating the condition number of a matrix.* NZ *is the number of nonzero elements in matrix.* A. COUNT *is the maximal number of nonzero elements kept in array* A *during the factorization. In the column under* Y12MFE *the computing time used in the solution of* $Ax = b$ *with* IR *is given (the numbers of iterations being given in brackets). In the last* 4 *columns the computing times for the* 4 *algorithms are given (the numbers of scalings being given in brackets). The matrix from* [29] *is* A = F2(300, 300, 100, 2, 100.0).

Source	Order	*NZ*	COUNT	Y12MFE	Algorithm 1	Algorithm 3	Algorithm 2	Algorithm 2*
[5]	1,000	6,400	16,769	19.45 (21)	0.23 (0)	0.34 (5)	0.34 (6)	10.16 (6)
[7]	1,176	9,864	18,552	17.81 (3)	0.23 (0)	0.23 (4)	0.34 (4)	11.16 (4)
[1]	425	1,339	1,339	0.52 (3)	0.03 (0)	0.05 (6)	0.06 (13)	1.52 (13)
[7]	822	4,028	4,865	2.19 (3)	0.09 (0)	0.13 (8)	0.18 (35)	5.55 (35)
[29]	300	710	710	0.28 (4)	0.02 (0)	0.03 (7)	0.04 (11)	0.77 (11)

The results presented in Table 1 show that Algorithm 2* is not suitable for sparse matrices. Therefore this algorithm is not used in the other computations. Algorithm 2* performs worse both when the matrix is large (see the results for the first two matrices in Table 1) and when the sparsity is well preserved (in the case where COUNT $\approx$ *NZ*, where COUNT is the largest number of nonzero elements kept in array A; see the last three examples in Table 1).

The computing times for Algorithm 2 are normally larger than the computing times for both Algorithm 3 and Algorithm 1 but the difference is not very large for many test-matrices. Moreover, the computing times for *all* three algorithms are as a rule much smaller than the computing time needed to solve $Ax = b$. Nevertheless, it must be emphasized that there exist matrices for which Algorithm 2 is rather inefficient. Several such matrices are given in [9]. An experiment with matrices of class $E(n, c)$ was carried out with $n = 100(100)5{,}000$ and $c = 2(1)4$ for each n. In each of these examples Algorithm 2 performed about n scalings and, therefore, is very expensive. It is even more expensive than the solution of $Ax = b$ with IR; however, it must be mentioned that the matrices of class $E(n, c)$ are positive definite matrices with narrow bands for small values of c and this fact was exploited in the runs by the use of a special option of Y12M in which the pivotal search is suppressed. Results obtained in this experiment are given in Table 2.

TABLE 2

Comparison of computing times obtained in runs with matrices of class $E(n, c)$. NZ is the number of nonzero elements of matrix A. COUNT *is the maximal number of nonzero elements kept in array A during the factorization. In the column under* Y12MFE *the computing time used in the solution of $Ax = b$ by* IR *is given (the numbers of iterations being given in brackets). The computing times needed in the calculation of κ_1^* are given in the last* 3 *columns (the number of scalings being given in brackets).*

Order	c	*NZ*	COUNT	Y12MFE	Algorithm 1	Algorithm 3	Algorithm 2
	2	4,994	4,994	0.95 (5)	0.10 (0)	0.14 (5)	2.18 (998)
1,000	3	4,992	5,989	1.32 (5)	0.11 (0)	0.16 (5)	2.20 (995)
	4	4,990	7,978	1.71 (5)	0.13 (0)	0.19 (5)	2.22 (992)
	2	9,994	9,994	2.13 (5)	0.19 (0)	0.27 (5)	8.38 (1998)
2,000	3	9,992	11,989	2.64 (5)	0.22 (0)	0.33 (5)	8.43 (1995)
	4	9,990	15,978	3.41 (5)	0.26 (0)	0.38 (5)	8.47 (1992)
	2	14,994	14,994	3.20 (5)	0.29 (0)	0.41 (5)	18.61 (2998)
3,000	3	14,992	17,989	4.29 (5)	0.34 (0)	0.49 (5)	18.67 (2995)
	4	14,990	23,978	5.49 (5)	0.40 (0)	0.57 (5)	18.74 (2992)

Many runs were carried out to compare the accuracy of the estimates calculated by the three algorithms for matrices with different orders, with different sparsity patterns, with different numbers of nonzero elements and with different magnitudes of the nonzero elements. Matrices of class $F2(m, n, c, r, \alpha)$ with $m = n$, $n = 200(50)300$, $c = 20(20)n - 20$, $r = 2(7)30$ and $\alpha = 10^k$ $(k = 0(1)9)$ were applied in one of the experiments. The LINPACK subroutine SGECO was also used in these runs. "Exact" values of $\kappa_1(A)$ were calculated by the use of the LINPACK subroutine DGEDI (where double precision is used). The results of this experiment, some of them being given in Tables 3-5, can be summarized by the following remarks.

Remark 3.1.The estimates κ_1^* obtained by Algorithm 2 and Algorithm 3 were the same for all 1,750 matrices used in this experiment. Therefore only results calculated by Algorithm 1, Algorithm 2 and the LINPACK subroutine SGECO are given in Tables 3-5. The numbers of scalings in Algorithm 2 were much smaller than n and the computing times for Algorithm 2 and Algorithm 3 were of the same order (being much smaller than the computing times needed to factorize A).

Remark 3.2. No relationship between the order n of the matrix and the accuracy of κ_1^* was observed for the matrices of this class. Neither was there a relationship

TABLE 3

Estimates of the condition numbers of matrices $A = \text{F2}(300, 300, 100, 2, \alpha)$. *The ratios between the estimates* κ^* *and the "exact"* $\kappa(A)$ *are given in brackets for each algorithm.*

α	Algorithm 1	Algorithm 2	LINPACK	$\kappa_1(A) = \|A\|_1 \|A^{-1}\|_1$
10^0	$7.50 * 10^1$ (0.24)	$7.50 * 10^1$ (0.24)	$7.50 * 10^1$ (0.24)	$3.04 * 10^2$
10^1	$3.72 * 10^2$ (0.44)	$3.72 * 10^2$ (0.44)	$3.72 * 10^2$ (0.44)	$8.46 * 10^2$
10^2	$3.88 * 10^4$ (0.61)	$3.84 * 10^4$ (0.61)	$3.84 * 10^4$ (0.61)	$6.40 * 10^4$
10^3	$4.12 * 10^6$ (0.66)	$4.12 * 10^6$ (0.66)	$4.12 * 10^6$ (0.66)	$6.19 * 10^6$
10^4	$4.15 * 10^8$ (0.67)	$4.15 * 10^8$ (0.67)	$4.15 * 10^8$ (0.67)	$6.17 * 10^8$
10^5	$4.16 * 10^{10}$ (0.67)	$4.16 * 10^{10}$ (0.67)	$4.16 * 10^{10}$ (0.67)	$6.17 * 10^{10}$
10^6	$4.16 * 10^{12}$ (0.67)	$4.16 * 10^{12}$ (0.67)	$4.16 * 10^{12}$ (0.67)	$6.17 * 10^{12}$
10^7	$4.16 * 10^{14}$ (0.67)	$4.15 * 10^{14}$ (0.67)	$4.16 * 10^{14}$ (0.67)	$6.17 * 10^{14}$
10^8	$4.16 * 10^{16}$ (0.67)	$4.16 * 10^{16}$ (0.67)	$4.16 * 10^{16}$ (0.67)	$6.17 * 10^{16}$
10^9	$4.16 * 10^{18}$ (0.67)	$4.16 * 10^{18}$ (0.67)	$4.16 * 10^{18}$ (0.67)	$6.17 * 10^{18}$

observed between the sparsity pattern (determined by parameter c) of the matrix and κ_1^*. Results obtained with $n = 300$ and $c = 100$ are given in Tables 3-5.

Remark 3.3. If $r = 2$, then the estimates of $\kappa_1(A)$ found by the different algorithms are practically the same; the results for $n = 300$ are given in Table 3. This is not a surprise and can be explained as follows. The number of the nonzero elements in nearly all rows and columns of the matrices of class F2 is equal to 2 when $r = 2$. Moreover, the sparsity pattern is perfectly preserved for such a matrix when the GE transformations are performed; COUNT $= NZ = 710$ for $n = 300$. Therefore Algorithm 1 and Algorithm 2 should perform in a quite similar way; see (1.4)-(1.5) and use the

TABLE 4

Estimates of the condition numbers of matrices $A = \text{F2}(300, 300, 100, 23, \alpha)$. *The ratios between the estimates* κ^* *and the "exact"* $\kappa(A)$ *are given in brackets for each algorithm.*

α	Algorithm 1	Algorithm 2	LINPACK	$\kappa_1(A) = \|A\|_1 \|A^{-1}\|_1$
10^0	$2.05 * 10^3$ (0.30)	$2.11 * 10^3$ (0.31)	$2.38 * 10^3$ (0.35)	$6.87 * 10^3$
10^1	$2.02 * 10^3$ (0.29)	$2.12 * 10^3$ (0.31)	$2.26 * 10^3$ (0.33)	$6.87 * 10^3$
10^2	$2.09 * 10^3$ (0.30)	$2.15 * 10^3$ (0.31)	$3.75 * 10^3$ (0.54)	$6.91 * 10^3$
10^3	$4.41 * 10^4$ (0.45)	$5.80 * 10^4$ (0.60)	$5.91 * 10^4$ (0.61)	$9.70 * 10^4$
10^4	$3.62 * 10^6$ (0.64)	$3.40 * 10^6$ (0.60)	$3.65 * 10^6$ (0.64)	$5.68 * 10^6$
10^5	$3.37 * 10^8$ (0.63)	$3.44 * 10^8$ (0.65)	$3.44 * 10^8$ (0.65)	$5.32 * 10^8$
10^6	$3.71 * 10^{10}$ (0.64)	$3.67 * 10^{10}$ (0.63)	$3.71 * 10^{10}$ (0.64)	$5.81 * 10^{10}$
10^7	$8.74 * 10^{12}$ (0.61)	$8.74 * 10^{12}$ (0.61)	$8.78 * 10^{12}$ (0.61)	$1.44 * 10^{12}$
10^8	$2.19 * 10^{14}$ (0.64)	$2.21 * 10^{14}$ (0.65)	$2.14 * 10^{14}$ (0.63)	$3.40 * 10^{14}$
10^9	$3.98 * 10^{16}$ (0.70)	$3.97 * 10^{16}$ (0.70)	$3.98 * 10^{16}$ (0.70)	$5.66 * 10^{16}$

TABLE 5

The intervals for the ratios $R = \kappa^*/\kappa(A)$ *where* $A = \text{F2}(300, 300, 100, r, \alpha)$ *with* $\alpha = 10^k$ $(k = 0(1)9)$ *for each r.*

r	Algorithm 1	Algorithm 2	LINPACK
2	[0.24, 0.67]	[0.24, 0.67]	[0.24, 0.67]
9	[0.29, 0.68]	[0.14, 0.66]	[0.26, 0.63]
16	[0.15, 0.67]	[0.15, 0.67]	[0.30, 0.63]
23	[0.29, 0.70]	[0.31, 0.70]	[0.33, 0.70]
30	[0.29, 0.71]	[0.29, 0.71]	[0.33, 0.71]

fact that for nearly all values of k only one t_i is different from t_i^+ and t_i^-. An element $a_{ij}^{(k)}$ can be chosen as pivotal in package Y12M if $u|a_{ij}^{(k)}| \leqq \max_{k \leqq j \leqq n} (|a_{ij}^{(k)}|)$. Parameter u is called the stability factor; [20], [23]-[25], [28]. The runs discussed in this section were carried out with $u = 4$. Since many rows contain only 2 nonzero elements and since the ratio of the absolute values of these two elements is normally larger than 4, the sparse pivotal strategy with $u = 4$ used in package Y12M and the partial pivoting, where $u = 1$, used in LINPACK perform in a quite similar way when $r = 2$. The above arguments are not true for $r > 2$ and the results obtained by the three algorithms were different for all $r > 2$ used in the runs. This is illustrated for $r = 23$ in Table 4. It should be mentioned that the differences are normally not very large; see Table 4 and Table 5.

Remark 3.4. Let $R = \kappa_1^*/\kappa_1(A)$. For all algorithms this ratio increases when α is increased from 10^0 to 10^4. For $\alpha > 10^4$ R varies normally in the interval [0.6, 0.7]. This means that the behaviour shown in Table 4 is typical for the matrices in this experiment.

Remark 3.5. In *all* runs $R \geqq 0.1$ was observed for *all* algorithms. When $n = 300$ and $c = 100$ this is demonstrated in Table 5. This shows that the estimates of the condition number are acceptable according to the criterion in [3, p. 375].

Remark 3.6. The parameter γ used in Algorithm 3 (see also [9]) to determine the proper scaling factors was set equal to 0.001 in this experiment. This is the value recommended in [9] when an attempt to minimize the number of scalings is carried out. It should be mentioned that the user has a possibility in varying the value of this parameter.

4. Sparse pivotal strategies and accuracy of the condition number estimates. The pivotal element that is to be used at step k ($k = 1(1)n-1$) has to be chosen so that: (i) *the sparsity pattern of A is preserved as well as possible* and (ii) *the stability of the computations is preserved as well as possible.* These two requirements work in opposite directions and cannot be satisfied simultaneously. Therefore a compromise is necessary. Any compromise leads normally to a relaxation of the stability requirements during the factorization (compared with the stability requirements in the partial pivoting for dense matrices). The influence of the relaxed stability requirements on the accuracy of κ^* is discussed in this section. A short description of the main pivotal strategy in package Y12M is needed before the discussion.

Suppose that the kth pivot is to be found. The sets $A_k = \{a_{ij}^{(k)} / k \leqq i, j \leqq n\}$, $R_{ik} = \{a_{ij}^{(k)} / k \leqq j \leqq n\}$ and $C_{jk} = \{a_{ij}^{(k)} / k \leqq i \leqq n\}$ are called *active parts at stage k* of matrix A, row i ($i = k(1)n$) and column j ($j = k(1)n$), respectively. Let $a_{ij}^{(k)} \in A_k$ be an arbitrary nonzero element in the active part at stage k of matrix A. The integer $M_{ijk} = [r(i, k)-1][c(j, k)-1]$, where $r(i, k)$ and $c(j, k)$ are the numbers of nonzero elements in the active parts at stage k of row i and column j, is called the *Markowitz cost* of element $a_{ij}^{(k)}$. Consider the set of *row* numbers:

$$(4.1) \qquad I_k = \{i_1, i_2, \cdots, i_{p(k)}\} \qquad (k = 1(1)n-1,\ i_m \in \{k, k+1, \cdots, n\},\ m = 1(1)p(k)).$$

Assume that the rows are ordered in nondecreasing numbers of nonzero elements in their active parts at stage k, i.e., $r(i_1, k) \leqq r(i_2, k) \leqq \cdots \leqq r(i_{p(k)}, k)$, and that the number of nonzero elements in the active part of any of the other rows in A_k (when there are such rows) is not smaller than $r(i_{p(k)}, k)$, i.e. that $(i \notin I_k \wedge k \leqq i \leqq n) \Rightarrow r(i_{p(k)}, k) \leqq r(i, k)$. The rows of set I_k are called the *best rows.* It is expected that the sparsity is preserved in a best way if the pivotal elements are in the rows whose row numbers belong to I_k ($k = 1(1)n-1$). Let

$$(4.2) \qquad B_k = \{a_{ij}^{(k)} / a_{ij}^{(k)} \in A_k \wedge u|a_{ij}^{(k)}| \leqq \max_{k \leqq j \leqq n} (|a_{ij}^{(k)}|) \wedge i \in I_k\}, \qquad u \geqq 1,$$

$$M_k = \min (M_{ijk}) \quad \text{for } a_{ij}^{(k)} \in B_k, \tag{4.3}$$

$$C_k = \{a_{ij}^{(k)} / a_{ij}^{(k)} \in B_k \wedge M_{ijk} = M_k\}. \tag{4.4}$$

Set B_k is called *the stability set at stage k*, the integer M_k is called *the optimal Markowitz cost at stage k* and the set C_k is called *the set of candidates for pivotal elements at stage k*. The real number u involved in (4.2) is called *the stability factor.*

The class of *improved generalized Markowitz strategies* (IGMS's; [20], [23], [24]) contains all pivotal strategies in which one of the maximal in absolute value elements in C_k $(k = 1(1)n-1)$ is chosen as pivotal at stage k of the GE. The class of IGMS's is a two-parameter family of pivotal strategies depending on the stability factor u and on the number $p(k)$ of best rows among the nonzero elements of which the pivotal element at stage k $(k = 1(1)n-1)$ must be selected. In package Y12M $u \in [4, 16]$ and $p(k) = \min$ (RPIV, $n-k+1$) are recommended (RPIV being a small integer; RPIV < 5 is recommended). An attempt to minimize (*locally*) the maximal number of fill-ins at stage k by taking into account the stability restriction (4.2) is made by (4.3)-(4.4).

Consider *the minimal Markowitz cost at stage k*, $M_k^* = \min (M_{ijk})$ for $a_{ij}^{(k)} \in A_k \wedge a_{ij}^{(k)} \neq 0$. It is clear that $M_k \geqq M_k^*$. If $M_k > M_k^*$, then *either* there is no nonzero element in the $p(k)$ best rows for which $M_{ijk} = M_k^*$ (in other words, $p(k)$ is too small) *or* there are nonzero elements among the $p(k)$ best rows for which $M_{ijk} = M_k^*$ but the stability requirement (4.2) is not satisfied for these elements (in other words, u is too small). This shows that one should expect to preserve the sparsity better by increasing either $p(k)$ or u. It is not very efficient to carry out the GE with a large $p(k)$ because this leads to a very considerable increase of the calculations in the pivotal search. Moreover, the elements for which $M_{ijk} = M_k^*$ are normally in rows with small numbers of nonzero elements in their active parts, but cannot be found because of the stability restriction (4.2). These arguments as well as many numerical tests indicate that as a rule it is not justified to use $p(k) > 4$. On the other hand, the sparsity can sometimes be preserved better by choosing a larger stability factor u. However, the use of a large u may cause large errors in the factors L and U. Indeed, it is easy to obtain the following bounds (see, for example, [20]):

$$|a_{ij}^{(k)}| \leqq (1+u) \max (|a_{ij}^{(k)}|) \quad \text{for } a_{ij}^{(k)} \in A_k, \tag{4.5}$$

$$b_n \leqq (1+u)^{n-1} b_1, \quad \text{where } b_m = \max_{1 \leqq k \leqq m} (|a_{ij}^{(k)}|), \quad m = 1(1)n, \tag{4.6}$$

$$|e_{ij}| \leqq 3.01 b_n \varepsilon n, \quad \varepsilon \text{ being the computer precision}, \quad e_{ij} \in E = LU - A. \tag{4.7}$$

The bounds show that the norm of matrix E from $LU = A + E$ may be very large. The commonly used partial pivoting for dense matrices can, roughly speaking, be obtained from the IGMS's by setting $u = 1$ and $p(k) = 1$. The above bounds (the last one with 2.01 instead of 3.01, see [18]), show again that the norm of E may be very large also when A is dense. Moreover, examples, where $b_n = 2^{n-1} b_1$, can be constructed. However, it is well known that in practice the norm of E is much smaller than the norm of A and the computation of κ_1^* for dense matrices A ($A+E$ being actually used in the calculations) is based on the assumption that this is true, [3, § 5]. *Thus, the above bounds are too pessimistic when A is dense and when partial pivoting is in use. Our experience indicates that this is also true for sparse matrices even when the stability factor u is rather large.*

The experiments from § 3, where $u = 4$ was used, were repeated with $u = 1{,}024$. In all cases $R > 0.1$ was found, where R is the ratio between the estimated κ_1^* and the "exact" $\kappa(A)$ obtained by computing A^{-1} in double precision. Another experiment

TABLE 6
Intervals for $R = \kappa^*/\kappa(A)$ *when* $A = \text{F2}(200, 200, 100, 8, \alpha)$ *with* $\alpha = 10^k$, $k = 0(1)9$.

u	Algorithm 1	Algorithm 2
4.0	[0.31, 0.67]	[0.12, 0.67]
16.0	[0.28, 0.66]	[0.30, 0.66]
64.0	[0.30, 0.68]	[0.15, 0.68]
256.0	[0.31, 0.66]	[0.31, 0.67]
1,024.0	[0.30, 0.67]	[0.30, 0.68]

with several different values of u was performed. In this experiment matrices $A = \text{F2}(200, 200, 100, 8, \alpha)$ with $\alpha = 10^k$, $k = 0(1)9$, were used. The values of u were 2^k, $k = 2(2)10$. The results are shown in Table 6. For the LINPACK subroutine SGECO the values of R were in the interval [0.27, 0.67] when the same matrices were run.

The results given in Table 6 as well as results obtained for a very large set of matrices confirm the above statement. An explanation of the fact that the condition number estimators for sparse matrices give good results even if u is rather large can be given as follows. Let u_k^* be such that $u_k^* |a_{ij}^{(k)}| = \max(|a_{ij}^{(k)}|)$, $k = 1(1)n-1$, holds for the element that will be chosen as pivotal at stage k. The relationship $u_k^* \ll u$ occurs often in the GE when u is large. *The possibility of having* $u_k^* \ll u$ *at stage k is enhanced by the fact that one of the largest in absolute value elements in* C_k *is chosen as a pivot.* If this requirement is violated by allowing any element of C_k to be chosen as a pivot (the resulting sparse pivotal strategy being called a GMS in [20], [24]), then the accuracy of L and U could be very poor for some matrices even for small values of u. If this is so, then the calculated solution of $Ax = b$ may be quite wrong [24], [31].

TABLE 7
Estimates of κ^* *for matrices* $A = \text{E2}(3600, c)$ *obtained by two sparse pivotal strategies and* LINPACK. *The largest error in the solution vector is given in brackets; all components of the exact solution are equal to* 1. *The growth of the elements during the* GE *is given in the last column for the* GMS; *the growth factor is equal to* 1 *when the* IGMS *is in use.*

c	GMS used		IGMS used		LINPACK (SPBCO)		b_n/b_1
200	$5.25 * 10^2$	$(8.34 * 10^0)$	$2.89 * 10^2$	$(8.77 * 10^{-5})$	$2.89 * 10^2$	$(6.21 * 10^{-4})$	$7.08 * 10^7$
300	$1.89 * 10^3$	$(6.31 * 10^2)$	$1.36 * 10^2$	$(3.15 * 10^{-5})$	$1.36 * 10^2$	$(2.69 * 10^{-4})$	$2.22 * 10^{10}$
400	$8.01 * 10^1$	$(9.18 * 10^{-1})$	$8.07 * 10^1$	$(2.19 * 10^{-5})$	$8.06 * 10^1$	$(8.69 * 10^{-5})$	$5.99 * 10^7$

It is surprising that in our experiments with matrices $E(n, c)$, $n \in [1{,}000, 10{,}000]$, $c \in [4, n-100]$, the calculated κ_1^* were not very bad (even when the solution is quite wrong); some examples are given in Table 7. The matrices used in this experiment are well-scaled, symmetric and positive definite, not very ill-conditioned and, what is most important, similar matrices arise often after the space discretization of partial differential equations. This illustrates the great importance of choosing a pivotal strategy of the class of IGMS's. Of course if u is very large, then the accuracy of both the calculated solution of $Ax = b$ and of κ_1^* may be poor for some matrices; see Table 8. However, the value of b_n/b_1 and/or the value of the minimal pivot used in the GE (both are calculated in package Y12M) will normally give a signal for the trouble. The values of b_n/b_1 are given in Table 8. For single precision computations on UNIVAC 1100/82 $\varepsilon = 1.49 * 10^{-8}$. Therefore $\varepsilon(b_n/b_1) \approx O(10^3)$ for the matrices in Table 8. In all our

TABLE 8

Estimates of κ^* for matrices $A = \text{F2}(300, 300, 100, 30, \alpha)$ obtained with $u = 2^{14}$ (the growth of the elements during the GE is shown in the last column).

α	Algorithm 1	Algorithm 2	LINPACK	$\kappa_1 = \|A\|_1 \|A^{-1}\|_1$	b_n/b_1
10^0	$3.65 * 10^5$	$3.55 * 10^6$	$2.69 * 10^3$	$8.74 * 10^3$	$2.35 * 10^{11}$
10^1	$9.75 * 10^4$	$8.33 * 10^4$	$2.55 * 10^3$	$8.74 * 10^3$	$1.73 * 10^{10}$
10^2	$3.12 * 10^6$	$6.84 * 10^5$	$3.49 * 10^3$	$8.74 * 10^3$	$5.11 * 10^{11}$
10^3	$8.01 * 10^5$	$2.63 * 10^5$	$4.68 * 10^4$	$7.65 * 10^4$	$1.55 * 10^{10}$
10^4	$2.36 * 10^7$	$2.19 * 10^7$	$2.31 * 10^6$	$3.57 * 10^6$	$3.75 * 10^{10}$

experiments the results were always satisfactory when $\varepsilon(b_n/b_1) \lessapprox O(1)$. More experiments are needed in order to decide if such a criterion can be applied in the judgement of the accuracy of the results obtained by the sparse condition number estimators.

Our experiments show that good estimates of the condition numbers of sparse matrices can be obtained when the stability factor u varies in a quite large interval. If u is large, then it is useful to check b_n/b_1 ("*the growth factor*") and/or the size of the minimal pivot. It must be emphasized that normally u is varied in [4, 16] and for such values of u the main pivotal strategy in package Y12M (which is of the class of IGMS's) gave good results in all runs. Therefore we may conclude that if $u \in [4, 16]$, then the sparse pivotal strategy ensures as good estimates κ_1^* as those obtained by the use of partial pivoting in the LINPACK subroutines.

5. Drop-tolerance and accuracy of the condition number estimates. A nonnegative parameter T, *a drop-tolerance*, is used in the GE process carried out by package Y12M. If $T > 0$ and if $|a_{ij}^{(k)}| \leqq T$, then $a_{ij}^{(k)}$ is removed from array A (see § 2). This means that this element is considered as a zero element in the further computations. In all experiments from § 3 and § 4 a very small value of T ($T = 10^{-25}$) was used. An attempt to investigate the influence of the drop-tolerance T on the performance of the condition number estimators for sparse matrices is carried out in this section. Some of the matrices (matrices $A = \text{F2}(300, 300, 100, r, \alpha)$ with $\alpha = 10^k$, $k = 0(1)9$, and with $r = 2(7)30$) used in § 3 with $T = 10^{-25}$ were also run with large values of the drop-tolerance, $T = 10^{-k}$, $k = 3(-1)1$ and with a stability factor $u = 4$. If $\alpha \leqq 10^4$, then the estimates κ_1^* were worse than those calculated with $T = 10^{-25}$ but not very bad. The intervals in which $R = \kappa_1^*/\kappa(A)$ varies are given in Table 9 for Algorithm 1 and Algorithm 2; the estimates calculated by Algorithm 3 were again the same as these calculated by Algorithm 2. If $\alpha > 10^4$, then the estimates are often very bad. Some results for $r = 9$ and $T = 10^{-1}$ are given in Table 10. Normally $R > 1$ when $\alpha > 10^4$ and when T is large. However, sometimes $R < 0.1$ has also been observed; see the second matrix in Table 10.

The conclusion from this experiment and from many other runs (the matrices from § 4 were run with $T = 10^{-3}$) is that the reliability of the sparse condition number estimators is suspicious when T is large. More precisely, the accuracy of κ_1^* may be poor for not very ill-conditioned matrices and the estimates are usually very bad when the matrices are very ill-conditioned. However, the use of a large drop-tolerance has to be combined with the use of iterative refinement, [23], [24], [31]. In this case reliable information about the result of the solution process is available from the iterative refinement process, [8, p. 49]. If iterative refinement is *not* used, then the drop-tolerance should be small and the sparse condition number estimators work very satisfactorily when the stability factor u is not very large; see § 3 and § 4.

TABLE 9
Intervals for $R = \kappa^*/\kappa(A)$ *for matrices* $A = \mathrm{F2}(300, 300, 100, r, \alpha)$ *with* $\alpha = 10^k$, $k = 0(1)4$ *for each* r.

T	r	Algorithm 1	Algorithm 2
	2	[0.24, 0.67]	[0.24, 0.67]
	9	[0.13, 0.62]	[0.06, 0.65]
10^{-3}	16	[0.12, 0.61]	[0.11, 0.62]
	23	[0.06, 0.59]	[0.04, 0.59]
	30	[0.39, 0.60]	[0.05, 0.59]
	2	[0.24, 0.67]	[0.24, 0.67]
	9	[0.13, 0.60]	[0.13, 0.59]
10^{-2}	16	[0.17, 0.60]	[0.17, 0.60]
	23	[0.34, 0.64]	[0.16, 0.64]
	30	[0.33, 0.64]	[0.33, 0.65]
	2	[0.24, 0.67]	[0.24, 0.67]
	9	[0.09, 0.72]	[0.09, 0.68]
10^{-1}	16	[0.34, 0.60]	[0.33, 0.69]
	23	[0.07, 0.59]	[0.07, 0.67]
	30	[0.04, 0.64]	[0.08, 0.64]

6. Concluding remarks. Three sparse condition number estimators have been developed and attached to package Y12M (see [23]-[25], [27]-[31]). These estimators have been tested with a large set of matrices. The results of the tests can be summarized as follows.

Remark 6.1. Algorithm 1 is the most efficient algorithm with regard to computing time used. Algorithm 2 may be inefficient for some matrices because too many scalings are carried out. However, this happens very seldom. It was very difficult to find matrices different from those given in [9] for which the number of scalings is large. Algorithm 3 is more expensive than Algorithm 1 but in general cheaper than Algorithm 2. Moreover, if the number of scalings is large for Algorithm 2, then it was always reduced very considerably by Algorithm 3 in our experiments.

Remark 6.2. Algorithm 2 is the safest. A very careful attempt to avoid overflows and divisions by zero is carried out by scaling different vectors when this algorithm is applied. Algorithm 1 is not so robust. The computations in this algorithm are carried out without any scaling. Algorithm 3 is again in the middle: it is not so robust as Algorithm 2 but is more robust than Algorithm 1.

Remark 6.3. The fact that the accuracy requirements are relaxed in the sparse pivotal strategies (compared with the accuracy requirements in the commonly used partial pivoting for dense matrices) seems to be not very important when both the stability factor u and the drop-tolerance T are sufficiently small. The results obtained

TABLE 10
Estimates κ^* *for ill-conditioned matrices* $A = \mathrm{F2}(300, 300, 100, 9, \alpha)$ *obtained by the use of* $T = 0.1$.

α	Algorithm 1	Algorithm 2	LINPACK	$\kappa_1(A) = \|A\|_1 \|A^{-1}\|_1$
10^5	$6.45 * 10^9$	$6.45 * 10^9$	$2.20 * 10^9$	$3.48 * 10^9$
10^6	$3.24 * 10^{10}$	$3.20 * 10^{10}$	$2.20 * 10^{11}$	$3.48 * 10^{11}$
10^7	$5.23 * 10^{19}$	$5.23 * 10^{19}$	$2.20 * 10^{13}$	$3.48 * 10^{13}$
10^8	$4.93 * 10^{20}$	$4.93 * 10^{20}$	$2.20 * 10^{15}$	$3.48 * 10^{15}$
10^9	$1.15 * 10^{22}$	$1.15 * 10^{22}$	$2.20 * 10^{17}$	$3.48 * 10^{17}$

in runs of several thousands of matrices with $u = 4$ and $T = 10^{-25}$ were acceptable in the sense that $R = \kappa_1^*/\kappa_1(A)$ was larger than 0.1 for all matrices tested and for all three algorithms.

Remark 6.4. It is not very easy to answer the question: *which algorithm is the best*? Examples where Algorithm 1 fails to find a good estimate can be constructed [3]. However, examples in which Algorithm 2 fails to calculate a good estimate can also be constructed, [2], [4]. This means that no algorithm, the statement being also true for Algorithm 3, will guarantee a good estimate of the condition number for any matrix. On the other hand, experience, including here the results of our experiments, indicates that this situation occurs very seldom (especially when the matrix is large). The situation is probably similar to that for the partial pivoting: *one knows that examples, where the partial pivoting fails, can be constructed, but experience indicates that this situation will happen very seldom in practice.* More experiments are needed in order to confirm or reject such a similarity between the condition number estimators and the partial pivoting.

Remark 6.5. It is also difficult to answer the question: *which algorithm should be kept in a package for sparse matrices*? Algorithm 1 should be chosen if one wants to keep the cheapest algorithm. Algorithm 2 should be kept if the safety is the most desired property. Algorithm 3 is a rather good compromise. This shows that each algorithm may be the best one in special circumstances. Therefore, we decided to keep temporarily all of them as options in a condition number subroutine. The default option is Algorithm 2. The reasons for this choice are the following. Normally, this algorithm carries out the computations with a few scalings only and the computing time for this algorithm is as a rule very small compared with the computing time for the factorization. At the same time, the computational process for this algorithm is very robust (with regards to overflows and divisions by zero).

Remark 6.6. The subroutine for calculating estimates of condition numbers for sparse matrices is fully documented and is available at the usual costs (for machine time, magnetic tapes, shipping etc.). Requests should be addressed to J. Wasniewski (Regional Computing Centre at the University of Copenhagen). It should be mentioned here that the sparse matrix estimators can be used in connection with package Y12M.

REFERENCES

[1] I. D. L. BOGLE, *A comparison of methods for solving large sparse systems of nonlinear equations*, Report, Department of Chemical Engineering and Chemical Technology, Imperial College, London, 1982.

[2] A. K. CLINE, A. R. CONN AND C. F. VAN LOAN, *Generalizing the* LINPACK *condition estimator*, in Numerical Analysis, Lecture Notes in Mathematics 909, J. P. Hennart, ed. Springer, Berlin, 1981, pp. 73-83.

[3] A. K. CLINE, C. B. MOLER, G. W. STEWART AND J. H. WILKINSON, *An estimate for the condition number of a matrix*, SIAM J. Numer. Anal., 16 (1979), pp. 368-375.

[4] A. K. CLINE AND R. K. REW, *A set of counter-examples to three condition number estimators*, this Journal, 4 (1983), pp. 602-611.

[5] J. J. DONGARRA, G. K. LEAF AND M. MINKOFF, *A preconditioned conjugate gradient method for solving a class of non-symmetric linear systems*, Report No. ANL-81-71, Argonne National Laboratory, Argonne, IL, 1981.

[6] J. J. DONGARRA, J. R. BUNCH, C. B. MOLER AND G. W. STEWART, LINPACK *Users' Guide*, Society for Industrial and Applied Mathematics, Philadelphia, 1979.

[7] I. S. DUFF AND J. K. REID, *Performance evaluation of codes for sparse matrices*, Report No. CSS 66, AERE, Harwell, England, 1978.

[8] G. E. FORSYTHE, M. A. MALCOLM AND C. B. MOLER, *Computer Methods for Mathematical Computations*, Prentice-Hall, Englewood Cliffs, NJ, 1977.

[9] R. G. GRIMES AND J. G. LEWIS, *Condition number estimation for sparse matrices*, this Journal, 2 (1981), pp. 384-388.
[10] C. B. MOLER, *Iterative refinement in floating point*, J. Assoc. Comput. Mach., 14 (1967), pp. 316-321.
[11] ———, *Three research problems in numerical linear algebra*, in Numerical Analysis, Proceedings of the Symposia in Applied Mathematics 22, G. H. Golub and J. Oliver, eds., American Mathematical Society, Providence, Rhode Island, 1978, pp. 1-18.
[12] D. P. O'LEARY, *Estimating matrix condition numbers*, this Journal, 1 (1980), pp. 205-209.
[13] K. SCHAUMBURG, J. WASNIEWSKI AND Z. ZLATEV, *Solution of ordinary differential equations. Development of a semiexplicit Runge-Kutta algorithm and applications to a spectroscopic problem*, Comput. Chem., 3 (1979), pp. 57-63.
[14] ———, *The use of sparse matrix technique in the numerical integration of stiff systems of linear ordinary differential equations*, Comput. Chem., 4 (1980), pp. 1-12.
[15] G. W. STEWART, *Introduction to Matrix Computations*, Academic Press, New York, 1973.
[16] V. V. VOEVODIN, *Computational Bases of the Linear Algebra*, Nauka, Moscow, 1977. (In Russian.)
[17] J. H. WILKINSON, *Error analysis of direct methods of matrix inversion*, J. Assoc. Comput. Mach., 8 (1961), pp. 281-330.
[18] ———, *Rounding errors in algebraic processes*, Notes in Applied Science, No. 32, HMSO, London, 1963.
[19] ———, *The Algebraic Eigenvalue Problem*, Oxford University Press, London, 1965.
[20] Z. ZLATEV, *On some pivotal strategies in Gaussian elimination by sparse technique*, SIAM J. Numer. Anal., 17 (1980), pp. 18-30.
[21] ———, *Modified diagonally implicit Runge-Kutta methods*, this Journal, 2 (1981), pp. 321-334.
[22] ———, *Comparison of two pivotal strategies in sparse plane rotations*, Comput. Math. Appl., 8 (1982), pp. 119-135.
[23] ———, *Use of iterative refinement in the solution of sparse linear systems*, SIAM J. Numer. Anal., 19 (1982), pp. 381-399.
[24] ———, *Sparse matrix technique for general matrices with real elements: pivotal strategies, decompositions and applications in* ODE *software*, in Sparsity and its Applications, D. J. Evans, ed. Cambridge University Press, Cambridge, 1985, pp. 185-228.
[25] Z. ZLATEV, K. SCHAUMBURG AND J. WASNIEWSKI, *Implementation of an iterative refinement option in a code for large and sparse systems*, Comput. Chem., 4 (1980), pp. 87-99.
[26] Z. ZLATEV AND P. G. THOMSEN, *Application of backward differentiation methods to the finite element solution of time dependent problems*, J. Numer. Meth. Engng., 14 (1979), pp. 1051-1061.
[27] ———, *Sparse matrices—efficient decompositions and applications*, in Sparse Matrices and Their Uses, I. S. Duff, ed., Academic Press, London, 1981, pp. 367-375.
[28] Z. ZLATEV, J. WASNIEWSKI AND K. SCHAUMBURG, Y12M—*solution of large and sparse systems of linear algebraic equations*, Lecture Notes in Computer Science, Vol. 121, Springer, Berlin, 1981.
[29] ———, *A testing scheme for subroutines for solving large linear problems*, Comput. Chem., 5 (1981), pp. 91-100.
[30] ———, *Comparison of two algorithms for solving large linear systems*, this Journal, 3 (1982), pp. 486-501.
[31] O. ØSTERBY AND Z. ZLATEV, *Direct methods for sparse matrices*, Lecture Notes in Computer Science, Vol. 157, Springer, Berlin, 1983.

SIAM J. SCI. STAT. COMPUT.
Vol. 7, No. 4, October 1986

009

ON GENERAL ROW MERGING SCHEMES FOR SPARSE GIVENS TRANSFORMATIONS*

JOSEPH W. H. LIU†

Abstract. This paper introduces general row merging schemes for the QR decomposition of sparse matrices by Givens rotations. They can be viewed as a generalization of row rotations to submatrix rotations (or merging) in the recent method by George and Heath [12]. Based on the column ordering and the structure of the given sparse matrix, we present an algorithm to determine automatically a sequence of submatrix rotations appropriate for sparse decomposition. It is shown that the actual numerical computation can be organized as a sequence of reductions of two upper trapezoidal full submatrices into another upper trapezoidal full matrix. Experimental results are provided to compare the practical performance of the proposed method and the George-Heath scheme. Significant reduction in arithmetic operations and factorization time is achieved in exchange for a very modest increase in working storage. The interpretation of general row merging as a special variable row pivoting method is also presented.

Key words. sparse matrix, orthogonal decomposition, Givens rotation, row/submatrix merging, variable row pivoting

AMS(MOS) subject classifications. 65F50, 65F25

1. Introduction. The use of Givens transformations in the orthogonal decomposition of a large sparse matrix is known to be very effective [6], [10], [12], [19], [20], [27]. Throughout this paper, A is assumed to be a large sparse m by n matrix ($m > n$) with full column rank. A sequence of Givens rotations can be applied to A to reduce it to upper trapezoidal form, that is,

$$A = Q\begin{pmatrix} R \\ 0 \end{pmatrix},$$

where Q is an m by m orthogonal matrix defined by the sequence of rotations, and R is an n by n upper triangular matrix.

In the triangulation of A by orthogonal transformations, it is well known that the factor R is mathematically equivalent to the Cholesky factor of A^TA. Since

$$A^TA = (P_rA)^T(P_rA)$$

for any row permutation P_r of A, the structure of the resulting triangular factor R depends only on the column ordering, but not on the row ordering of A [17], [20], [31]. Based on the connection between orthogonal and Cholesky factors, George and Heath [12] recommend ordering the columns of A using fill-reducing orderings for the symmetric and positive definite matrix A^TA, a task with reliable and efficient algorithms and robust implementations [15]. The use of such column orderings will have the desirable effect of reducing the number of nonzeros in the resulting factor matrix R and the number of operations to compute R.

* Received by the editors August 30, 1983, and in revised form August 7, 1985. This research was supported in part by the Canadian Natural Sciences and Engineering Research Council under grant A5509, and by the Applied Mathematical Sciences Research Program, Office of Energy Research, U.S. Department of Energy under contract DE-AC05-84OR21400 with Martin Marietta Energy Systems Inc., and by the U.S. Air Force Office of Scientific Research under contract AFOSR-ISSA-84-00056. This paper was prepared using a *troff* program running on the UNIX operating system.

† Department of Computer Science, York University, Downsview, Ontario, Canada M3J 1P3.

Assume that the columns of A have been ordered appropriately. George and Heath also devise a computational scheme, with the advantage of having simple and efficient data storage management. The method predetermines the sparsity structure of the factor R symbolically based on the structure of A^TA prior to any numerical computation. After a static data structure has been set up for R, the data matrix A is accessed one row at a time, and the row is rotated into the static storage for R. Thus, the scheme can be viewed as one with a sequence of row rotations merging rows into a fixed upper triangular factor structure. Experimental results in [13] indicate that their method can be used as an effective practical scheme to solve sparse linear least squares problems.

In this paper, we consider the generalization of row rotations to submatrix rotations in the George-Heath scheme. In essence, instead of having only one (fixed) upper triangular structure, we allow the use of more than one (necessarily dynamic) triangular structures in the course of computation. In other words, a row may be annihilated or merged into one of many templates of upper triangular matrices, which will eventually be merged together to form one single triangular factor matrix R. We use *general row merging schemes* to refer to this approach, which will result in highly efficient and practical schemes for sparse orthogonal factorization.

An outline of this paper is as follows. In section 2, a submatrix rotation is formally defined. We also introduce the notion of a row merge tree, which can be used to induce a sequence of such submatrix rotations. The connection of general row merging with *variable row pivoting* methods (Gentleman [10]) is also presented. We can view the scheme as one that generates a special variable row pivoting sequence.

In section 3, we present an algorithm that will generate an appropriate row merge tree for sparse rotations. The motivation for this algorithm is also discussed; it is based on some well-understood observations in the Cholesky decomposition of sparse positive definite symmetric matrices. There is a close connection between the row merge tree introduced in this paper and the element merge tree used by Eisenstat et al. [8] and the elimination tree employed by Duff [5], and others [21], [24], [30].

Some relevant properties of the generated row merge tree are also presented in section 3. We show that the entire computation process based on this tree can be organized into a sequence of merging of *upper trapezoidal full* submatrices. Full matrix techniques can then be used for sparse QR decomposition, as in the case of sparse Gaussian elimination [5], [28].

Section 4 contains a discussion of the overall sparse factorization algorithm based on general row merging. Some implementational details are also described.

In section 5, we compare the performance of the row merging scheme with the George-Heath method. A theoretical analysis on a model problem is provided to illustrate that the general row merging scheme can be substantially better than the original scheme. Experimental results on the two schemes for a collection of practical problems used by George, Heath and Ng [13] are also provided. The use of general row merging is demonstrated to be efficient, effective and highly competitive. Significant reduction in arithmetic operations and factorization time is achieved with only a very modest increase in working storage.

Section 6 contains the conclusions along with some remarks on the relationship between the algorithm developed in this paper with other known approaches. In particular, general row merging is closely related to the multi-frontal method by Duff and Reid [7] for the solution of sparse symmetric systems. The scheme here may be regarded as an implementation of the multi-frontal method on the symmetric decomposition of A^TA based on the matrix A.

2. On general row merging.

2.1. Row merge tree and submatrix merging. We assume that the reader is familiar with the basic operation of a sparse Givens rotation on a pair of matrix rows. After a *plane rotation* on two sparse rows with common first nonzero subscript, the first nonzero of one of them will become zero and the structure of the two transformed rows becomes the *union* of those of the original [4], [10], [19], [20], [27].

It is helpful for our discussion to extend the domain of a plane rotation, which involves only two rows. A *row rotation* has been used by George and Ng [17] to refer to the sequence of plane rotations that will annihilate a given row into an upper triangular factor matrix. Here, we extend this further to *submatrix rotation* or *merging*, which refers to the sequence of row rotations annihilating the rows of a given submatrix one by one into an upper triangular matrix. The end result will be another upper triangular matrix, with possibly different zero-nonzero structure due to contributions from the rows of the first submatrix. It should be clear that the overall sequence of plane rotations will be completely determined by the initial structure of the upper triangular matrix, and by the structure and the row ordering of the submatrix.

In this context, the George-Heath scheme may be regarded as using row rotations, merging each row from the original matrix into the partially formed upper triangular factor matrix, whose final structure has been predetermined. The scheme proposed in this paper using row merge trees can be viewed as using the more general submatrix merging operations. In this paper, the readers are assumed to be familiar with notions related to trees: subtree, forest, parent node, children nodes, leaf nodes, tree traversal (see, for example, [1]).

A *row merge tree* for an m by n matrix A is defined to be a strictly binary tree[1] with m leaves, each corresponding to a row in the matrix. We now describe how the structure of a row merge tree will induce a computational sequence for performing Givens rotations.

Consider any node x in a given row merge tree. It defines a subtree rooted at x, which in turn specifies a set of leaves (or rows) in the subtree. Let us then associate with x an upper triangular matrix obtained by the orthogonal reduction of the corresponding rows in its subtree. Note that the structure of this upper triangular matrix can sometimes be very sparse with possibly many zero rows, and is completely specified by the structure of the rows associated with the leaves in the subtree (irrespective of the order in which the rotations are applied). Moreover, the upper triangular matrix associated with the root of the overall row merge tree will be the factor matrix R to be determined.

The upper triangular matrix associated with an interior node of the row merge tree can be computed simply by a submatrix merge operation on the two submatrices associated with its left and right children nodes, provided that they have been formed. Since the rows in the leaf nodes are already in upper triangular form (with many zero rows), a recursive argument can then be used to compute the triangular matrix for any interior node in the row merge tree, and hence the factor matrix R associated with the root node. Indeed, each interior node in the tree represents the computational task of a submatrix rotation, which merges an upper triangular matrix into another upper triangular matrix.

In other words, the structure of a row merge tree can be used to specify a sequence of $m-1$ submatrix merging operations (since there are $m-1$ interior nodes in a strictly binary tree of m leaves), the result of which is the triangular factor matrix R.

---

[1] A *strictly binary tree* is a binary tree where each interior node has both a left and a right child node.

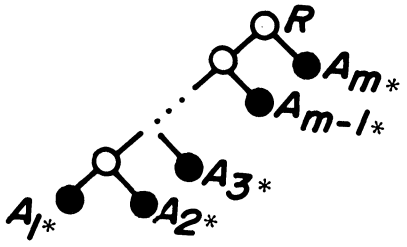


FIG. 2.1. *An unbalanced row merge tree.*

We should point out that the George-Heath scheme, which rotates rows into a fixed triangular matrix, can also be interpreted in the form of a row merge tree. The tree structure is unbalanced, and its general form for a matrix with m rows is illustrated in Figure 2.1. Throughout this paper, we use the notation A_{r*} to represent the r-th row of the matrix A.

	1	2	3	4	5	6	7	8	9
1	x				x		x	x	
2	x				x		x	x	
3	x				x		x	x	
4	x				x		x	x	
5		x			x			x	x
6		x			x			x	x
7		x			x			x	x
8		x			x			x	x
9			x			x	x	x	
10			x			x	x	x	
11			x			x	x	x	
12			x			x	x	x	
13				x		x		x	x
14				x		x		x	x
15				x		x		x	x
16				x		x		x	x

FIG. 2.2. *A* 16 *by* 9 *matrix example.*

2.2. A matrix example. Figure 2.2 contains a 16 by 9 matrix example. This example will be used throughout the paper. In this section, we shall use it to illustrate the different notions presented in section 2.1. A row merge tree is specified for this matrix in Figure 2.3.

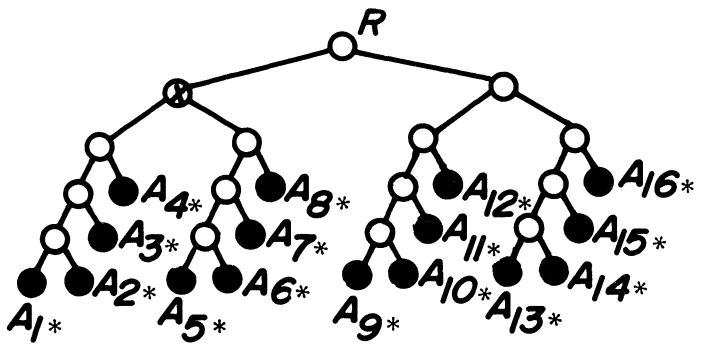
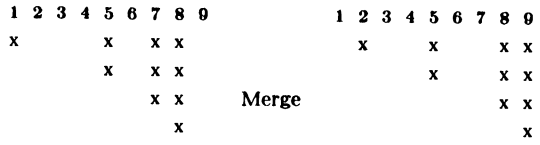
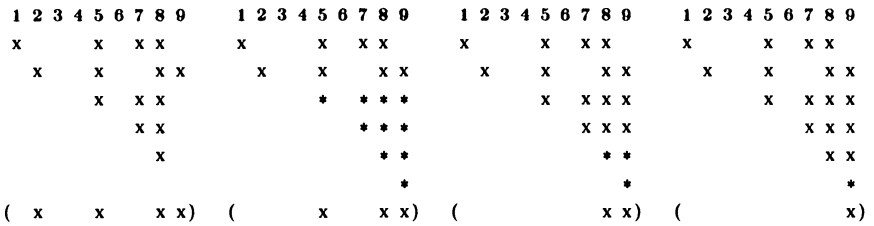


FIG. 2.3. *A row merge tree for matrix in Fig.* 2.2.

Consider the formation of the triangular matrix associated with the node x in Figure 2.3. It can be obtained by the merging of the following two submatrix structures:



It should be clear that these two triangular structures are associated with the left and right children nodes of x respectively. Indeed, the first matrix structure corresponds to the factor resulting from the reduction of the first four rows of the original matrix, the second structure to the next four rows. To aid a better understanding of the matrix merging operation, we display for this example the resulting structures of the row rotations from the four rows of the second matrix into the first. An '*' is used here to indicate those nonzeros actually involved in each row merging operation. The rows being merged are illustrated and they are enclosed in brackets.



Note that there are two rows completely unaffected by the merging. In effect, there are three non-trivial row rotations, and six plane rotations are used to annihilate six nonzeros during the process. One of the annihilated nonzeros is an intermediate fill, which is created from a previous plane rotation. It should be emphasized that the term "*merging*" is used in this paper to refer to the sequence of row rotations, whereby nonzeros are annihilated. In some sense, the two matrices are *assembled* to form one upper triangular matrix through the annihilation process.

It is instructive to compare the use of the unbalanced row merge tree in Figure 2.1 and the merge tree in Figure 2.3 on this example. Figure 2.4 shows three selected stages in the decomposition process when the unbalanced tree (that is, the George-Heath scheme) is used. Here, we use '0' to represent a Givens rotation that reduces a nonzero in the original matrix, an 'i' for a rotation that annihilates an intermediate fill (a nonzero created in the course of computation). As before, an "*" is used to depict a nonzero matrix value involved in the merging operation. For clarity, we have enclosed each submatrix associated with a merging operation in a rectangle. Figure 2.5, on the other hand, illustrates some intermediate stages if the balanced merge tree of Figure 2.3 is used.

The use of the more general merge tree reduces the number of plane rotations from 46 to 42. This can be attributed to the reduction of intermediate fills that need to be annihilated. This example shows that savings in computational cost can be achieved by generalizing row rotations to submatrix merging based on some appropriate row merge trees. However, it is only fair to point out that a reordering of the rows can make the unbalanced merge tree competitive in arithmetic cost for this particular example (reordered in increasing order of last column subscript). This example is selected here because it serves to illustrate other interesting points throughout the

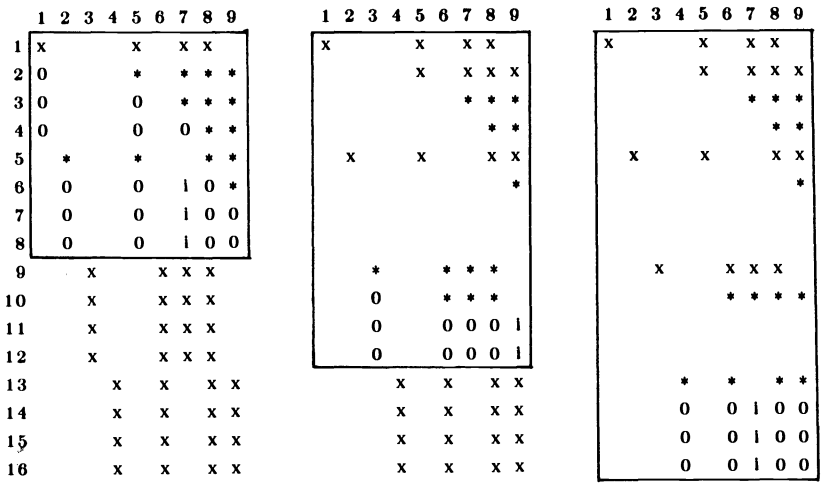


FIG. 2.4. *Merging by the unbalanced merge tree in Fig.* 2.1.

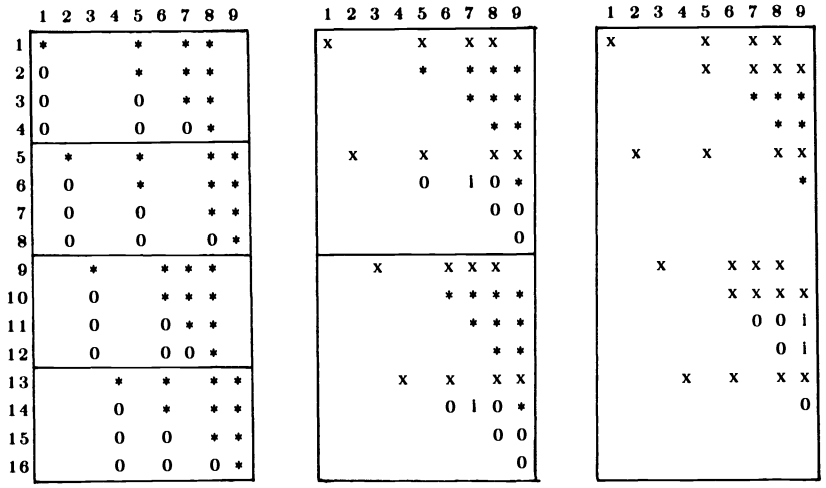


FIG. 2.5. *Merging by the balanced merge tree in Fig.* 2.3.

paper. In section 5, substantial reduction in arithmetic costs will be demonstrated on some practical problems.

2.3. An interpretation: variable row pivoting. The notion of using submatrix merging based on the structure of a row merge tree in sparse orthogonal decomposition is new, but it is related to some existing sparse techniques. We shall discuss its connection with variable row pivoting in this section.

Consider the George-Heath scheme [12]. The rows of the matrix are rotated one by one into the static data structure previously set up for the factor R. Since the pivot row always comes from R, the scheme can be regarded as one using fixed pivots [4], [31]. In view of the choice of pivot row, we shall refer to it as a *diagonal row pivoting* scheme. It is well known in the literature [4], [20], [27] that *any* sequence of plane rotations will produce the same structure for R (with a fixed column ordering). But,

a judicious choice of rotation sequence can sometimes result in drastic reductions in the number of plane rotations and the number of arithmetic operations. Choosing a good row ordering is one such technique for diagonal row pivoting methods [16], [17].

Variable row pivoting is another technique. It was first proposed by Gentleman [10], and later considered in [4], [31]. Unlike the diagonal row pivoting method where the pivot row is predetermined by the location of the nonzero to be annihilated, this method allows a variable choice of pivot rows. The added flexibility can lead to substantial savings in the sparse case. Indeed, Gentleman [10] illustrates impressive savings in the use of variable row pivots for an example arising from the least square analysis of factorial design.

The use of variable row pivoting schemes has not been popular even though the idea was conceived back in 1975. This is primarily due to two reasons: there is no efficient and effective automatic scheme to generate the sequence of variable row pairs for rotations, and the organizational details in performing the numerical computation are apparently complicated. It should be pointed out that Duff [4] suggests a scheme based on some local minimizing criterion to generate a variable row pivoting sequence, but his experimental results indicate that the generated variable sequence does not provide much improvement over the diagonal pivoting scheme.

Now let us consider the submatrix merging technique on a row merge tree as described in section 2.1. Since each submatrix merging is a sequence of row rotations, each of which is a sequence of plane rotations, the overall scheme is a sequence of plane rotations using variable row pivots. So, the general row merging approach can be regarded as a special case of variable row pivoting, and the determining factors are the structure of the row merge tree and the order in which submatrices are to be merged. As we shall see in this paper, the use of submatrix merging helps to resolve the apparent difficulties mentioned above in the variable pivoting approach.

Figure 2.5 shows three different stages in the factorization process for the matrix example in Figure 2.2 using general row merging. That it uses variable row pivots can be readily seen from the matrix sequence. The relationship among the different notions can be summarized as follows:

$$\begin{array}{ccccc} \textit{Row Merge} & & \textit{Sequence of} & & \textit{Sequence of} \\ \textit{Tree} & \rightarrow & \textit{Submatrix Merging} & \rightarrow & \textit{Variable} - \textit{Pivot Rotations} \end{array}$$

3. Generation of row merge trees.

3.1. An algorithm for row merge trees. Naturally, we want to generate row merge trees that will induce submatrix rotation sequences so that the amount of computation in the overall QR decomposition is minimized. In this section, we provide a simple algorithm that will generate effective row merge trees based on the given matrix structure and its column ordering. Though it may not always minimize the number of arithmetic operations, the scheme works extremely well in practice. This is not surprising as we shall see that it is motivated from some well-understood observations in sparse Cholesky factorization.

Let the column ordering of the given m by n matrix A be fixed. We first describe the algorithm in terms of (triangular) submatrix merges. Throughout the course of the algorithm, a set Γ of sparse upper triangular matrices will be maintained. Consider a row A_{r*} with the first nonzero in column f. It is associated with an upper triangular matrix with all zero rows except for row f, which is the same as A_{r*}. Let Γ be initialized with the set of these m triangular matrices.

The algorithm can then be described as follows:

```
for column c := 1 to n
begin
    Merge all triangular matrices in Γ with nonzero column c;
    Replace these matrices in Γ by the resulting triangular matrix;
end;
```

At the end of the above algorithm, the upper triangular matrix factor R can be obtained directly from the set Γ. On the surface, this approach may look similar to the diagonal row pivoting scheme, with the rows arranged in ascending order of the column subscript of the first nonzero. The merging sequences in Figure 2.4 and Figure 2.5 should illustrate that they are indeed different. The merging sequence in Figure 2.5 is using the above algorithm. The first matrix pattern shows the execution corresponding to loop index $c = 1, 2, 3, 4$; after which the set Γ will contain four upper triangular matrices. The second pattern illustrates the merging of matrices involving columns 5 and 6 respectively, so that two matrices will be left in the set Γ.

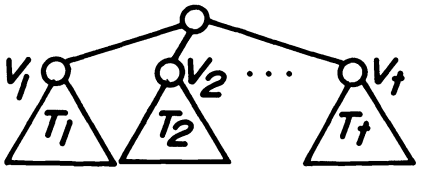
The generation of the merge tree structure corresponding to the above matrix merging sequence is quite straightforward. The structure can be determined by working logically on the column subscripts. For completeness and for later reference, we describe it as follows.

Let $x_1, x_2, \ldots, x_n$ denote the columns of A, and for each row A_{r*}, let q_r be the set of nonzero column subscripts in the r-th row, that is

$$q_r = \{x_c \mid a_{rc} \neq 0\}.$$

The following algorithm will generate a row merge tree structure (not necessarily binary!) from the set of rows $\{q_r\}$. It maintains a forest F containing a set of *labelled* trees (rather than a set of triangular matrices). The label of each tree node is a subset of the column set $\{x_1, x_2, \ldots, x_n\}$. The forest F is initialized to contain m labelled trees, each having one single node labelled by q_r, $r = 1, \ldots, m$ respectively.

```
for c := 1 to n
begin
    Find all trees in F, whose root includes x_c in its set label;
    Replace these trees, say, T_1, ..., T_t having set
        labels V_1, ..., V_t in their roots, by a new labelled tree:
```



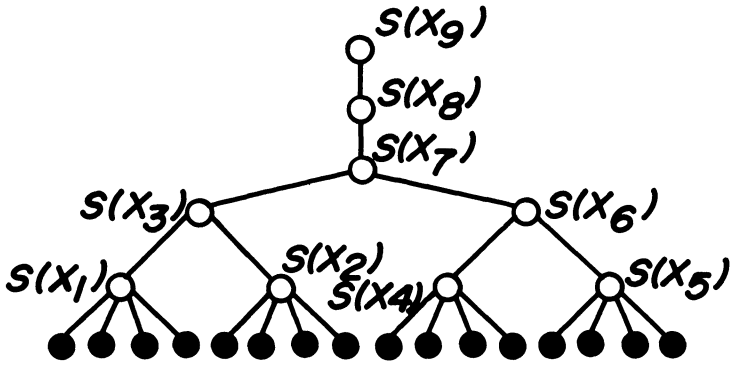

```
    Label the new root by the column subset (V_1 ∪ ... ∪ V_t)
end;
```

In general, at the end, the forest F may still have more than one tree. That will correspond to a structurally reducible A^TA. For simplicity, we assume that F is a tree at the termination of this algorithm. The results in this section can be extended quite easily to the general case.

Note that the tree generation algorithm is column driven, and the structure of the merge tree depends on the column order $x_1, \ldots, x_n$ and the row structures. The merge

tree formed has m leaves, each labelled by a column subscript set of a row of A. There are n interior nodes, each of which can be associated with a column of A. If an interior node is associated with the column x_c, we shall use $S(x_c)$ to denote its *set label.* Each is formed by simply taking the union of the labels of its children nodes. Their significance in the computational process will be considered later in the section.

It should be noted that the tree obtained from the algorithm may not always be a binary tree (since in general, $n \neq m-1$). For example, the application of this algorithm to the matrix structure in Figure 2.2 will give the following merge tree:



So, strictly speaking, it is not a row merge tree. However, it can be transformed into a binary tree by introducing additional interior nodes (binary splitting) if a node has more than two children, and by removing those parent nodes that have only one child. For example, the row merge tree used in Figure 2.3 can be regarded as one such transformed binary tree for the above tree structure.

Needless to say, there are a number of different ways to perform binary-splitting. To find the best possible splitting in the context of sparse QR decomposition is an interesting problem. There are matrix examples in which an effective binary-splitting transformation is of great importance to the overall efficiency of the method (e.g. the one used by Gentleman in [10] to demonstrate savings by using variable pivots).

As we shall see in section 4, a simple binary-splitting method is used in this paper. It will be shown to give an effective overall scheme for the practical problems tested, when compared with the George-Heath method. More sophisticated splitting strategies based on some local minimization criterion (e.g. [4], [31]) should be promising. More will be said about this in section 4.

3.2. Properties of row merge trees generated. In this section, we study some properties of the row merge tree generated by the algorithm in the previous subsection. We shall establish the important observation that the actual numerical computation based on such row merge trees can be organized as a sequence of (upper triangular) *full* submatrix merging operations.

Recall that the structure generated is a tree with m leaves and n interior nodes. Each leaf node corresponds to a row in the original matrix, and each interior node to a column. Moreover, in the course of the algorithm, each interior node is assigned a label $S(x_c)$, which is a subset of column subscripts. For convenience in the discussion, we shall use $S(x_c)$ to refer to the node with this label in the tree.

Consider the subtree rooted at the node $S(x_c)$. Let the set labels of the children of this node be

$$V_1, V_2, \ldots, V_t.$$

Consider the matrix merging operation associated with the node $S(x_c)$. We assume that at this stage the submatrices associated with its children nodes have already been computed. The following observations are quite obvious, and we list them without any proof.

Observation 3.1. For $i \neq j$, $(V_i \cap V_j) \cap \{x_1, \ldots, x_{c-1}\} = \varnothing$.

Observation 3.2. $c = \min \{k \mid x_k \in V_1 \cap \ldots \cap V_t\}$.

Observation 3.3. In the submatrices of the children nodes, if the first nonzero of a row is x_f, with $f < c$, then this row is the computed row of R_{f*} in the factor matrix R.

Observation 3.4. The matrix merging operation associated with the node $S(x_c)$ only involves columns in the set

$$S(x_c) \cap \{x_c, \ldots, x_n\}.$$

In other words, for each child V_i, only the part of the submatrix with columns $V_i \cap \{x_c, \ldots, x_n\}$ is involved in this merging operation.

Observation 3.5. The subscript set $S(x_c) \cap \{x_c, \ldots, x_n\}$ corresponds to the nonzero structure of the c-th row of the (upper triangular) Cholesky factor obtained by the symbolic factorization on the structure of A^TA.

It is appropriate here to point out that in general, the Cholesky structure obtained by the symbolic factorization of A^TA may overestimate the structure for the orthogonal factor R [3] (though, mathematically they are the same). However, in this paper, we shall assume that the two structures are the same. It is reasonable, since any (sparse) matrix can be reordered to a block upper triangular form where each block diagonal submatrix has this property. Readers are referred to the paper by Coleman et al. [3] for details. With this assumption, the subscript set in Observation 3.5 now corresponds to the structure of the row R_{c*} in the factor R.

The structure of the row merge tree formed can be characterized simply by the following two observations.

Observation 3.6. The node $S(x_f)$ is the parent node of a leaf node q_r if and only if

$$f = \min \{k \mid x_k \in q_r\}.$$

Observation 3.7. The node $S(x_f)$ is the parent node of an interior node $S(x_c)$ if and only if

$$f = \min \{k \mid x_k \in S(x_c) - \{x_1, \ldots, x_c\}\}.$$

To aid our further discussion, we introduce some new terminology. Let T be a sparse upper triangular matrix. If *both* the i-th row and the i-th column of T are zeros, the removal of them will result in a smaller upper triangular matrix. Consider the removal of *all* such zero row/column pairs; if the resulting matrix is full upper triangular, then T is said to be *essentially full.* If the remaining matrix is full upper trapezoidal, then T is an *essentially full upper trapezoidal* matrix.

Such a matrix structure can be characterized by the number v of non-null rows, and the set V of subscripts of non-null columns. We shall use the notation $TZ(v, V)$ to denote the structure of such an essentially full upper trapezoidal matrix. In the notation, for convenience we further assume that the set V is represented as an array $V[*]$ containing the subscripts in ascending order, that is, $V[k] < V[k+1]$, for $k = 1, \ldots, v-1$.

In the merging of two essentially full upper trapezoidal matrix, if every (non-null) row is involved in a non-trivial rotation, then the resulting upper trapezoidal matrix is again essentially full. The next lemma provides a simple sufficient condition.

LEMMA 3.8. *The merging of* $TZ(u,U)$ *and* $TZ(v,V)$, *with*

$$U[1] = V[1]$$

gives the structure $TZ(w,W)$, *with*

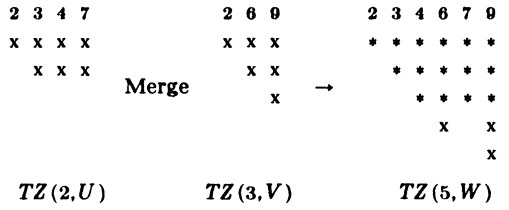
$$w = \min\{u+v, |W|\}, \qquad W = U \cup V,$$

provided that

$$u+v > \max\{u', v'\}$$

where $U[u] = W[u']$, *and* $V[v] = W[v']$. (*That is,* u' $\{v'\}$ *is the location of the subscript* $U[u]$ $\{V[v]\}$ *in the array* $W[*]$.)

The following example shows that the merging of two essentially full upper trapezoidal matrices may not always give rise to one that is essentially full.



Note that for this example, the condition in Lemma 3.8 is not satisfied: u' is 2 since $U[u] = 3$ appears second in the W-sequence, while v' is 6 since $V[v] = 9$ is the sixth subscript in W, so that $\max\{u', v'\} = 6$ and $u+v = 5$.

Even though, in general, the resulting matrix from the merging of two essentially full upper trapezoidal matrices is not essentially full, the condition in Lemma 3.8 holds when there are more rows than columns. In particular, it holds for upper triangular matrices, so that the merging of essentially full upper triangular matrices always gives an essentially full upper triangular matrix (since $\max\{u', v'\} = w < u+v$). For all practical purposes, we can assume that the resulting upper trapezoidal matrix after the merging operation is essentially full. Indeed, in instances where it is not, the bookkeeping overhead incurred will offset the advantage gained in treating the resulting upper trapezoidal matrix as sparse.

The result of Lemma 3.8 can be applied in the context of submatrix merging based on the row merge tree generated from the last subsection. The submatrix associated with each leaf in the merge tree corresponds to a row in the original matrix, which is obviously an essentially full trapezoidal matrix. By Lemma 3.8, the merging of two rows with common first subscript always generate an essentially full upper trapezoidal matrix (since $u+v = 2$ and $u' = v' = 1$).

In general, consider any interior node $S(x_c)$ and the submatrices associated with its children nodes with set labels $V_1, \ldots, V_t$. By Observations 3.2–3.5, we can remove rows involving columns before x_c from these matrices and then merge the remaining portions of the submatrices together. If these remaining submatrices are essentially full, that is,

$$TZ(u_1, V_1 \cap \{x_c, \ldots, x_n\}) \ldots TZ(u_t, V_t \cap \{x_c, \ldots, x_n\})$$

where $u_1, \ldots, u_t$ are the respective number of rows, then based on the remark after Lemma 3.8, from a practical standpoint, we can treat the result of the merging as another essentially full upper trapezoidal matrix: $TZ(u, S(x_c) \cap \{x_c, \ldots, x_n\})$.

In summary, in the numerical QR factorization of an m by n sparse matrix, we can first determine a row merge tree using the algorithm in section 3.1. Then, using its structure, we can organize the computation as $m-1$ major steps, each being a

submatrix rotation of two essentially full upper trapezoidal matrices to yield another matrix with the same property.

3.3. Graph theoretic interpretation and motivation. In this section, a graph theoretic interpretation of general row merging in QR decomposition will be presented. This helps to provide the motivation for the row merge tree generation algorithm described earlier. This section can be skipped without breaking the continuity of the paper.

Readers are assumed to be familiar with graph theoretic terminology associated with sparse matrix computation. In particular, the concepts of undirected graphs associated with symmetric matrices, nodes, edges, adjacent sets, cliques, and reachable sets are assumed. Readers are referred to [15] for formal definitions.

Consider the orthogonal decomposition of the given sparse matrix A. As noted in the introduction, the triangular factor matrix R corresponds to the Cholesky factor of the symmetric matrix $M = A^TA$, so that its structure can be obtained by the symbolic factorization of the symmetric graph $G(M)$ using the *elimination graph model* [29].

The determination of the elimination graph sequence for a symmetric graph is well known [15], [29]. When a node y in the symmetric graph $G(M)$ is eliminated from the graph, the set of adjacent nodes of y becomes a clique (complete subgraph) in the new elimination graph. In other words, all the cliques containing the node y will be merged to form a new clique after the elimination of y. This process of *clique merging* is the basis in the study of Gaussian elimination for symmetric positive definite systems.

To make the connection between the matrix structure of A and the graph structure of $G(M)$, we note that each row in the matrix A can be identified as a clique in the symmetric graph $G(M)$ [17]. This is because each row A_{r*} is used in an outer-product $(A_{r*})^T(A_{r*})$ in forming M. We can therefore say that each row of A is a representation of a clique in $G(M)$.

Since the node elimination sequence of the graph $G(M)$ provides a natural way of specifying a clique merging sequence (see [8] for a discussion of *element merge tree*), and each row in A corresponds to a clique in $G(M)$, the row merge tree generated by the algorithm in the previous subsection corresponds to the clique merge sequence as defined by the node elimination sequence. This observation can be best illustrated by an example. Consider the matrix structure (same as that of Figure 2.2) and its corresponding symmetric graph in Figure 3.1.

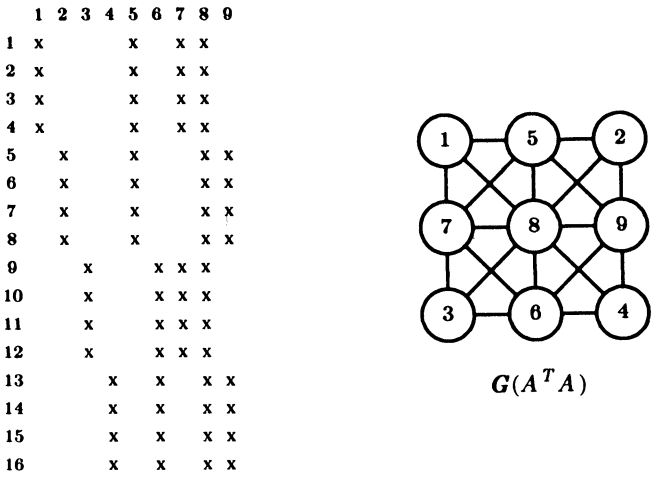


FIG. 3.1. *Matrix structure A and its symmetric graph* (A^TA).

Node Eliminated	Cliques Merged	Clique Formed
x_1	$\{x_1,x_5,x_7,x_8\}$	$\{x_1,x_5,x_7,x_8\}$
x_2	$\{x_2,x_5,x_8,x_9\}$	$\{x_2,x_5,x_8,x_9\}$
x_3	$\{x_3,x_6,x_7,x_8\}$	$\{x_3,x_6,x_7,x_8\}$
x_4	$\{x_4,x_6,x_8,x_9\}$	$\{x_4,x_6,x_8,x_9\}$
x_5	$\{x_5,x_7,x_8\}$, $\{x_5,x_8,x_9\}$	$\{x_5,x_7,x_8,x_9\}$
x_6	$\{x_6,x_7,x_8\}$, $\{x_6,x_8,x_9\}$	$\{x_6,x_7,x_8,x_9\}$
$\{x_7,x_8,x_9\}$	$\{x_7,x_8,x_9\}$, $\{x_7,x_8,x_9\}$	$\{x_7,x_8,x_9\}$

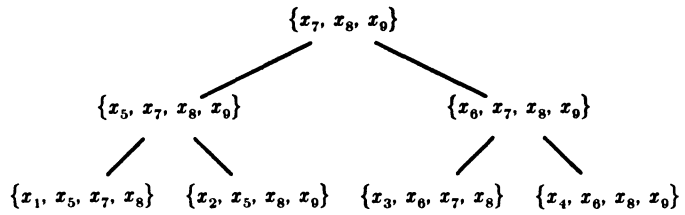


FIG. 3.2. *Clique merging in node elimination.*

Figure 3.2 contains the node elimination sequence and the clique merging operations. It also displays the clique merges in the form of a tree. Its correspondence with the row merge tree in Figure 2.3 should be clear. In the elimination of the node x_c, the resulting clique formed after merging corresponds to the subscript set

$$S\{x_c\} \cap (x_c, \ldots, x_n\}$$

which is associated with the nonzero structure of the row R_{c*} in the factor matrix R.

The success of the row merge tree approach in sparse QR decomposition depends on the choice of the column ordering. Following George and Heath [12], we choose the column ordering to minimize fill in the symmetric decomposition of $M = A^TA$; the practical ones are the *minimum degree* ordering and *nested-dissection* type orderings [15]. In these ordering schemes, at each step, a node is picked for elimination if the resulting clique size is small. In the context of (essentially full) submatrix merging, the clique size corresponds to the size of the essentially full trapezoidal matrix. By keeping this size small, this will help to reduce the number of intermediate fills and the amount of computation required to perform the merging operation. This explains why such ordering schemes are appropriate column orderings for the general row merging scheme.

In the study of symmetric Cholesky factorization, a tree structure, called an *elimination tree*, has been used in many different contexts: the multi-frontal method [7], [28], parallel elimination [21], compact row storage scheme [24], and relative row-index storage scheme [30]. Without going into the details, we merely state that it follows quite readily from Observation 3.7 that the row merge tree constructed by the algorithm in section 3.1 with the leaf nodes removed is the elimination tree for the symmetric matrix A^TA. This connection is useful from a practical view, since there is already an efficient algorithm to generate the elimination tree from the structure of a symmetric matrix [24]. The row merge tree can thus be determined by forming the elimination tree of A^TA.

4. An overall sparse orthogonal factorization scheme. In this section, we put all the observations on row merging into the overall perspective of a scheme for performing sparse QR decomposition.

In the row merge tree, the numerical computation of matrix merging associated with an interior node can be performed so long as the submatrices associated with its children nodes have already been computed. In other words, if we view the computational sequence in terms of a tree traversal sequence (see e.g. [1]), the children nodes have to be visited before the parent node. Following the *multi-frontal* method by Duff and Reid [7] for symmetric decomposition, we shall visit the nodes in the merge tree systematically in a *depth-first order* (that is, a *postorder* traversal of the tree [1]). This will enable us to manage the numerical merging by means of a stack of upper trapezoidal essentially full matrices. The stack is used to store all the partially formed trapezoidal matrices, and the next one required for submatrix merging is always at the top of the stack. Indeed, in the computational scheme, we either merge an incoming row from the original matrix or a trapezoidal matrix from the top of the stack into a working upper triangular matrix.

Since there is a one-to-one correspondence between columns of the matrix and interior nodes in the row merge tree, a postordering on the merge tree, in effect, corresponds to a column reordering of the original matrix A. It should be noted that this reordering is equivalent to the previous column ordering in terms of fills in the factor matrix.

The overall scheme is described as follows:

Step 1: (*Column Ordering*)
Find a fill-reducing ordering P_c (e.g. the minimum degree ordering) for the structure of A^TA. Obtain the structure of AP_c using P_c as the column ordering.

Step 2: (*Row Merge Tree*)
Generate the (not necessarily binary) row merge tree based on the structure of AP_c by the column-driven algorithm in section 3.1.

Step 3: (*Column Reordering*)
Generate a postordering of the row merge tree (by a depth-first search). Let $\bar{P}_c$ be the corresponding equivalent column ordering of A. Order the columns of A using $\bar{P}_c$.

Step 4: (*Symbolic Factorization and Storage Allocation*)
Perform the symbolic factorization on the symmetric structure of $(A\bar{P}_c)^T(A\bar{P}_c)$, and obtain the structure of the corresponding factor R. Create a static storage for the rows of R based on the information computed.

Step 5: (*Row Ordering*)
Sort the rows of $A\bar{P}_c$ in *ascending* order of its first column subscript; let the corresponding row ordering be P_r.

Step 6: (*Numeric Factorization*)

```
Initialize a stack of essentially full upper trapezoidal matrices;
for i := 1 to n
begin
  From the structure of R, let i, i_1, i_2, ..., i_t
    be the locations of the nonzeros in row R_{i*};
  Obtain working space for a full t+1 by t+1 upper triangular matrix;
  If the top upper trapezoidal matrix on the stack has i as its first subscript
    then Pop it from the stack and merge it into the working triangular matrix;
  While the next row from P_r A P̄_c has i as its first subscript
    do Merge the row into the working triangular matrix;
  Save the first row of working triangular matrix as the row R_{i*}
    into the static storage set up for R;
```

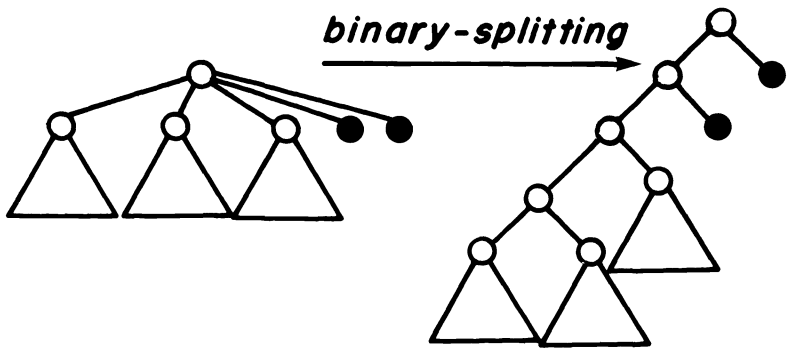
Consider the remaining non-null rows of working matrix as one upper trapezoidal matrix with the subscript set: $i_1, \ldots, i_t$;

If the top upper triangular matrix on the stack has i_1 as its first subscript
then Merge the remaining working matrix into this top matrix
else Push the remaining working matrix onto the stack;

end;

In step 2, the row merge tree can be generated by forming the elimination tree of the symmetric matrix A^TA, as described in section 3. The elimination tree generation algorithm in [24] can be used.

Step 6 describes the main steps in the overall numerical computational phase. Some minor details are omitted for clarity; for example, more than one row from the working triangular matrix may be fully formed, so that they can be saved simultaneously before the remaining rows are pushed onto the stack. The algorithm as described, uses a specific binary-splitting method for parent nodes with more than two children nodes. It should be clear that the splitting strategy can be depicted by the following example:



Here, the shaded nodes represent rows from the original matrix. This splitting has the advantage of low non-numeric computational overhead, simple implementation, and relatively small stack storage requirement.

The author has experimented with other splitting methods. In the problems tested, there is little or no improvement in the overall execution time, although in some cases, the operation counts are noticeably reduced. This can be attributed to the extra bookkeeping overhead for the more complicated strategies, which offsets the reduction in arithmetic operations. Therefore, in the implementation the above simple splitting strategy is used, and it will be shown to be quite effective in the experiments reported in section 5. The author is still looking for other practical ways of finding good and reliable splitting transformations based on the row structures.

In the above algorithm, we assume that the factor matrix R is stored row by row in a compact form. Moreover, the structure of R is represented using the compressed subscript scheme. We assume that the reader is familiar with these representations and is referred to [15] for details.

It should be noted that in the numerical factorization phase, the temporary storage required for the working upper triangular matrix in the numeric factorization phase can overlap with the storage for the part of R remaining to be computed. There is always enough room in this part of R for the current working matrix. However, additional storage is needed for the stack of upper trapezoidal full matrices. The amount required can be determined symbolically based on the structure of the matrix

$P_r A \bar{P}_c$. In this way, stack space can be preallocated in step 5 before numerical computation begins.

Each submatrix in the stack can be specified by its number of rows and its column subscript set. Each such column subscript set can be identified as a row structure in R. This piece of information can, therefore, be obtained from the structural representation of R, and it is not necessary to store it explicitly. Furthermore, the sizes of these full trapezoidal submatrices are given by the number of nonzeros in the rows of the factor matrix R (see Observation 3.5).

5. Comparison of general row merging with the George-Heath scheme.

5.1. Theoretical comparison on a model problem. In this section, the general row merging strategy is analyzed when applied to a model problem. This problem arises typically in the natural factor formulation of finite element methods [2]. It is quite suited to the new approach, and we use it to demonstrate the possible substantial savings in arithmetic cost.

Consider the k by k regular grid with $(k-1)^2$ small squares. Associated with each of the k^2 grid nodes is a variable. Associated with each square (called an *element*) is a set of s equations involving the four variables at the corners of the square. The assembly of these equations results in a large sparse overdetermined system of equations

$$Ax = b$$

where the dimension of the matrix A is m by n, with

$$m = s(k-1)^2, \qquad n = k^2.$$

Note that the regular grid reflects the structure of the symmetric matrix $A^T A$. Each row in the matrix A can be identified with an element in the grid. The example in Figure 3.1 is the 3 by 3 model problem.

For the analysis, we assume that the columns of the matrix A are numbered by the nested dissection ordering [11]. Based on this dissection column ordering (that is, variable ordering in the regular grid), we use the general row merging algorithm as described in section 4. We further assume that the s equations associated with each element are initially reduced to an upper triangular submatrix. If $s \geqq 4$, each requires $4s-10$ rotations; in total, this initial phase takes $(4s-10)(k-1)^2$ rotations. As we shall see, this step does not affect the order of magnitude of the counts in the overall scheme.

Let $G(k)$ be the number of Givens rotations required to perform the orthogonal decomposition of the matrix A using the general row merging algorithm. Furthermore, let $\theta(k)$ be the corresponding cost. In the analysis, following [17], we use the number of nonzeros in the transformed pivot row as the cost to perform a sparse plane rotation. Note that the actual number of multiplicative operations is usually a multiple of this cost. These quantities can be computed by solving a set of recurrence relations. Since the derivation is quite straightforward, we shall omit the details. The results are stated in the following theorems, and readers are referred to [22] for the analysis.

THEOREM 5.1.

$$G(k) = \frac{31}{4} k^2 \log_2 k + 4sk^2 + O(k^2).$$

THEOREM 5.2.

$$\theta(k) = \frac{829}{84} k^3 + 10sk^2 + O(k^2 \log_2 k).$$

It is interesting to compare the result with that of the George-Heath method, which uses the diagonal row pivoting approach. In [26], Ng shows that the diagonal pivoting scheme on the model problem requires no more than

$$\frac{1419}{224}sk^3 + O(k^2\log_2 k)$$

number of operations. Note that in this case, the high order term depends on the value of the parameter s. For $s=4$ (typical in finite element application), the constants of proportionality for diagonal row pivoting and general row merging are 25.33 and 9.87 respectively. For $s>4$, the difference is more dramatic.

To demonstrate the actual difference, we count the exact number of Givens transformations and the exact cost to do the sparse QR factorization. Table 5.1 contains these counts for the George-Heath scheme (diagonal row pivoting) and the general row merging method on the model problem with $s=4$. The matrix column ordering is obtained by the theoretical nested dissection scheme as described by George [11].

TABLE 5.1
Theoretical results on the k by k grid problem, with nested dissection column ordering.

k	Diagonal Row Pivoting				General Row Merging			
	Rotation	$/k^2\log k$	Cost	$/k^3$	Rotation	$/k^2\log k$	Cost	$/k^3$
20	20282	11.7	201057	25.1	12076	7.0	80440	10.06
30	58802	13.3	781082	28.9	31625	7.2	273504	10.13
40	121690	14.3	1944957	30.4	61502	7.2	636269	9.94
50	212698	15.1	4003334	32.0	102275	7.3	1243867	9.95
60	332154	15.6	7117352	33.0	154437	7.3	2137404	9.90

We observe from the Table 5.1 that the theoretical savings in rotations and operations are quite substantial when the general row merging scheme is used. In the 60 by 60 grid problem, the diagonal scheme uses more than *twice* the number of rotations, and more than *three* times the arithmetic cost required by the new scheme. We also note that for the k by k model problem, the working stack storage requirement is of the order of k^2.

5.2. Numerical experiments on some practical problems. The algorithm as described in section 4 has been implemented and in this section, we provide numerical results from experimental runs of the code. We compare the performance of this code with the implementation of the George-Heath scheme by George and Ng [18] in the SPARSPAK-B package for solving linear least squares problems.

For both schemes, the *minimum degree ordering* is used for the matrix column ordering. It is the most popular general-purpose fill-reducing ordering strategy for symmetric matrices. The recent modified version of the minimum degree algorithm in [23] is used in our experiments.

For row ordering, in the row merging scheme, the rows are arranged in increasing order of the *first* nonzero column subscript, as described in the algorithm of section 4. On the other hand, in the George-Heath scheme, the rows of the matrix are ordered in increasing sequence of *last* nonzero column subscript. This row ordering strategy has been recommended by George et al. [16] for the diagonal pivoting scheme when the columns are arranged by the minimum degree ordering.

In the results tabulated, only multiplicative operations are accounted for in the operation counts, and the times reported are in CPU seconds of an VAX 11/780 with a floating point accelerator. Single precisions are used in the experiments. There are two sets of test examples. The first set consists of the model problem with different values of k. The numerical results are tabulated in Table 5.2. They show that the savings in computation are realized in computer implementations. Indeed, for $k = 50$, the George-Heath scheme, compared to the new row merging approach, requires *four* times more CPU time and *seven* times as many arithmetic operations.

It should be noted that the operation counts in Table 5.2 include some trivial multiplicative operations in the actual orthogonal factorization. These operations can be avoided by testing for zero operands. However, there would be little or no improvement in CPU time. The reason is that we are only trading trivial multiplicative operations for logic in zero operand detections, and the amount of other non-numeric overhead stays the same (George and Ng, private communication).

TABLE 5.2
QR factorization statistics for model problems with minimum degree column ordering.

k	SPARSPAK-B		General Row Merging	
	Factor Opcount	Factor Time	Factor Opcount	Factor Time
10	160352	2.33	50824	1.80
20	2197730	24.78	418484	9.97
30	7851479	84.34	1320864	26.73
40	20934178	219.26	3157880	55.01
50	42867428	432.72	6106096	97.13

The second test set consists of the ten problems used by George, Heath and Ng in their comparison paper on methods for solving sparse linear least squares systems [13]. They represent a wide variety of structures arising from practical applications; readers are referred to the comparison paper for details about the problems. We list them in Table 5.3 for reference.

TABLE 5.3
Matrix problems for the second test set.

Problem	Number of Rows	Cols	Nonz	Problem Description
1	313	176	1557	Sudan survey data
2	1033	320	4732	Analysis of gravity-meter observations (well-conditioned)
3	1033	320	4719	Analysis of gravity-meter observations (ill-conditioned)
4	1850	712	8755	Similar to Problem 2, but larger
5	1850	712	8638	Similar to Problem 4, but larger
6	784	225	3136	15×15 grid problem
7	1444	400	5776	20×20 grid problem
8	1512	402	7152	3×3 geodetic network with 2 observations per node
9	1488	784	7040	4×4 geodetic network with 1 observation per node
10	900	269	4208	Geodetic network problem provided by U.S. National Geodetic Survey (ill-conditioned)

In Table 5.4, we tabulate the statistics for the orthogonal factorization by the two methods on the ten test problems. "Column Order Time" for SPARSPAK-B is the time used to generate the minimum degree ordering for A^TA; for general row merging, it includes the time for the minimum degree column ordering, the row merge tree generation and the postordering of the merge tree. From the results tabulated, the time used for the extra processing is not very significant.

TABLE 5.4
Factorization statistics on ten test problems.

Problem	SPARSPAK-B			General Row Merging		
	Column Order Time	QR Fact Opcount	QR Fact Time	Column Order Time	QR Fact Opcount	QR Fact Time
1	0.43	98289	1.76	0.59	56004	1.93
2	3.15	412132	6.44	3.52	234056	6.47
3	3.24	419323	6.64	3.58	236580	6.35
4	5.79	2955637	34.03	6.47	754444	15.62
5	5.86	2983343	33.74	6.64	755724	15.24
6	0.34	750547	9.22	0.51	167004	4.97
7	0.61	2197730	24.94	0.90	418444	10.09
8	0.42	723103	11.12	0.67	361340	9.61
9	0.80	760795	11.38	1.35	395088	10.30
10	3.08	1228537	14.18	3.36	710532	11.02

For all ten problems, there is an impressive reduction in arithmetic operation counts. Although the reduction in execution time is not commensurate with the reduction in operation counts due to non-numeric overhead, the saving in time is still substantial. For problems 4 and 5, the CPU time is reduced by more than 50%.

The reduction in execution time is achieved at the expense of an increase in working storage to manipulate a stack of full upper triangular matrices. In Table 5.5, we tabulate the size of the stack required to perform the numerical factorization for these ten problems. "Maximum Stack Size" is the maximum number of real values ever stored at one time in the stack during the course of the numerical computation, and "Max No. of Stack Entries" refers to the maximum number of triangular matrices ever pushed onto the stack. The additional storage required by row merging is given

TABLE 5.5
Working storage requirement for general row merging.

Problem	No. of Nonz(A)	No. of Nonz(R)	No. of Subscript(R)	Maximum Stack Size	Max No. of Stack Entries
1	1557	1627	499	213	4
2	4732	2575	901	428	4
3	4719	2573	915	428	4
4	8755	7410	3261	1341	7
5	8638	7415	3267	1341	7
6	3136	2786	1129	982	6
7	5776	6118	2171	2322	6
8	7152	4091	1268	611	6
9	7040	8300	2666	845	6
10	4208	4697	1867	953	4

by the sum of "Maximum Stack Size" and three times "Max No. of Stack Entries". These numbers are rather modest when compared to the number of nonzeros in the original matrix A or the orthogonal factor matrix R. For completeness, we have also included the number of compressed subscripts required to represent the structure of R in the table under the heading "No. of Subscript(R)".

We have not included the time to do the symbolic factorization, the row ordering, and the solution back-substitution in the table. The respective times required by the two schemes are roughly the same, and they are insignificant when compared with the numerical factorization time.

6. Concluding remarks. In this paper, we have introduced general row merging schemes for the sparse QR decomposition using Givens rotations. The merging sequence can be represented conveniently in the form of a tree, which we have called a row merge tree. An algorithm is provided which will automatically select the rows to be merged, based on the column ordering and the structure of the given sparse matrix. The tree constructed is closely related to the element merge tree and the elimination tree used in the study of sparse Cholesky factorization. This approach of general row merging can also be interpreted as a special type of variable row pivoting (as opposed to diagonal row pivoting).

The advantages of this approach are two-fold. Firstly, it can be used to reduce the computational cost in doing the sparse QR factorization. An analysis of the scheme on the k by k model problem shows that it is significantly better than diagonal row pivoting schemes, which includes the recent George-Heath method. Secondly, we have shown that the computation using this approach can be organized into a sequence of reductions of two upper trapezoidal full submatrices. This has some significant practical implications in the implementation of this scheme. Experimental results on practical problems show that substantial savings in operations and CPU time can be achieved. It should also be noted that fast Givens [9] can be used in the matrix merging operations in the implementation.

Finally, we provide some remarks on the scheme developed in this paper with other known approaches.The recent work by George, Heath and Plemmons [14] on out-of-core solution of large sparse least squares problem can be interpreted in the context of a row merge tree. Our work here shows that the technique can be exploited even in the in-core solution.The work in [25] by Manneback, Murigande and Toint on the solution of large sparse least square problems arising from geodetic Doppler multi-station adjustment using elimination of local variables at data collection centres is related to general row merging in a similar way. It can be regarded as doing the computation associated with some subtrees of the row merge tree in a distributive computing environment.

The recent practical scheme by George and Heath is simple and efficient in terms of data and storage management, and adapts extremely well to row-by-row processing. The scheme proposed in this paper by general row merging may be considered as generalizing row rotations to submatrix rotations/merging in their method. The resulting scheme compares very favorably with the George-Heath method. Indeed, our implementation generally requires less time to perform the sparse QR decomposition on all the problems tested, with a very modest increase in working storage. On some problems, the CPU time reduction is substantial.

General row merging can also be viewed as a special type of variable row pivoting method. Work has been done by Duff [4], and Zlatev [31] to study the use of variable pivots. They have devised column and row ordering strategies appropriate for variable pivot selections, and dynamic storage methods to implement the numerical factoriz-

ation. The method proposed in this paper differs in several aspects. Our choice of column and row orderings (and hence the variable pivots) is modeled from the structure of A^TA, while they work directly on the structure of A. In view of the relation between orthogonal and Cholesky factors, our choice generally produces good orderings and effective variable row-pairs for rotations. Secondly, we have organized the computation as a sequence of full matrix rotations (merging), which can be managed orderly by means of a stack. Thirdly, some of the intermediate fills created in a submatrix merge will be annihilated *during* this merging operation, so that they need not be kept. The rest will be pushed as a full trapezoidal matrix onto the stack. Experiments in section 5 indicate that the amount of stack storage required is quite small. The dynamic storage methods proposed in [4], [31] apparently do not have these features.

Another related work is the multi-frontal method by Duff and Reid [7] on the solution of sparse indefinite systems. The execution sequence determined by the row merge tree in this paper can be viewed as performing the computation on multiple *fronts*. Indeed, general row merging may be regarded as an implementation of the multi-frontal method on the sysmmetric decomposition A^TA based on the matrix A. Our scheme for sparse orthogonal decomposition and the TREESOLVE package by Reid [28] for sparse Cholesky factorization of large finite element systems use similar techniques on two different classes of problems. It is interesting to point out that in the multi-frontal method [5], [7], the basic operation is the factorization of a full submatrix, and for the scheme in this paper, it is the merging of two full submatrices.

Acknowledgments. The author is happy to express his gratitude to Alan George and Esmond Ng for kindly providing him with a version of the SPARSPAK-B package and their collection of test problems, and for their many helpful discussions on the manuscript. He would also like to thank the referees whose excellent suggestions and constructive criticisms have greatly improved the overall presentation of the material in this paper.

REFERENCES

[1] A. V. AHO, J. E. HOPCROFT AND J. D. ULLMAN, *Data Structures and Algorithms*, Addison-Wesley, Reading, MA, 1983.

[2] J. H. ARGYRIS AND O. E. BRONLUND, *The natural factor formulation of the stiffness matrix displacement method*, Comput. Meth. Appl. Mech. Eng., 5 (1975), pp. 97-119.

[3] T. F. COLEMAN, A. EDENBRANDT AND J. R. GILBERT, *Predicting fill for sparse orthogonal factorization*, Technical Report 83-578, Dept. Computer Science, Cornell Univ., Ithaca, NY, October 1983.

[4] I. S. DUFF, *Pivot selection and row ordering in Givens reduction on sparse matrices*, Computing, 13 (1974), pp. 239-248.

[5] ———, *Full matrix techniques in sparse Gaussian elimination*, in Proc. Dundee Conference on Numerical Analysis, Lecture Notes in Mathematics 912, G. A. Watson, ed., Springer-Verlag, Berlin, 1982, pp. 71-84.

[6] I. S. DUFF AND J. K. REID, *A comparison of some methods for the solution of sparse overdetermined systems of linear equations*, J. Inst. Math. Appl., 17 (1976), pp. 267-280.

[7] ———, *The multi-frontal solution of indefinite sparse symmetric linear systems*, ACM Trans. Math. Software, 9 (1983), pp. 302-325.

[8] S. C. EISENSTAT, M. H. SCHULTZ AND A. H. SHERMAN, *Applications of an element model for Gaussian elimination*, in Sparse Matrix Computations, D. J. Rose, ed., Academic Press, New York, 1976, pp. 85-96.

[9] W. M. GENTLEMAN, *Least squares computations by Givens transformations without square roots*, J. Inst. Math. Appl., 12 (1973), pp. 329-336.

[10] ———, *Row elimination for solving sparse linear systems and least squares problems*, in Proc. Dundee Conference on Numerical Analysis, Lecture Notes in Mathematics 506, G. A. Watson, ed., Springer-Verlag, Berlin, 1975, pp. 122-133.

[11] J. A. GEORGE, *Nested dissection of a regular finite element mesh*, SIAM J. Numer. Anal., 10 (1973), pp. 345-363.

[12] J. A. GEORGE AND M. T. HEATH, *Solution of sparse linear least squares problems using Givens rotations*, Linear Alg. and Appl., 34 (1980), pp. 69-83.

[13] J. A. GEORGE, M. T. HEATH AND E. G. Y. NG, *A comparison of some methods for solving sparse linear least squares problems*, SIAM J. Sci. Stat. Comput., 4 (1983), pp. 177-187.

[14] J. A. GEORGE, M. T. HEATH AND R. J. PLEMMONS, *Solution of large scale sparse least squares problems using auxiliary storage*, SIAM J. Sci. Stat. Comput., 2 (1981), pp. 416-429.

[15] J. A. GEORGE AND J. W. H. LIU, *Computer Solution of Large Sparse Positive Definite Systems*, Prentice-Hall Inc., Englewood Cliffs, NJ, 1981.

[16] J. A. GEORGE, J. W. H. LIU AND E. G. Y. NG, *Row ordering schemes for sparse Givens transformations, I. Bipartite graph model*, Linear Alg. and Appl., 81 (1984), pp. 55-81.

[17] J. A. GEORGE AND E. G. Y. NG, *On row and column orderings for sparse least squares problems*, SIAM J. Numer. Anal., 20 (1983), pp. 326-344.

[18] ———, SPARSPAK: *Waterloo Sparse Matrix Package User's Guide for* SPARSPAK-B, Research Report CS-84-37, Dept. Computer Science, Univ. Waterloo, Waterloo, Ontario, November 1984.

[19] G. H. GOLUB AND C. VAN LOAN, *Matrix Computations*, Johns Hopkins Press, Baltimore, 1983.

[20] M. T. HEATH, *Numerical methods for large sparse linear least squares problems*, SIAM J. Stat. Sci. Comput., 5 (1984), pp. 497-513.

[21] A. G. JESS AND H. G. KEES, *A data structure for parallel L/U decomposition*, IEEE Trans. Comput., C-31 (1982), pp. 231-239.

[22] J. W. H. LIU, *On general row merging schemes for sparse Givens transformations*, Technical Report CS-83-04, Dept. Computer Science, York Univ., Downsview, Ontario, July 1983.

[23] ——, *On multiple elimination in the minimum degree algorithm*, Technical Report CS-83-03, Dept. Computer Science, York Univ., 1983; ACM Trans. Math Software, to appear.

[24] ———, *A compact row storage scheme for the Cholesky factor using elimination trees*, Technical Report CS-84-02, Dept. Computer Science, York Univ., Downsview, Ontario, 1984; ACM Trans. Math Software, to appear.

[25] P. E. MANNEBACK, Ch. MURIGANDE AND Ph. L. TOINT, *A modification of an algorithm by Golub and Plemmons for large linear least squares in the context of Doppler positioning*, Technical Report 84/101, Dept. Applied Mathematics, Facultés Universitaires de Namur, Namur, Belgium, January 1984.

[26] E. G. Y. NG, *Row elimination in sparse matrices using rotations*, Research Report CS-83-01, Dept. Computer Science, Univ. Waterloo, Waterloo, Ontario, 1983.

[27] O. OSTERBY AND Z. ZLATEV, *Direct Methods for Sparse Matrices*, in Lecture Notes in Computer Science 157, Springer, Berlin, 1983.

[28] J. K. REID, TREESOLVE: *a* FORTRAN *package for solving large sets of linear finite element equations*, CSS 155, Computer Science & Systems Division, AERE Harwell, Oxfordshire, March 1984.

[29] D. J. ROSE, *A graph-theoretic study of the numerical solution of sparse positive definite systems of linear equations*, in Graph Theory and Computing, R. C. Read, ed., Academic Press, New York, 1973, pp. 183-217.

[30] R. SCHREIBER, *A new implementation of sparse Gaussian elimination*, ACM Trans. Math Software, 8 (1982), pp. 256-276.

[31] Z. ZLATEV, *Comparison of two pivotal strategies in sparse plane rotations*, Comp. Math. Appls., 8 (1982), pp. 119-135.

SIAM J. SCI. STAT. COMPUT.
Vol. 7, No. 4, October 1986

010

A LINEAR TIME IMPLEMENTATION OF PROFILE REDUCTION ALGORITHMS FOR SPARSE MATRICES*

L. MARRO†

Abstract. The profile reduction method is intended for time and storage reduction in solving a linear system of equations $Mx = b$ using direct methods.

A Frontal Increase Minimization strategy (FIM strategy) is a generalization of the so-called King's numbering criterion. This class of strategy is used in some other classical algorithms (Levy's, Snay's, Gibbs's algorithms).

Although efficient, these algorithms are far greater time consumers in their original implementation than other classical profile reduction algorithms (e.g., Reverse Cuthill McKee algorithm).

In this paper we first apply the principles given by the authors to propose a unified "classical" implementation of the above mentioned algorithms. Then we provide some time complexity estimates for this implementation.

Secondly, we describe an improved implementation of the FIM strategy algorithms using a new insight into the numbering process and best appropriate data structures. This implementation is proven linear in time complexity with respect to the number of nonzeros in M for all the above-mentioned algorithms.

Finally, we provide practical execution times on a collection of test problems, highlighting the improvement achieved by the new implementation and its efficiency for small problems. The evaluation of the performance/cost ratio of the FIM strategy algorithms in the new implementation shows that they are competitive compared to other classical profile reduction algorithms.

Key words. G.1.3 [numerical analysis] numerical linear algebra, sparse and very large systems; G.4 [mathematics of computing] mathematical software, algorithm analysis

1. Introduction. A profile (or envelope) reduction method is intended for time and storage reduction in solving a linear system $Mx = b$ by direct methods, where the $n \times n$ matrix M is sparse and structurally symmetric ($m_{ij} \neq 0$ iff $m_{ji} \neq 0$). We assume that M fulfills the necessary conditions for numerical stability of the Gauss algorithm and that it is irreducible.

Sparse matrices normally suffer some fill-in when they are factored, i.e., the triangular factor matrix will have nonzeros in some of the positions which are zero in M. A wise selection of a $n \times n$ permutation matrix P can often drastically reduce fill-in and in turn imply a reduction of factorization cost. We solve, then, the equivalent problem $(PMP^t)(Px) = Pb$.

Profile reduction methods are one of the most widely used for fill-in reduction (see for example (George 1981)). Roughly speaking, the objective is to order the matrix so that the nonzeros of PMP^t are clustered on the average, nearest possible to the main diagonal. We define $f_i = \min\{j \mid m_{ij} \neq 0\}$ for $i = 1, \cdots, n$. This locates the leftmost nonzero element in each row. The envelope of the matrix M is $\mathrm{Env}(M) = \{m_{ij} \mid f_i \leqq j < i\}$. The profile is then defined as the number of elements in $\mathrm{Env}(M)$.

We study the problem using graph theory concepts. Given M we can construct an ordered undirected graph $G_\alpha = (S, E, \alpha)$, where S is the set of vertices, E the set of edges and α a numbering of the vertices from 1 to n. Each vertex v_i of S corresponds to a row i of M, and each edge $[v_i, v_j]$ corresponds to a nonzero m_{ij} of M. The unordered undirected graph $G = (S, E)$ corresponds to the equivalence class of matrices PMP^t.

The problem of profile reduction has already been seriously investigated and several efficient (in the practical sense) algorithms are available for solving this problem. Among those, we shall pay special care to the following ones: King's algorithm (King

* Received by the editors May 15, 1984; accepted for publication (in revised form) October 30, 1985.
† Computer Science Department, Aerospatiale Cannes, 06322 Cannes, La Bocca, France.

1970), Levy's algorithm (Levy 1971), Snay's algorithm (Snay 1976) and Gibbs's algorithm (Gibbs 1976a). These algorithms have been proven very effective for profile reduction in certain cases (see numerical results in (Everstine 1979), (Marro 1980) among others) but unfortunately they are great time consumers compared with other classical profile reduction algorithms such as the Reverse Cuthill McKee algorithm (see (George 1981)) or the Gibbs, Poole and Stockmeyer algorithm (Gibbs 1976b).

Recently, Lewis (Lewis 1982) presented a new implementation of the Gibbs's algorithm which improves execution times with respect to the original implementation given by Gibbs. Empirical experiments show a significant improvement, the new implementation appearing as faster than the original GPS algorithm. However we shall see that the Lewis implementation does not achieve linearity.

The first part of our work is an accurate analysis of King's, Levy's, Snay's and Gibbs's algorithms. We shall prove that the first three algorithms and a restricted form of the last one use the same numbering criterion: at each step, the vertex to be numbered is the one that minimizes the increase of front size, where the front will be defined as the set of vertices adjacent to at least one numbered vertex. Thus, this strategy is to be referred to hereinafter as "Frontal Increase Minimization strategy" (FIM strategy), and the above-mentioned algorithms as the "classical FIM strategy algorithms". We will see that they essentially differ in the definition of their individual search sets, i.e., the set of vertices in which the vertex to be numbered is sought at each step.

This synthesis will allow us to propose a unified implementation of the above-mentioned algorithms which is a framework for all FIM strategy algorithms and will be referred to as the "classical FIM strategy implementation" as it is obtained applying the principles given by the authors in their original papers.

We are providing an estimate of the time complexity of this implementation together with its worst case time complexity and corollaries results for King's and Gibbs's algorithms.

The second part of our work describes a more efficient implementation of FIM strategy. To improve efficiency we use a new insight into the numbering process as well as a more appropriate data structure. The time complexity analysis shows that the execution time is linear in the number of edges of G for all the classical FIM strategy algorithms, and, more generally, when the search set can be updated in linear time for one numbering.

This time complexity is the same as the one given for the RCM algorithm by Chan and George (Chan 1980). Numerical results show that the implementation is in practice more efficient than the classical one when the dimension of the set S is greater than one hundred.

Finally, we give a brief comparison of profile results and execution times of the FIM strategy algorithms in the new implementation with those of some other classical profile reduction algorithms.

An outline of this paper is as follows. In § 2, we introduce the necessary definitions using the graph theory formulation. In § 3, we analyze King's, Levy's, Snay's and Gibbs's algorithms with respect to the FIM strategy. In § 4, we describe the classical FIM strategy implementation and we give the corresponding time complexity results. Section 5 is devoted to the description of the new implementation, which is subsequently analyzed. Section 6 presents some numerical results and practical considerations.

2. Definitions and preliminary results. For classical definitions of the graph theory, the reader is referred to (Berge 1973).

Let $G=(S, E)$ be a simple, undirected graph, assumed to be connected, where S is a finite nonempty set of *n vertices* and E is a finite set of *m edges*. Each edge is

denoted by $[u, v]$, $u, v \in S$. The vertices $u \in S$ and $v \in S$ are said to be *adjacent* (neighbors) iff $[u, v] \in E$. $V(x)$ is the *adjacent set* of vertex x, i.e., $V(x) = \{y \in S \mid [x, y] \in E\}$. If $X \subset S$ we denote $V(X)$ the set of the neighbors of the vertices belonging to X.

The adjacent set to X is $\mathrm{Adj}(X) = V(X) - X$. We call the *degree* of the vertex x, denoted by $\deg(x)$, the number $|V(x)|$, (where $|X|$ denotes the number of elements in X). We will denote $\deg_M = \max_{x \in S} [\deg(x)]$.

We will denote by $d(x, y)$ the *distance* from vertex x to vertex y in G, and if $X \subset S$, $y \notin X$, $d(y, X) = \min_{x \in X} [d(y, x)]$. A *level structure* denoted by $L_G = \{N_0, N_1, \cdots, N_p\}$ is a partition of S into subsets N_i such that $N_{-1} = \phi$, $N_{p+1} = \phi$ and, for all i $0 \leqq i \leqq p$ $\mathrm{Adj}(N_i) \subset N_{i-1} \cup N_{i+1}$. The N_i sets are called *levels* and p is the *depth* of the level structure. The *width* of L_G will be $w(L) = \max_{0 \leqq i \leqq p} [|N_i|]$. A *descending level structure* is a level structure such that all the vertices belonging to level N_i must have a neighbor in level N_{i-1}, $\forall i$, $1 \leqq i \leqq p$. Equivalently if $x \in N_i$, $d(x, N_0) = i$. Thus a descending lever structure is fully determined by the level N_0 and will be denoted $L_G(N_0)$. A level structure is said to be *rooted* at $r \in S$ and denoted by $L_G(r)$ if L_G is a descending level structure and $N_0 = \{r\}$.

A *numbering* α of G is a bijective mapping $\alpha : S \to [1, n]$ ($[1, n]$ denoting the set of integers $\{1, \cdots, n\}$). We denote $N(G)$ the set of numberings of G and G_α where $\alpha \in N(G)$, the graph numbered by α. The numbers k, $k \in [1, n]$, are assigned in increasing order and the *step* k of the numbering will be the selection of vertex x such that $\alpha(x) = k$. A vertex x such that $\alpha(x) = 1$ will be called the *starting vertex* of the numbering. We define $\forall x \in S$, $\rho_\alpha(x) = \min [\min_{y \in V(x)}(\alpha(y)), \alpha(x)]$; then the *envelope* of G_α is the set

$$\mathrm{Env}(G_\alpha) = \{\{x, y\} \mid \rho_\alpha(x) \leqq \alpha(y) < \alpha(x), \forall x, y \in S\}.$$

The *profile* of G_α will be $\Pi_\alpha(G) = |\mathrm{Env}(G_\alpha)|$.

In this paper, we consider only algorithms without backtrack, i.e., each number is set once and only once to a vertex. It is the general case for classical numbering algorithms.

Let $SN^{(k)}$ be the set of numbered vertices at step k. We have $|SN^{(k)}| = k$; by convention $SN^{(0)} = \phi$ and $SN^{(1)} = \{x\}$, x being the starting vertex of the numbering. We define $SF_i^{(k)} = \{x \in S - SN^{(k)} \mid d(x, SN^{(k)}) = i\}$. $SF_1^{(k)}$ will be called the *numbering front* (or simply *front*) and we denote $L_k = |SF_1^{(k)}|$. At each step we have $SF_1^{(k)} = \mathrm{Adj}(SN^{(k)})$. We define $SR^{(k)} = S - (SN^{(k)} \cup SF_1^{(k)})$.

Figure 1 illustrates these different subsets of S.

The numbering process develops in the following way: First number a vertex, i.e., build $SN^{(k)}$, and then this defines $SF_1^{(k)}$ and $SR^{(k)}$.

The *search set* at step k denoted by $ER^{(k)}$ is the subset of $S - SN^{(k)}$ in which the vertex to be numbered at step $k+1$ is sought.

All the classical algorithms use a search set although it is not explicitly defined. For example, the choice criterion of the Cuthill-McKee algorithm (at each step vertex with the lowest numbered neighbor is selected) implies that the vertex to be numbered belongs to the front. Other algorithms can specify an explicit search set, for instance Snay's algorithm specifies that the vertex to be numbered must belong to $SF_1^{(k)} \cup SF_2^{(k)}$.

We introduce below a classification of $N(G)$ based on the definition of the search set for each class of numbering. A *progressive numbering* is an ordering in which the vertex numbered at step $k+1$ is chosen out of $S - SN^{(k)}$ (global search). A *p-frontal* numbering is a numbering where the selected vertex at step $k+1$ must belong to the set $\bigcup_{i=1}^{p} SF_i^{(k)}$. A 1-frontal numbering will be simply called *a frontal numbering.*

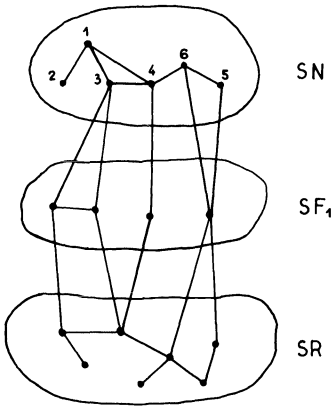


FIG. 1. *The sets* $SN^{(k)}$, $SF_1^{(k)}$, $SR^{(k)}$.

Let $L_G = \{N_0, N_1, \cdots, N_p\}$ a level structure of G. A numbering α will be said *consistent* with L_G if the vertices number of each level form an interval in $[1, n]$, i.e., when one vertex x of a level N_i is numbered, all the vertices of this level must be numbered prior to any vertex not belonging to N_i. Note that this type of numbering does not necessarily take account of the order of the level.

A numbering consistent with a level structure which follows the order of level will be called an *ordered consistent numbering.*

If $L_G(N_0)$ is a descending level structure, an ordered consistent numbering with $L_G(N_0)$ will be called a *descending numbering* with respect to $L_G(N_0)$.

Let $L_G(r)$ be a rooted level structure of G. *A rooted descending numbering* with respect to $L_G(r)$ is a descending numbering such that r is the starting vertex of the numbering.

We denote by $N_P(G)$, $N_F(G)$, $N_D(G)$, $N_{RD}(G)$, the sets of progressive, frontal, descending, rooted descending numberings of G.

Note that we have $N_{RD} \subset N_F \subset N_P$. This implies that properties true for progressive numberings will be also true for p-frontal and *rooted descending numberings.*

We define now an important notion for our work. We call the *front evolution set* at step k for a given vertex x the set $\Gamma(x) = \{(x) \cup V(x)\} \cap SR^{(k-1)}$. We denote $\gamma_x = |\Gamma(x)|$. If $\alpha(x) = k$, $\Gamma^{(k)}$ will denote the set $\Gamma(x)$ at step k and $\gamma_k = |\Gamma^{(k)}|$.

The concept of front plays a primary role in our analysis. It had been recently introduced for profile reduction studies by several researchers ((Marro 1980), (George 1981), (Amit 1982)). Theoretical results have shown the close connection between reduction of computational cost when using the envelope method for the Gauss decomposition and the minimization of the front size on the n steps of the numbering (see Theorem 2.3 below). Using graph theory, we can infer from these results lower bounds for space memory (profile) and for number of operations needed for these methods (Marro 1980). R. Amit and C. Hall provide a space memory estimate for frontal and envelope methods (Amit 1982).

In the following the front concept will be used to improve the knowledge of the numbering evolution providing us with an improved means to achieve the necessary updates during the numbering.

We give below some results we need in this paper. Detailed proofs and comments can be found in (Marro 1980).

The following lemma precisely describes the numbering process.

LEMMA 2.1. *Let α be a numbering of G. Then we have*

$$SN^{(0)}=\varnothing,\quad SF_1^{(0)}=\varnothing,\quad SR^{(0)}=S$$

and $\forall\, k\in[1,n]$, let $x\in S$ such that $\alpha(x)=k$. Thus

$$SN^{(k)}=SN^{(k-1)}\cup\{x\},$$
$$SF_1^{(k)}=SF_1^{(k-1)}\cup\Gamma^{(k)}-\{x\},$$
$$SR^{(k)}=SR^{(k-1)}-\Gamma^{(k)}.$$

Proof. The result is immediate from the definitions.

The following lemma states an interesting property of the front evolution set.

LEMMA 2.2. *Let $\alpha\in N_P(G)$. The sets $\Gamma^{(k)}$ for $k=1,\cdots,n$ (such that $\Gamma^{(k)}\neq\varnothing$) form a partition of $S\Rightarrow\sum_{k=1}^{n}\gamma_k=n$ and $\sum_{k=1}^{n}\sum_{y\in\Gamma^{(k)}}|V(y)|=2m$.*

Proof. See (Marro 1980).

The following result connects the profile obtained by a numbering with the front size at each step. It is a key result to justify the use of front in the studies of profile reduction numberings.

THEOREM 2.3. *The profile $\Pi_\alpha(G)$ obtained by a numbering α is the sum on the n steps of the size of the numbering front at each step*

$$\Pi_\alpha(G)=\sum_{k=1}^{n}|SF_1^{(k)}|.$$

Proof. See (Marro 1980).

3. Classical algorithms of FIM strategy. In this section we first define the FIM strategy, and secondly, study some previously published profile reduction algorithms with respect to this strategy.

We call Front Increase Minimization strategy (FIM strategy) the following numbering strategy: "The vertex numbered at step k is the one that minimizes L_k from L_{k-1}". From Lemma 2.1 we note that this is equivalent selecting at step k the vertex with minimum γ_x. Theorem 2.3 proves that FIM strategy is effective as a local (step-by-step) one for profile reduction.

The first author who (formally) set up a similar criterion for profile reduction is King (King 1970). He defines the front evolution set $\bar{\Gamma}(x)$ as being $V(x)\cap SR^{(k)}$ and number at step $k+1$ a vertex with minimum $|\bar{\Gamma}(x)|=\bar{\gamma}_x$ belonging to $SF_1^{(k)}$. We note that King's definition of the front evolution set is identical to the one we give in § 2 for vertices belonging to $SF_1^{(k)}$ but differs if x belongs to $SR^{(k)}$. With King's definition we have if $x\in SF_1^{(k)}$, $L_{k+1}=L_k+\bar{\gamma}_x-1$ and if $x\notin SF_1^{(k)}$, $L_{k+1}=L_k+\bar{\gamma}_x$. Thus, in this definition, the front increase differs for equal value of $\bar{\gamma}_x$ and if we want to apply the FIM strategy with a general search set we have to introduce another parameter to take account of that set to which vertex x belongs. This had been done by Snay (Snay 1976) who introduced a parameter $n_x=1$, if $x\in SF_1^{(k)}$ and $n_x=0$ if $x\in SR^{(k)}$ and selected the vertex with minimum $[\bar{\gamma}_x-n_x]$.

Snay proposes the search set $\cup_{i=1}^{2}SF_i^{(k)}$.

However, only a global search can lead to a correct step-by-step front minimization; thus Levy (Levy 1971) proposes to search for vertex x with minimum $\bar{\gamma}_x$ (the King definition) in the set $S-SN^{(k)}$.

Summarizing the above discussion, we can state the following result from Lemma 2.1.

PROPOSITION 3.1.

- *King's algorithm implements the* FIM *strategy for frontal numberings.*
- *Snay's algorithm implements the* FIM *strategy for* 2*-frontal numberings.*
- *Levy's algorithm implements the* FIM *strategy for progressive numberings* (*with* γ_x *being defined as in* § 2).

Numerical studies ((Gibbs 1976c), (Everstine 1979), (Marro 1980)) show that these FIM strategy algorithms are very efficient on certain classes of graphs; but unfortunately they sometimes lead to very poor results. Such "unstable" behavior happens when the minimization at each step does not complete a global optimization.

These "unstable" behaviors of the FIM strategy justify Gibbs (Gibbs 1976a) in writing a so-called "hybrid" algorithm: according to the analysis of the author, the algorithm builds an ordered consistent numbering with a precalculated general level structure, applying the King criterion for the selection of the vertex to be numbered. Thus this algorithm is known as the Gibbs-King algorithm (see for example (Lewis 1982)). However this designation must be considered with care. In Gibbs's paper the vertex to be selected must belong to the first level with unnumbered vertices (e.g., level N_i) and is the one which has fewest connections with the set

$$N_{i+1} - (N_{i+1} \cap SF_1^{(k)}).$$

For a general level structure this is different from King's criterion. See, for example, Fig. 2.

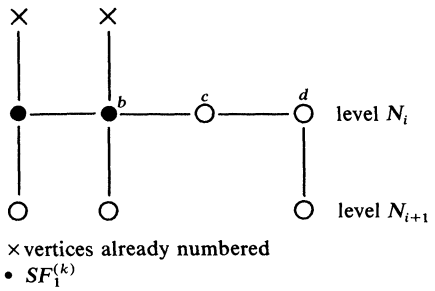


FIG. 2

For the FIM strategy and King's criterion $\gamma_a = 1$, $\gamma_b = 2$, $\gamma_c = 2$. For Gibbs's criterion $G_a = 1$, $G_b = 1$, $G_c = 0$. Thus FIM and King's strategy number vertex a and $L_{k+1} = L_k$, Gibbs's algorithm number vertex c and $L_{k+1} = L_k + 1$.

Similar case may occur for general level structures and for descending level structures for the numbering of vertices belonging to N_0. For a rooted descending level structure, Gibbs's criterion is always King's (and in this case the FIM strategy criterion). Hence we have

PROPOSITION 3.2. *Gibbs's algorithm implements the* FIM *strategy for rooted descending numberings.*

Justification and efficiency of the various search sets described above is a subject so far hardly studied, which would deserve some in-depth investigation. We can make the following remarks.

The only search set a priori theoretically satisfactory for a proper implementation of the step by step minimization is the Levy search set.

King's search set and Snay's search set are proposed to reduce the computation time for one numbering. Numerical results show that these two search sets seem to be not too restrictive, that is, almost in each step a vertex with minimum γ_x belongs to

these search sets. However, it is not difficult to find cases in which such search sets lead to very unsatisfactory results (e.g., star tree for the King algorithm).

The search set proposed by Gibbs is governed by another type of motivation as it is designed to direct numbering on a privileged path. Gibbs's search set is particularly effective wherever local optimization is insufficient to achieve global optimization of front size. But this search set is sometimes too restrictive as the Gibbs's algorithm gives poor results in some cases where other FIM strategy algorithms are very efficient such as network test cases or some finite elements problems (Marro 1980).

4. General implementation of FIM strategy. We describe here a general implementation of the FIM strategy which is a generalization from the analysis of § 3 of the implementation given by the above-mentioned authors.

Procedure GENERALFIM
Input An undirected graph $G=(S, E)$
Output A numbering α of vertices of S
Begin
1. INITIALIZE
2. (*Main Loop*). **For** $k:=1$ **to** n **do**
Begin
2.1. COMPUTE • **For each** y belonging to $ER^{(k-1)}$ compute γ_y;
2.2. SELECT • Choose a vertex belonging to $ER^{(k-1)}$ such that $\gamma_x=\min_{y\in ER^{(k-1)}}[\gamma_y]$;
2.3. Number x by k; $SN^{(k)}\leftarrow SN^{(k-1)}\cup\{x\}$; delete x from $ER^{(k-1)}$;
2.4. UPDATE Update of the sets $SF_1^{(k)}$, $SR^{(k)}$; Build $ER^{(k)}$;
End;
End GENERALFIM

4.1. Implementation details. In the initialization step we need to build $ER^{(0)}$. For a progressive numbering this set is S and thus is known immediately. The first vertex numbered will be a vertex of minimum degree which gives, for step 1, the minimum increase of front. For other types of numbering we need to build $ER^{(0)}$ explicitly and thus these algorithms require a starting vertex. In practice, such vertex is selected in such a way as to provide a global approach to front size optimization. Of course, this feature may be used for a progressive numbering.

Given a starting set D (possibly the empty set or a selected vertex) and a parameter ERK to choose through the different algorithms, step INITIALIZE may be written as follow:

Procedure INITIALIZE (D, ERK)
Begin
$SN^{(0)}\leftarrow\varnothing$
$SF_1^{(0)}\leftarrow\varnothing$
$SF_2^{(0)}\leftarrow\varnothing$
$SR^{(0)}\leftarrow S$
Case of (ERK) King, Levy, descending, Snay
Label King $ER^{(0)}\leftarrow D$
Label Levy $ER^{(0)}\leftarrow S$

Label descending $ER^{(0)} \leftarrow D$
Label Snay $ER^{(0)} \leftarrow D$
End INITIALIZE

The computations made in steps COMPUTE and SELECT do not depend on the search set used. The step COMPUTE may be implemented as follows.

Procedure COMPUTE
Begin
For all $y \in ER^{(k-1)}$ **do**
Begin $\gamma_y \leftarrow |(\{y\} \cup V(y)) \cap SR^{(k-1)}|$; **end**;
End COMPUTE

As we compute each γ_y for y belonging to $ER^{(k-1)}$, step SELECT is straightforward.

For the UPDATE step, the partition $SF_1^{(k)}$, $SR^{(k)}$, $SN^{(k)}$ of S can be maintained using a three-state array of length n. Update of Levy's search set is straightforward and Lemma 2.1 provides the King search set with an immediate update. For the Snay search set we will use the definition of $SF_2^{(k)}$. For a descending algorithm we admit that the considered level structure is built independently by a Breadth First Search algorithm and that it is available throughout the numbering. Thus step UPDATE can be implemented in a general manner for classical search sets. Let x be the vertex numbered at step k, and ERK be the same as in the INITIALIZE procedure.

Procedure UPDATE (x, ERK)
Begin
1. $\Gamma(x) \leftarrow (\{x\} \cup V(x)) \cap SR^{(k-1)}$;
$SF_1^{(k)} \leftarrow SF_1^{(k-1)} \cup \Gamma^{(k)} - \{x\}$;
$SR^{(k)} \leftarrow SR^{(k-1)} - \Gamma^{(k)}$;
2. (*update $ER^{(k)}$ from its definition*)
Case of (ERK) King, Levy, Descending, Snay.
Label King $ER^{(k)} \leftarrow SF_1^{(k)}$;
Label Levy $ER^{(k)} \leftarrow SF_1^{(k)} \cup SR^{(k)}$;
Label descending $N_i \leftarrow N_i - \{x\}$;
If $N_i \neq \varnothing$
then $ER^{(k)} \leftarrow N_i$;
Else $ER^{(k)} \leftarrow N_{i+1}$;
Label Snay $SF_2^{(k)} \leftarrow \text{Adj}\,(SF_1^{(k)}) \cap SR^{(k)}$;
$ER^{(k)} \leftarrow SF_1^{(k)} \cup SF_2^{(k)}$;
End UPDATE

4.2. Complexity analysis. This implementation is $O(m)$ for storage requirement using the classical representation of a graph by adjacency lists.

For execution time we note that for classical search sets except Snay's, the step UPDATE is executed in $O(\deg(x))$ independently of the size of $ER^{(k-1)}$. For Snay's algorithm step UPDATE is $O[\sum_{y \in SF_1^{(k)}} \deg(y)]$ for one step, thus $O[\sum_{k=1}^{n} \sum_{y \in SF_1^{(k)}} \deg(y)]$ for one numbering.

Hence, for one numbering step UPDATE is $O(m)$ for King's, Levy's and descending algorithms. For Snay's algorithm the time complexity of step UPDATE can be bounded by $(\deg_M \sum_{k=1}^{n} |SF_1^{(k)}|)$.

The step COMPUTE requires a $O[\deg(y)]$ loop, which has to be executed at each step $|ER^{(k-1)}|$ times. For one numbering step COMPUTE is $\sum_{k=1}^{n} \sum_{y \in ER^{(k-1)}} \deg(y) \leqq \deg_M \sum_{k=1}^{n} |ER^{(k-1)}|$. Clearly, for all the classical search sets

including the Snay one, step COMPUTE dominates the time complexity of the above implementation. Hence we have the following result:

PROPOSITION 4.1. *The time complexity of the above implementation of* FIM *strategy is* $O[\deg_M \sum_{k=1}^{n} |ER^{(k-1)}|]$.

This result shows off the role of $ER^{(k)}$ in the time complexity estimation. We note that if we decrease the search set, we improve the time complexity bound.

However, expressed in terms of the size of the problem, Proposition 4.1 leads to an overestimated result: for all the search sets, it is possible that $|ER^{(k)}| = n - k$ whereby $\sum_{k=1}^{n} |ER^{(k-1)}|$ is $O(n^2)$; as $\deg_M$ may be $O(n)$, Proposition 4.1 will give an $O(n^3)$ worst case time complexity bound. To obtain a better bound for sparse graphs, we note that, at each step $ER^{(k)} \subseteq S$, thus $\sum_{y \in ER^{(k)}} \deg(y) \leqq \sum_{y \in S} \deg(y)$ which may be bounded with respect to the number of edges in the graph (a classical result). For Snay's algorithm we have also $\sum_{y \in SF_1^{(k)}} \deg(y) \leqq \sum_{y \in S} \deg(y) = 2m$. Hence the following result:

PROPOSITION 4.2. *The above implementation of the* FIM *strategy is in the worst case* $O(m \cdot n)$ *for the time complexity of one numbering.*

We note that for the Levy algorithm this time complexity bound is always achieved. For King's and descending algorithms we set up other complexity meaningful parameters.

For the King algorithm we have $\sum_{k=1}^{n} |ER^{(k-1)}| = \Pi_\alpha(G)$ by Theorem 2.3. Hence:

COROLLARY 4.3. *The time complexity of the above implementation of King's algorithm is* $O[\deg_M \cdot \Pi_\alpha(G)]$.

Let $L_G(N_0)$ be a level structure. For a descending algorithm with respect to $L_G(N_0)$ we have

$$\sum_{k=1}^{n} |ER^{(k-1)}| = \sum_{i=1}^{p} \sum_{j=1}^{|N_i|} (|N_i| - j) = \frac{1}{2} \sum_{i=1}^{p} |N_i|^2$$

$$\leqq \frac{w(L)}{2} \sum_{i=1}^{p} |N_i| = n \cdot \frac{w(L)}{2}.$$

Hence we have:

COROLLARY 4.4. *Let* $L_G(N_0)$ *be a descending level structure. The time complexity of the above implementation of* FIM *strategy algorithm, descending with respect to* $L_G(N_0)$ *is* $O[deg_M \cdot n \cdot w(L)]$.

In the above implementation, the SELECT step is performed during the COMPUTE step. However, we note that, if the two steps are done independently, the step SELECT requires the reading or sorting of $|ER^{(k)}|$ values at each step. This separate process occurs in the Lewis implementation of the Gibbs-King algorithm (Lewis 1982).

Thereby, the time complexity of this implementation, independently of the starting vertex selection, is at least $O[\sum_{i=1}^{p} |N_i|^2]$ i.e., $O[w(L) \cdot n]$, which is not linear.

5. Improved implementation of the FIM strategy algorithms.

5.1. Principles. We have seen in our analysis of classical implementation of the FIM strategy that the most costly operations were the computation of γ_x for each vertex belonging to $ER^{(k)}$ at each step and the test on $ER^{(k)}$ for finding the vertex to be numbered. Each of these operations requires the perusal of the whole set $ER^{(k)}$. The basic idea of our improvement has been to disconnect these two operations from the size of $ER^{(k)}$. To implement this idea, we used two types of techniques, say:

—an insight into the numbering process,

—an efficient data structure.

In our implementation, from a proper initialization value, the γ_x values for those vertices x which belong to the search set are put in ascending order and, at each step, only a few values of this parameter are recomputed and reordered. All the classical search sets are updated in a way that is irrespective of their sizes. Now the selection of the vertex to be numbered is immediate.

5.2. General presentation.

Analysis of the numbering process. A detailed analysis of the numbering process has done away with the computation of all γ_x values for all vertices belonging to $ER^{(k)}$ at each step. We use the following two lemmas, whose proofs are straightforward.

LEMMA 5.1. *Let $\alpha \in N_P(G)$. At step $k=1$, we have*

$$\forall x \in S, \qquad \Gamma(x) = \{x\} \cup V(x) \Rightarrow \gamma_x = \deg(x) + 1.$$

LEMMA 5.2. *Let $\alpha \in N_P(G)$. $\forall y \in S$, γ_y is modified at step k, iff $y \in \Gamma^{(k)} \cup V(\Gamma^{(k)})$.*

Thus, from the initialization value of Lemma 5.1, we update at each step only the value of γ_y for vertices belonging to the subset of S as defined in Lemma 5.2.

From the analysis of § 4.2 we need to improve the update of the Snay search set. We use the following lemma.

LEMMA 5.3. *$\forall k \in [1, n]$ and x such that $\alpha(x) = k$, we have*

$$SF_2^{(k)} = SF_2^{(k-1)} \cup [\mathrm{Adj}(\Gamma^{(k)}) \cap SR^{(k)}] - SF_2^{(k-1)} \cap [\{x\} \cup V(x)].$$

Proof. From the definition we have

$$SF_2^{(k)} = \mathrm{Adj}(SF_1^{(k)}) \cap SR^{(k)}.$$

Using Lemma 2.1 we write

$$SF_2^{(k)} = \mathrm{Adj}(SF_1^{(k-1)}) \cap SR^{(k)} \cup \mathrm{Adj}(\Gamma^{(k)}) \cap SR^{(k)} - \mathrm{Adj}(x) \cap SR^{(k)}$$

as $\mathrm{Adj}(x) \cap SR^{(k)} = \varnothing$; and applying Lemma 2.1, we have

$$\begin{aligned} SF_2^{(k)} &= \mathrm{Adj}(SF_1^{(k-1)}) \cap SR^{(k-1)} \cup \mathrm{Adj}(\Gamma^{(k)}) \cap SR^{(k)} \\ &\quad - \mathrm{Adj}(SF_1^{(k-1)}) \cap \Gamma^{(k)}, \\ SF_2^{(k)} &= SF_2^{(k-1)} \cup \mathrm{Adj}(\Gamma^{(k)}) \cap SR^{(k)} \\ &\quad - \mathrm{Adj}(SF_1^{(k-1)}) \cap [SR^{(k-1)} \cap (\{x\} \cup V(x))], \end{aligned}$$

whence the result.

Data structure description. To do away with accessing the whole $ER^{(k)}$ set to be able to select the vertex to be numbered, we intend to reach only those vertices with minimum γ_y belonging to $ER^{(k)}$. To achieve this goal we use a classical data structure: vertices having the same γ_y are stored in a doubly-linked list. The entry points of each list are stored in ascending order with respect to the value of γ_y using arrays ORI (for the head) and FIN (for the tail) of length $\deg_M + 1$. The first vertex with $\gamma_y = i$ is in ORI (i); the last vertex is in FIN (i). Note that this organization allows us to manage each list as a queue. This point will be used later. Deleting and inserting (at one end) a vertex in these lists is easily done in a constant number of operations independent of the size of the list, because the values managed are identical to the addresses in the arrays of forward and backward pointers. The same analysis and results which provide us with the update of $ER^{(k)}$, allow us to update the lists at each step in such a way that they include only vertices belonging to $ER^{(k)}$.

This data structure is depicted on an example in Fig. 3.

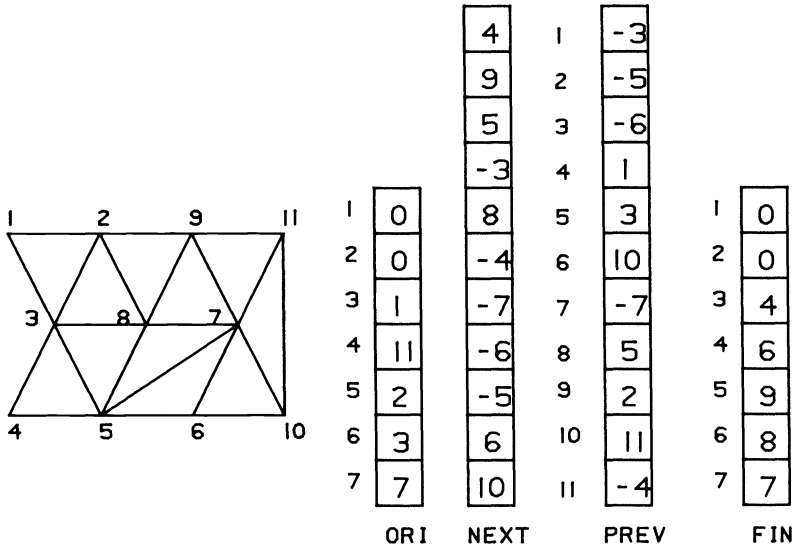


FIG. 3

Now, we immediately select a vertex of minimum γ_y by maintaining a pointer to the first nonzero entry in ORI and picking up the first vertex because any vertex of this list belongs to $ER^{(k)}$. A special feature is provided for vertices such that $\gamma_y = 0$, that are stored in a stack data structure to be numbered immediately.

Thus we note that using this data structure, the crucial steps become the update of γ_x and of the search set. Another example of an application of the same type of data structure in a numbering algorithm can be found in (George 1980).

5.3. Description of the new implementation. We give here an overall plan of the new implementation of the FIM strategy algorithms. Details will be given in § 5.4 and an evaluation of its complexity in § 5.5.

Procedure NEWFIM
Input An undirected graph $G=(S, E)$
Output A numbering α of vertices of S
Begin
1. (*Initialize*)
   1.1. INITIALIZE
   1.2. $\forall y \in ER^{(0)}$. Build the doubly-linked lists using
   $\gamma_y = \deg(y)+1$;
2. (*Main Loop*)
   For $k := 1$ **to** n **do**
   Begin
   2.1. SELECT
   If $\exists y \in ER^{(k-1)}$ such that $\gamma_y = 0$
   Then $x \leftarrow y$;
   Else
   $x \leftarrow$ the first vertex of the first nonempty list;
   2.2. NUMBER Number x by k; $SN^{(k)} \leftarrow SN^{(k-1)} \cup \{x\}$;
   delete x from $ER^{(k-1)}$; $\gamma_k \leftarrow \gamma_x$;

```
2.3. UPDATE
     Begin
     If γ_k ≠ 0 then
       Begin
       Γ^(k) ← ∅; V(Γ^(k)) ← ∅;
     2.3.1. UPDATE.1      Update γ_y for each
                          y ∈ [Γ^(k) ∪ V(Γ^(k))];
     2.3.2. UPDATE.2      Update SF_1^(k), SR^(k), ER^(k);
     2.3.3. UPDATE.DLL    Update the doubly-linked lists
                          for y ∈ [Γ^(k) ∪ V(Γ^(k))] ∩ ER^(k);
       End;
         Else;
     End UPDATE;
  End;
End NEWFIM;
```

5.4. Implementation details. The INITIALIZE procedure is the same as the one for the classical implementation.

In § 5.2 we noted that step UPDATE is crucial for the complexity of the new implementation. Given x the vertex numbered at step k, we build the sets $\Gamma^{(k)}$ and $V(\Gamma^{(k)})$ updating the value of γ_y for vertices belonging to these sets. It is important to note that each vertex is easily included once only in these sets. Once these sets are known, $SF_1^{(k)}$ and $SR^{(k)}$ are updated in the same manner as in the classical implementation. The search sets are also updated in the same manner except for the Snay one for which we use Lemma 5.3. For vertices belonging to $[\Gamma^{(k)} \cup V(\Gamma^{(k)})] \cap ER^{(k)}$ the update of the doubly-linked lists is done in classical mode. We do not describe here the necessary instructions (see, for instance, (Aho 1974)). The procedure UPDATE.DLL is used to update the value of $\gamma_{\min}$ and to pick up the vertices with $\gamma_y = 0$ which will be numbered first. Such vertices are stored in a stack data structure IZER.

From Lemmas 5.2 and 5.3, procedures UPDATE.1, UPDATE.2 and UPDATE.DLL can be implemented as follows.

```
Procedure UPDATE.1 (x)
(*build {Γ^(k) ∪ V(Γ^(k))} and update γ_y for y belonging to this set*)
Begin
For each y ∈ {{x} ∪ V(x)} do
  Begin
  If x ∉ SF_1^(k-1) then γ_y ← γ_y − 1;
  If y ∉ SN^(k) ∪ SF_1^(k-1)
    Then begin
      Γ^(k) ← Γ^(k) ∪ {y}; γ_y ← γ_y − 1;
      End;
    Else;
  End;
For each y ∈ Γ^(k) do
Begin
For each x ∈ V(y) do
  Begin
  V(Γ^(k)) ← V(Γ^(k)) ∪ {z}; γ_z ← γ_z − 1;
  End;
```

```
End;
End UPDATE.1

Procedure UPDATE.2 (x, Γ^(k), V(Γ^(k)), ERK)
(*the parameter ERK has the same meaning as in classical implementation*)
Begin
SF_1^(k) ← SF^(k-1) ∪ Γ^(k) − {x};
SR^(k) ← SR^(k-1) − Γ^(k);
Case of (ERK) King, Levy, descending, Snay
  Label King          ER^(k) ← SF_1^(k);
  Label Levy          ER^(k) ← SF_1^(k) ∪ SR^(k);
  Label descending    N_i ← N_i − {x};
                      If N_i ≠ ∅ then ER^(k) ← N_i;
                                 Else ER^(k) ← N_(i+1);
  Label Snay          SF_2^(k) ← SF_2^(k-1) ∪ [Adj (Γ^(k)) ∩ SR^(k)] −
                      SF_2^(k-1) ∩ [{x} ∪ V(x)];
                      ER^(k) ← SF_1^(k) ∪ SF_2^(k);
End

Procedure UPDATE.DLL
Begin
IZER ← ∅
For each y ∈ {Γ^(k) ∪ V(Γ^(k))} ∩ ER^(k) do
  Begin
    delete y from the old list;
    If γ_y = 0 then IZER ← IZER ∪ {y};
      Else
      Begin
      Insert y in the new list γ_y;
      If γ_y ≦ γ_min then γ_min ← γ_y; Else;
      End
  End;
End UPDATE.DLL;
```

The SELECT procedure is straightforward. The vertices (if any) with $\gamma_y = 0$ are first numbered. If not, the first vertex in the list with $\gamma_y = \gamma_{\min}$ is chosen.

```
Procedure SELECT
Begin
If IZER ≠ ∅ then x ← HEAD (IZER);
  Else
    Begin;
    Q ← list of vertices with γ_y = γ_min;
    x ← HEAD (Q);
    End;
End SELECT;
```

5.5. Complexity analysis. The implementation is $O(m)$ for storage requirement using adjacency lists for graph representation. The partition of S is updated in an array of length n, as the values of γ_x. The doubly-linked lists need two arrays of length n and two arrays of length $\deg_M + 1$.

As noted before step UPDATE is crucial for the time complexity of the algorithm. (In the following x will be the vertex numbered at each step.)

Clearly step UPDATE.1 is $O(|V(x) \cup \cup_{y\in\Gamma(x)} V(y)|)$. As in the classical implementation, the operations done in step UPDATE.2 are $O[\deg(x)]$ for King's and Levy's search sets. For descending algorithm step UPDATE.2 is $O(\sum_{i=1}^{p} |N_i|)$ and thereby $O(n)$ for one numbering. Update of the Snay search set is now $O(|\cup_{y\in\Gamma(x)} V(y)|)$ for each step of the numbering. Step UPDATE.DLL has the same time complexity as step UPDATE.1.

Hence, for one numbering, the time complexity of step UPDATE for all classical search sets:

$$\begin{aligned} T(S) &= 2 \sum_{x\in S} \Big| V(x) \cup \bigcup_{y\in\Gamma(x)} V(y) \Big| + \sum_{x\in S} \Big| \bigcup_{y\in\Gamma(x)} V(y) \Big| \\ &\leqq 2 \sum_{x\in S} |V(x)| + 2 \sum_{x\in S} \Big| \bigcup_{y\in\Gamma(x)} V(y) \Big| + \sum_{x\in S} \Big| \bigcup_{y\in\Gamma(x)} V(y) \Big| \\ &\leqq 4m + 3 \sum_{x\in S} \sum_{y\in\Gamma(x)} |V(y)|. \end{aligned}$$

Now, the key result to demonstrate the linearity of the above implementation is Lemma 2.2, which states that $\sum_{x\in S} \sum_{y\in\Gamma(x)} |V(y)| = 2m$, since $\cup_{k=1}^{n} \Gamma^{(k)}$ forms a partition of S.

Therefore we have $T(S) \leqq 10m$, i.e. step UPDATE has a time complexity of $O(m)$ for one numbering.

Steps SELECT and NUMBER are $O(1)$; hence we have the following result.

THEOREM 5.4. *The above-described implementation is $O(m)$ for the time complexity of all classical* FIM *strategy algorithms.*

5.6. Additional remarks. If the FIM strategy is to be implemented with precision for descending numberings, step SELECT has to be modified. In effect due to the numbering of vertices such that $\gamma_x = 0$ as soon as they are present, the new implementation does not exactly achieve descending numbering. However, this modification can only improve the results as it decreases the front size as early as possible. Therefore it is used in the algorithm tested below.

Often numbering algorithms use a second choice criterion for tie breaking among vertices with the same value of γ_x. For instance King (King 1970) proposed to select a vertex having the maximum waiting time in the front, i.e., that vertex which came in first in the front. Such strategy improves on the whole the results obtained by the algorithm (see (Marro 1980)).

In our implementation such a feature will necessitate reading the whole list of $\gamma_{\min}$. The number of such readings is difficult to estimate; in the worst case it is bounded by $n-k$ at step k, whereby step SELECT should have a time complexity in $O[\sum_{k=1}^{n} (n-k)]$ leading to an $O(n^2)$ worst time complexity for the algorithm. To avoid this drawback we may perform orderly insertion of vertices in the procedure UPDATE.DLL, in accordance with the second choice criterion. In this way the time complexity of step UPDATE will increase to $O(m \log_2 n)$, through management of the lists as *AVL*-trees (see (Aho 1974)), $O(m \log_2 n)$ now becoming the time complexity of the implementation.

However the numerical results show a very small influence on the running time of the first method for tie break strategy (Marro 1982).

For the numerical tests presented below, we implement another second choice strategy which retains the linear time complexity of the implementation. We manage

each doubly-linked list as a queue data structure (see § 5.2) inserting the new vertex at the end of the list and selecting the vertex to be numbered at the head of the list. Such strategy can be formulated as "in case of tie, select that vertex for which the value of γ_x has not been modified over the greatest number of steps".

This strategy leads to results which are on the whole equivalent to those obtained using the waiting-time criterion.

6. Numerical results. In this section we give numerical results in order to demonstrate the practical efficiency of the new implementation. We also compare the performance/cost ratio of the new implementation of FIM strategy algorithms with other classical profile reduction algorithms, the *RCM* and *GPS* ones. These results evidence that FIM strategy algorithms become competitive in this field even for large problems.

Our test set consists of three kinds of data, in that they are arranged as 39 finite element tests (ranging from $n = 59$ to $n = 2680$), 3 network tests ($n = 118$, 1723, 5300), and 9 graded-L-shaped tests (ranging from $n = 265$ to 2233).

The finite element test set is an outgrowth of the Everstine collection (Everstine 1979) excluding the nonconnected tests and including tests from structural analysis models studied at Aerospatiale Cannes (Marro 1980). Network tests are those used by Lewis and Poole in their comparative study (Lewis 1980). The L-shaped tests are from George's collection (George 1981).

All these tests can be obtained from I. A. Duff or J. G. Lewis.[1]

All execution times displayed in the tables below are given in CPU seconds on a CII HB IRIS 80 computer (about 0.75 Mips).

TABLE 6.1
Execution times for the Levy search set on the graded L-shaped tests.

N	Time	Time/m ($\times 10^{-4}$)
265	0.54	7.2
406	0.83	7.1
577	1.20	7.2
778	1.62	7.2
1009	2.11	7.2
1270	2.54	6.8
1561	3.12	6.8
1882	4.03	7.3
2233	4.87	7.4

Table 6.1 displays the results from our implementation of FIM strategy algorithms with Levy's search set on the graded L-shaped tests. We give the execution times for one numbering and the execution time/m ratio for this numbering with a view to providing an evaluation of the new implementation's run time complexity constant.

The Levy algorithm, which is the FIM strategy algorithm used by Everstine for his comparative study, is currently used in such Finite Element computer programs as NASTRAN (Everstine 1979).

[1] *Sparse matrix test problems.* Contact:
I. A. Duff, Numerical Analysis Group, Computer Sciences & System Division, Bldg. 8.9, AERE, Harwell, Oxfordshire OX11 0RA, U.K.
J. G. Lewis, Boeing Computer Service Co., Mail Stop 9C-01, 565 Andover Park West, Tukwila, WA 98188.

TABLE 6.2
Execution times for the King search set.

N	Classical implementation	New implementation	$\text{Time}_{\text{old}}/\text{Time}_{\text{new}}$
59	0.07	0.08	0.9
61	0.12	0.09	1.3
66	0.06	0.10	0.6
72	0.06	0.09	0.7
87	0.12	0.13	0.9
96	0.30	0.17	1.8
118	0.13	0.13	1
144	0.25	0.19	1.3
161	0.51	0.30	1.7
162	0.30	0.26	1.2
187	0.95	0.33	2.9
193	1.66	0.46	3.6
209	0.82	0.35	2.3
221	0.45	0.39	1.2
229	1.20	0.39	3.1
245	0.63	0.37	1.7
256	2.22	0.44	5.0
268	1.83	0.48	3.8
292	2.02	0.51	4
307	1.68	0.56	3
310	0.67	0.57	1.2
361	1.18	0.70	1.7
419	1.80	0.71	2.5
445	4.14	0.85	4.9
503	6.92	1.05	6.6
592	3.50	1.07	3.3
634	14.34	1.27	11.3
715	7.38	1.40	5.3
758	2.32	1.31	1.8
838	14.91	1.62	9.2
869	3.46	1.66	2.1
878	3.76	1.79	2.1
918	15.58	1.65	9.4
992	7.44	2.16	3.4
1005	21.04	1.52	13.8
1007	5.07	2.00	2.5
1054	22.06	1.90	11.6
1072	20.10	1.91	10.5
1242	13.41	2.38	5.6
1723	8.77	2.13	4.1
2680	35.64	5.52	6.4
5300	60.87	8.43	7.2
Total	289.7	48.2	6.0

In Table 6.2 we compare the execution times of King's algorithm in its classical versus new implementation on the finite element and network test cases. Execution times are given for one numbering, excluding the starting vertex selection running time.

The classical implementation is coded using a doubly-linked list to manage $SF_1^{(k)}$. Vertices such that $\gamma_y = 0$ are numbered as soon as they are detected without further reading, and thus step COMPUTE involves a number of readings not greater than

$\Pi_\alpha(G)$. In this way, the classical implementation tested can be considered as a good conventional one. The waiting time in front is used as the second choice criterion.

Results in Table 6.2 demonstrate the superiority of the new implementation. Total execution time is divided by a factor 6. The $\text{Time}_{\text{old}}/\text{Time}_{\text{new}}$ ratio, while it varies with m and the profile obtained, is greater than 1 however, as soon as n is greater than 100 and it increases with n. This demonstrates that the new implementation is efficient for small and large problems alike.

Theorem 5.4 proves that all classical FIM strategy algorithms are of linear time complexity in the new implementation. However change in the search set may have an influence on the practical running time, for instance on how many times the loop in the procedure UPDATE-DLL is executed. To estimate such influence, we run the new implementation with the Snay search set on the 42 tests above. For one numbering in each individual case the total running time is 58.8 s. Thus, we note only a slight increase of the execution time. Because the whole set updated is included in $SF_1^{(k)} \cup SF_2^{(k)}$, practical time complexity is the same for Levy and Snay search sets.

The question is now raised whether the FIM strategy in the new implementation becomes competitive, in terms of profile reduction performances/execution times ratio, with other profile reduction algorithms such as RCM or GPS algorithms.

As clues toward an answer, we present here some results of the GPS, RCM, 2-frontal and descending FIM strategy algorithms on networks and large finite elements tests ($N > 600$ in our collection).

For the GPS algorithm we use the FORTRAN code given in (Crane 1976). For the RCM algorithm we use the implementation given by Chan and George (Chan 1980) which has been proven linear in time complexity.

Profile reduction performances of all the algorithms tested are very sensitive to the starting vertex chosen. The problem of selecting efficiently the best starting vertex is not an easy one and will not be considered here. All the results given below are obtained using the vertex selected by the GPS algorithm.

In order to be consistent with the GPS results, the execution times given include starting vertex selection and computation of profile results. In FIM strategy algorithms, direct results are computed using Lemma 2.1 and Theorem 2.3, which is faster than the graph permutation used in the GPS algorithm. For the descending FIM strategy algorithm, direct and reverse numberings results are computed as in the Gibbs-King algorithm (Gibbs 1976a). The FIM strategy algorithms use the tie break criterion described in § 5.6.

Table 6.3 displays average profile and execution times of the above-mentioned algorithms on network test cases. The results are given for symmetric data.

It is clear, from the results in Table 6.3, that descending algorithms are inefficient on network test cases. Snay's algorithm is now by far prevalent in term of performance/cost ratio. The results are quite convincing as regards use of this algorithm on network models.

TABLE 6.3

	GPS		RCM		FIM-descending		FIM-2-frontal	
N	Time	Profile	Time	Profile	Time	Profile	Time	Profile
118	0.22	6.10	0.17	6.60	0.26	6.17	0.27	5.9
1723	4.16	40.3	1.77	43.1	5.56	36.1	3.59	16.6
5300	15.43	108.3	5.62	125.30	14.19	102.75	13.14	49.32

TABLE 6.4

	GPS		RCM		FIM-descending		FIM-2-frontal	
N	Time	Profile	Time	Profile	Time	Profile	Time	Profile
634	2.7	59.6	2.0	56.9	3.1	55.0*	3.1	66.3
715	8.3	57.5	7.7	53.0	8.9	50.5	8.9	37.5*
758	3.2	9.9*	1.6	10.3	3.1	10.4	2.8	10.6
838	3.8	44.9	2.8	45.1	4.2	44.3	4.2	43.4*
869	4.2	17.8	2.9	18.5	4.6	18.6	4.4	16.2*
878	4.4	21.7*	3.4	24.1	5.2	23.0	5.0	24.3
918	3.9	22.2*	2.7	25.6	4.4	23.1	4.5	48.9
992	11.1	33.3*	9.4	37.4	11.8	35.3	11.2	36.6
1005	2.8	41.9	1.7	42.0	3.7	39.3*	3.0	77.4
1007	5.4	21.6*	4.1	23.8	6.3	23.5	5.0	23.2
1054	4.7	51.1	3.4	39.3	5.2	36.9*	5.0	59.6
1072	6.4	69.1	5.2	52.0	6.9	47.5	7.1	39.1*
1242	6.2	43.9	4.8	43.6	7.1	40.0	6.9	35.6*
2680	10.3	37.9	5.3	38.5	11.1	36.0*	10.6	46.6
Total	77.4	532.4	57.0	510.1	85.6	483.4	81.7	565.3

* Best results.

Table 6.4 provides the same data as Table 6.3, this time on large finite element tests.

For this set of problems, the situation is less clear. The fastest algorithm is the RCM one. The use of the GPS code described in (Lewis 1982) will reduce the GPS execution time by about 50%. Due to the descending level structure generation and the computation of reverse results the FIM-descending algorithm is more costly than the 2-frontal one. However all the execution times are now acceptable.

The 2-frontal FIM strategy algorithm, while presenting some "unstable" behaviors ($n = 918, 1005$) and being the least efficient on the whole, performs significantly better than the descending algorithms in some cases ($n = 715, 1072, 1242$).

The descending FIM strategy algorithm performs best on the whole in profile reduction. It clearly becomes competitive with GPS and RCM algorithms in terms of performance/cost ratio. From (Marro 1980) we note that the Gibbs-King algorithm performs slightly better than the FIM-descending algorithm except on one case $n = 838$. On this collection of test problems the sum of the average profile for the Gibbs-King algorithm is 485.5.

This comparative study demonstrates that the profile reduction performance of FIM strategy algorithms are crucially dependent on the fitting on the search set used to the numbered graph.

This well substantiated finding strongly appeals for further improvement of the FIM strategy algorithms, namely in that the search set should be allowed to vary depending on the case handled. In (Marro 1985) we described an implementation of such variability as to fit the search set into the numbered graph and respond to the evolution of numbering. This leads to a significant overall improvement of the FIM strategy results. A forthcoming paper will present our work on the subject.

7. Conclusion. In this paper we have described a new implementation of those FIM strategy algorithms used in sparse matrix profile reduction which is more efficient than the classical implementations.

The linear time complexity for all classical search sets justifies selecting search sets only for the purpose of efficiency obtained for profile results with only a limited increase of time. Numerical results demonstrate that FIM strategy algorithms are now competitive in terms of performance/cost ratio with other classical algorithms such as RCM of GPS one. Such an improved implementation offers several characteristics which are attractive for further developments.

The local update of the γ_x values and of the search set, and the fast selection procedure for the vertex to be numbered were instrumental in providing the opportunity to design a FIM strategy algorithm in which the search set can vary throughout the numbering process. Such a "dynamic allocation" of the search set amounts to a significant breakthrough from the classical algorithms, in which the search sets are preassessed, nonflexible ones. Such adjustment of the search set to the graph to be numbered and to the evolution of numbering leads to significant improvements of results (Marro 1985).

In another domain of application, we may note that the FIM strategy has been recently rediscovered by various authors (Pina 1981), (Hoït 1982), as a renumbering strategy to improve the efficiency of the Gauss Frontal method (Irons 1970). We feel that the implementation we described here above could also be used in this field with a few adjustments.

Acknowledgment. The author is most thankful to professor Max Fontet for his many apt suggestions for the fulfillment of this work, as they considerably improved the original draft.

REFERENCES

(Aho, 1974) A. V. AHO, J. E. HOPCROFT AND J. D. ULLMAN, *The Design and Analysis of Computer Algorithms*, Addison-Wesley, Reading, MA, 1974.

(Amit, 1982) R. AMIT AND C. HALL, *Storage requirements for profile and frontal elimination*, SIAM J. Numer. Anal., 19 (1982), pp. 205-218.

(Berge, 1973) C. BERGE, *Graphes et Hypergraphes*, Dunod, Paris, 1973.

(Chan, 1980) W. M. CHAN AND A. GEORGE, *A linear time implementation of the reverse Cuthill-McKee algorithm*, BIT, 20 (1980), pp. 8-14.

(Crane, 1976) H. L. CRANE, N. E. GIBBS, W. G. POOLE AND P. K. STOCKMEYER, *Algorithm* 509: *Matrix bandwidth and profile reduction*, ACM Trans. Math. Software, 2 (1976), pp. 375-377.

(Everstine, 1979) G. C. EVERSTINE, *A comparison of three resequencing algorithms for the reduction of matrix profile and wavefront*, Internat. J. Numer. Meth. Engrg., 14 (1979), pp. 837-853.

(George, 1980) A. GEORGE AND J. W. H. LIU, *A minimal storage implementation of the minimum degree algorithm*, SIAM. J. Numer. Anal., 17 (1980), pp. 282-299.

(George, 1981) ———, *Computer Solution of Large Sparse Positive Definite Systems*, Prentice-Hall, Englewood Cliffs, NJ, 1981.

(Gibbs, 1976a) N. E. GIBBS, *Algorithm* 509: *a hybrid profile reduction algorithm*, ACM Trans. Math. Software, 2 (1976), pp. 378-387.

(Gibbs, 1976b) N. E. GIBBS, W. G. POOLE JR AND P. K. STOCKMEYER, *An algorithm for reducing the bandwidth and profile of a sparse matrix*, SIAM J. Numer. Anal., 13 (1976), pp. 236-250.

(Gibbs, 1976c) ———, *A comparison of several bandwidth and profile reduction algorithms*, ACM Trans. Math. Software, 2 (1976), pp. 322-330.

(Hoït, 1982) M. HOÏT AND E. L. WILSON, *An equation numbering algorithm based on a minimum front criteria*, Computers and Structures, 16 (1982), pp. 255-239.

(Irons, 1970) B. IRONS, *A frontal solution program for finite element analysis*, Internat. J. Numer. Meth. Engrg., 2 (1970), pp. 5-32.

(King, 1970) I. P. KING, *An automatic reordering scheme for simultaneous equations derived from network systems*, Internat. J. Numer. Meth. Engrg., 2 (1970), pp. 523-533.

(Levy, 1971) R. LEVY, *Resequencing of the structural stiffness matrix to improve computational efficiency*, Jet. Prop. Lab. Quart. Techn. Rev., 1 (1971), pp. 61-70.

(Lewis, 1980) J. G. LEWIS AND W. G. POOLE JR, *Ordering algorithms applied to sparse matrices in electrical power problems*, Proc. Conference: Electric Power problems: The Mathematical Challenge, Seattle, WA, 1980, Society for Industrial and Applied Mathematics, Philadelphia, PA, 1980.

(Lewis, 1982) ———, *Implementation of the Gibbs-Poole-Stockmeyer and Gibbs-King algorithms*, ACM Trans. Math. Software, 8 (1982), pp. 180-189.

(Marro, 1980) L. MARRO, *Methodes de réduction de la largeur de bande et du profil efficace des matrices creuses*, Thèse de 3ème cycle en Mathématiques, Nice, 1980. The main theoretical results of the author's thesis were presented at the first Mathematics for Computer Science symposium, Paris, March 16-18, 1982, in the paper titled: "A theoretical study of profile reduction numbering for sparse matrices".

(Marro, 1982) ———, *Implementation en $O(m)$ des algorithmes à stratégie MAF pour la réduction du profil efficace des matrices creuses*, Rapport d'Etudes Libres, Aerospatiale Cannes, 1982.

(Marro, 1985) ———, ADER: *Un algorithme à ensemble de recherche évolutif pour la réduction du profil des matrices creuses*, Rapport d'Etudes Libres, Aerospatiale Cannes, 1984-85. (A summarized version of this work was presented at the 4th International Symp. for Innovative Numer. Meth. in Engrg. 23-28 March 1986, Atlanta, GA, Computational Mechanics Publication, Springer-Verlag.)

(Pina, 1981) H. L. G. PINA, *An algorithm for frontwidth reduction*, Internat. J. Numer. Meth. Engrg., 17 (1981), pp. 1539-1546.

(Snay, 1976) R. A. SNAY, *Reducing the profile of sparse symmetric matrices*, Internat. Bull. Géodésique, 50 (1975), pp. 341-352.

SIAM J. SCI. STAT. COMPUT.
Vol. 7, No. 4, October 1986

011

THE OPERATOR COMPACT IMPLICIT METHOD FOR FOURTH ORDER ORDINARY DIFFERENTIAL EQUATIONS*

JEFFREY C. BUELL†

Abstract. Two fourth order accurate approximations to general linear fourth order two-point boundary value problems with Dirichlet boundary conditions are evaluated. The first is a new implementation of the operator compact implicit (OCI) method. Its derivation will be given along with proofs of theoretical convergence and stability properties. In particular, it will be shown that it has no formal cell Reynolds number limitations. The second is centered second order differencing with Richardson extrapolation. Overall computational efficiency and the results of several numerical tests are discussed. Some comments are made on extending the OCI method to other boundary conditions and to nonlinear problems.

Key words. operator compact implicit, Richardson extrapolation, ordinary differential equations, truncation error

AMS(MOS) subject classification. 65L10

1. Introduction.

We consider here fourth order approximations to the linear two-point boundary value problem

$$y^{(4)}(x) = f(x, y(x), y^{(1)}(x), y^{(2)}(x), y^{(3)}(x)) \tag{1.1}$$
$$= g(x) - a(x)y(x) - b(x)y^{(1)}(x) - c(x)y^{(2)}(x) - d(x)y^{(3)}(x),$$

with the Dirichlet boundary conditions

$$y^{(1)}(0) = B_1, \qquad y(0) = B_2, \tag{1.2}$$
$$y^{(1)}(1) = B_3, \qquad y(1) = B_4,$$

where $y \in \mathbb{R}$, $x \in [0, 1]$, and $y^{(k)}(x)$ denotes the kth derivative of y. We derive first the operator compact implicit (OCI) method for this problem, and then review centered second order (SO) differencing with Richardson extrapolation (RE) to fourth order. It is the intent here to evaluate the OCI method with respect to the RE method.

In implementing standard schemes for solving (1.1), (1.2), each derivative is represented independently. The OCI method presented here, on the other hand, takes advantage of the operator f so that each derivative may be approximated to higher order without expanding the mesh star. The method is "implicit" since values of the operator are required at three points of the five point star.

A possible alternative to OCI or RE methods is to approximate each derivative individually to fourth order using centered differencing. However, it soon becomes apparent that there are many disadvantages to this procedure. The scheme yields a seven band matrix which requires more than twice the arithmetic to solve than a five band matrix and results in an approximation that is considerably less accurate than the RE method. At the boundaries, either noncentered differencing is required for the two highest derivatives (which increases the truncation error by an order of magnitude and destroys the matrix structure) or fictitious points must be created. Finally, it can be shown that oscillations may be introduced at a relatively small cell Reynolds number. Another possibility is to implement a scheme based on deferred corrections, but this has many of the same disadvantages. Neither of these methods is considered further.

* Received by the editors March 1984 and in revised form August 1985.

† Department of Mechanical, Aerospace and Nuclear Engineering, University of California, Los Angeles, California 90024.

Early research on high order compact difference schemes was performed by Osborne [1] in 1967. The coefficients in his approximations were determined by requiring that a certain set of interpolation functions satisfy both the differential equation and difference equations. Since then, his analysis was generalized by Doedel [2], and Lynch and Rice [3], among others. Swartz [4] complements the latter paper by providing suitable approximations for general side conditions. These papers considered arbitrary order operators, however they were implemented only for second order operators. Also, they require the solution of a set of algebraic equations at each grid point in order to construct the difference equations. Furthermore, they usually require evaluating the coefficients of the differential equation at points between the mesh points. If the coefficients are not analytic functions (that is, they are known only at the grid points), then the additional step of defining and evaluating a high order interpolation function will have to be made.

Much work has been done on the second order problem corresponding to (1.1), (1.2) (see, for example, [1]-[8]). Ciment et al. [5] traced the development of "compact implicit" schemes for second order operators. They solved a 3×3 block tridiagonal system of equations for a fourth order approximation to the dependent variable and its first two derivatives. Alternatively, the second derivative can be eliminated to yield a 2×2 block tridiagonal system. However, this formalism does not allow reduction to a scalar tridiagonal system. They then developed an OCI scheme for parabolic equations by finding a relationship between the spatial operator and the dependent variable and substituting the result into the differential equation. A fourth order accurate tridiagonal system is obtained, but the method has a cell Reynolds number limitation of $R \leqq 12^{1/2}$. A multi-parameter family of OCI methods are developed by Berger et al. [6], where they allowed certain error terms to be $O(h^4)$, rather than zero. By varying these parameters, the methods can be optimized for high cell Reynolds number problems. The above methods, as well as the one considered here, are all based on Taylor series expansions of the truncation error. They are also of "polynomial type" since the difference equation coefficients are polynomials in the differential equation coefficients and the mesh spacing. It has been shown (see [6], [7], and references therein) that all polynomial type schemes yield $O(1)$ errors at the edges of boundary layers for moderate cell Reynolds numbers even though the methods may be formally stable. Uniform consistency can be achieved only by implementing a method of "exponential type," as is done by Leventhal [7]. Applying this idea to fourth order differential equations with high cell Reynolds numbers will be the subject of a future investigation.

These schemes are derived very differently from the OCI scheme of Stepleman [8]. The differential equation that he solved was in the form of (1.1) (that is, not parabolic), except that it was second order instead of fourth order. The operator in this case consists of everything except the highest order derivative and is used directly to find consistent approximations to even higher order derivatives. These derivatives are then substituted in Taylor series expansions to obtain fourth order approximations to the lower order derivatives. This procedure is followed here and will yield a fourth order accurate approximation to (1.1), (1.2) from the solution of a pentadiagonal matrix. In a similar manner, Katti [9] treated a nonlinear fourth order equation that did not contain any first, second, or third order derivatives. He was able to achieve sixth order accuracy with a five band matrix, but the lack of intermediate derivatives in his differential equation limits the usefulness of his results. For maximum simplicity of implementation, we require here that the matrix elements be defined analytically as a function of the differential equation coefficients evaluated only at the grid points, and the mesh spacing h. The schemes of Stepleman and Katti satisfy this requirement

with little modification when applied to linear equations. The formalism of Ciment et al. [5] and Berger et al. [6] could also be used to develop OCI schemes for fourth order equations. In some sense these are equivalent to the formalism of Stepleman, but here the individual derivatives are easily recovered, which is advantageous for solving nonlinear equations or systems of equations.

In § 2 we give some of the details of the derivation of the OCI method applied to (1.1), (1.2) and consider the local truncation error. Much of the notation here follows Stepleman [8]. All the proofs in § 2 are based on Taylor series expansion and straightforward (but sometimes lengthy) algebra, so most of them are omitted. In § 3 we obtain the stability results for the methods considered, and we compare their truncation errors. In § 4 the methods are tested on three representative problems. In § 5 we summarize our results and consider extensions of the OCI method to non-Dirichlet boundary conditions, and to nonlinear equations.

2. The operator compact implicit method. We would like to approximate (1.1), (1.2) to fourth order using a pentadiagonal matrix representation. In order to achieve our goal, Taylor series expansions are needed for various terms, and it will be seen that f and y must satisfy certain conditions. In particular, we assume that

(a) there exists a solution $y(x) \in C^8[I]$ to (1.1) and (1.2),

(b) f is bounded (that is, a, b, c, d, g are bounded),

(c) $b(x)$, $c(x)$, $d(x)$ are Lipschitz continuous (for example, $|b(x_2)-b(x_1)| \leqq L_b|x_2-x_1|$, $x_1, x_2 \in [0,1]$, L_b is the Lipschitz constant for $b(x)$).

Let $x_i=(i-1)h$, $h=1/(N-1)$, $i=1,\cdots,N$, and N some positive integer. Also let $y_i=y(x_i)$, $y_i^{(k)}=y^{(k)}(x_i)$ and $a_i=a(x_i)$, etc. Here, $C^n[I]$ is the space of n-times continuously differentiable functions on the unit interval $0 \leqq x \leqq 1$. We consider first centered discretizations that use no more than five grid function values. Define

$$y_i' = (y_{i+1}-y_{i-1})/2h, \tag{2.1}$$

$$y_i'' = (y_{i-1}-2y_i+y_{i+1})/h^2, \tag{2.2}$$

$$y_i'^0 = (y_{i-2}-8y_{i-1}+8y_{i+1}-y_{i+2})/12h, \tag{2.3}$$

$$y_i''^0 = (-y_{i-2}+16y_{i-1}-30y_i+16y_{i+1}-y_{i+2})/12h^2, \tag{2.4}$$

$$y_i'''^0 = (-y_{i-2}+2y_{i-1}-2y_{i+1}+y_{i+2})/2h^3, \tag{2.5}$$

$$y_i^{(iv)} = (y_{i-2}-4y_{i-1}+6y_i-4y_{i+1}+y_{i+2})/h^4. \tag{2.6}$$

Well known Taylor series arguments show

LEMMA 1. *Let $y \in C^8[I]$. Then*

$$y_i' = y_i^{(1)}+\frac{h^2}{6}y_i^{(3)}+\frac{h^4}{120}y_i^{(5)}+O(h^6),$$

$$y_i'' = y_i^{(2)}+\frac{h^2}{12}y_i^{(4)}+\frac{h^4}{360}y_i^{(6)}+O(h^6),$$

$$y_i'^0 = y_i^{(1)}-\frac{h^4}{30}y_i^{(5)}+O(h^6),$$

$$y_i''^0 = y_i^{(2)}-\frac{h^4}{90}y_i^{(6)}+O(h^6),$$

$$y_i'''^0 = y_i^{(3)}+\frac{h^2}{4}y_i^{(5)}+\frac{h^4}{40}y_i^{(7)}+O(h^6),$$

$$y_i^{(iv)} = y_i^{(4)} + \frac{h^2}{6} y_i^{(6)} + \frac{h^4}{80} y_i^{(8)} + O(h^4).$$

The higher order error terms are given here explicitly for later use. In § 3 we assume that $y \in C^9[I]$ so that the last error term in the last expression above becomes $O(h^6)$. Clearly, if we have second order approximations to the fifth and sixth derivatives that are contained in the five point mesh star, then (2.5) and (2.6) can be corrected to fourth order. We introduce some noncentered differencing first. Define

$$y_i'^{-} = (-3y_{i-1} - 10y_i + 18y_{i+1} - 6y_{i+2} + y_{i+3})/12h, \tag{2.7}$$

$$y_i'^{+} = (-y_{i-3} + 6y_{i-2} - 18y_{i-1} + 10y_i + 3y_{i+1})/12h, \tag{2.8}$$

$$y_i''^{-} = (11y_{i-1} - 20y_i + 6y_{i+1} + 4y_{i+2} - y_{i+3})/12h^2, \tag{2.9}$$

$$y_i''^{+} = (-y_{i-3} + 4y_{i-2} + 6y_{i-1} - 20y_i + 11y_{i+1})/12h^2, \tag{2.10}$$

$$y_i'''^{-} = (-3y_{i-1} + 10y_i - 12y_{i+1} + 6y_{i+2} - y_{i+3})/2h^3, \tag{2.11}$$

$$y_i'''^{+} = (y_{i-3} - 6y_{i-2} + 12y_{i-1} - 10y_i + 3y_{i+1})/2h^3. \tag{2.12}$$

A Taylor series expansion of the terms on the r.h.s. of (2.7) shows that

$$y_i'^{-} = y_i^{(1)} + \frac{h^4}{20} y_i^{(5)} + O(h^5).$$

Later we will need $y_{i-1}'^{-}$, but with the error term evaluated at x_i. Thus we use another Taylor series argument to obtain

$$y_i'^{-} = y_i^{(1)} + \frac{h^4}{20} y_{i+1}^{(5)} + O(h^5).$$

A similar analysis of (2.8) through (2.12) yields

LEMMA 2. *Let $y \in C^8[I]$. Then*

$$y_i'^{-} = y_i^{(1)} + \frac{h^4}{20} y_{i+1}^{(5)} + O(h^5),$$

$$y_i'^{+} = y_i^{(1)} + \frac{h^4}{20} y_{i-1}^{(5)} + O(h^5),$$

$$y_i''^{-} = y_i^{(2)} - \frac{h^3}{12} y_{i+1}^{(5)} + \frac{11}{360} h^4 y_{i+1}^{(6)} + O(h^5),$$

$$y_i''^{+} = y_i^{(2)} + \frac{h^3}{12} y_{i-1}^{(5)} + \frac{11}{360} h^4 y_{i-1}^{(6)} + O(h^5),$$

$$y_i'''^{-} = y_i^{(3)} - \frac{h^2}{4} y_{i+1}^{(5)} - \frac{h^4}{60} y_{i+1}^{(7)} + O(h^5),$$

$$y_i'''^{+} = y_i^{(3)} - \frac{h^2}{4} y_{i-1}^{(5)} - \frac{h^4}{60} y_{i-1}^{(7)} + O(h^5).$$

We note that in the last two expressions of Lemma 2 there are no terms proportional to h^3, which is unusual for noncentered differencing. We are now ready to use the operator f to find the higher order derivatives we need in Lemma 1. Let

$$f_i^{+} = f(x_i, y_i, y_i', y_i'', y_i'''^{+}), \tag{2.13}$$

$$f_i^- = f(x_i, y_i, y_i', y_i'', y_i'''^-), \tag{2.14}$$

$$y_i^{(v)} = (f_{i+1}^+ - f_{i-1}^-)/2h. \tag{2.15}$$

Substituting (2.13) and (2.14) into (2.15), using Lemmas 1 and 2, and (1.1) yields

$$y_i^{(v)} = \left[y_{i+1}^{(4)} - y_{i-1}^{(4)} - \frac{h^2}{6}(b_{i+1}y_{i+1}^{(3)} - b_{i-1}y_{i-1}^{(3)}) - \frac{h^2}{12}(c_{i+1}y_{i+1}^{(4)} - c_{i-1}y_{i-1}^{(4)}) + \frac{h^2}{4}(d_{i+1}y_i^{(5)} - d_{i-1}y_i^{(5)}) \right] \Big/ 2h + O(h^3).$$

Taylor series expansions give

$$y_i^{(v)} = y_i^{(5)} + \frac{h^2}{6} y_i^{(7)} + \rho_i + O(h^3), \tag{2.16}$$

where

$$\rho_i = -\frac{h}{12}(b_{i+1} - b_{i-1})y_i^{(3)} - \frac{h^2}{12}(b_{i+1} + b_{i-1})y_i^{(4)} - \frac{h}{24}(c_{i+1} - c_{i-1})y_i^{(4)} - \frac{h^2}{24}(c_{i+1} + c_{i-1})y_i^{(5)} + \frac{h}{8}(d_{i+1} - d_{i-1})y_i^{(5)}.$$

If $b(x)$ is Lipschitz continuous, then $b_{i+1} - b_{i-1} \sim O(h)$. With a similar argument for $c(x)$ and $d(x)$ we obtain

LEMMA 3. *Let $y \in C^8[I]$ and $b(x)$, $c(x)$, $d(x)$ be Lipschitz continuous. Then*

$$\rho_i = O(h^2).$$

We proceed to find a second order approximation to $y_i^{(6)}$. Define

$$y_i^{t0} = y_i'''^0 - \frac{h^2}{4} y_i^{(v)}, \tag{2.17}$$

$$y_i^{t-} = y_i'''^- + \frac{h^2}{4} y_{i+1}^{(v)}, \tag{2.18}$$

$$y_i^{t+} = y_i'''^+ + \frac{h^2}{4} y_{i-1}^{(v)}. \tag{2.19}$$

Lemmas 2 and 3 combined with (2.17) through (2.19) give directly

LEMMA 4. *Let the hypothesis of Lemma 3 hold. Then*

$$y_i^{t0} = y_i^{(3)} + O(h^4),$$
$$y_i^{t-} = y_i^{(3)} + O(h^4),$$
$$y_i^{t+} = y_i^{(3)} + O(h^4).$$

We need four more definitions:

$$f_i^{t0} = f(x_i, y_i, y_i'^0, y_i''^0, y_i^{t0}), \tag{2.20}$$

$$f_i^{t-} = f(x_i, y_i, y_i'^-, y_i''^-, y_i^{t-}), \tag{2.21}$$

$$f_i^{t+} = f(x_i, y_i, y_i'^+, y_i''^+, y_i^{t+}), \tag{2.22}$$

$$y_i^{(vi)} = (f_{i-1}^{t-} - 2f_i^{t0} + f_{i+1}^{t+})/h^2. \tag{2.23}$$

Substitute (2.20) through (2.22) into (2.23), and use the four Lemmas and (1.1) to obtain

$$y_i^{(vi)} = (y_{i-1}^{(4)} - 2y_i^{(4)} + y_{i+1}^{(4)} + h^2\sigma_i)/h^2 + O(h^2),$$

where

$$\sigma_i = -\frac{h^2}{60}(3b_{i-1}+4b_i+3b_{i+1})y_i^{(5)} - \frac{h}{12}(c_{i+1}-c_{i-1})y_i^{(5)}$$
$$-\frac{h^2}{360}(11c_{i-1}+8c_i+11c_{i+1})y_i^{(6)} - \frac{h^2}{120}(3d_{i-1}+4d_i+3d_{i+1})y_i^{(7)}$$
$$-\frac{1}{4}(d_{i-1}+2d_i+d_{i+1})\rho_i.$$

A Taylor series expansion yields

$$y_i^{(vi)} = y_i^{(6)} + \frac{h^2}{12}y_i^{(8)} + \sigma_i + O(h^2). \tag{2.24}$$

Again, the last term in (2.24) becomes $O(h^3)$ if $y \in C^9[I]$. We thus have

LEMMA 5. *Let the hypothesis of Lemma 3 hold. Then*

$$y_i^{(vi)} = y_i^{(6)} + O(h^2).$$

Combining (2.3), (2.4), (2.6), (2.17), and (2.23) with (1.1) yields our difference approximation to (1.1)

$$y_i^{(iv)} - \frac{h^2}{6}y_i^{(vi)} = f_i^{t0} + \tau_i,$$

or

$$y_i^{(iv)} - \tfrac{1}{6}(f_{i-1}^{t-} + 4f_i^{t0} + f_{i+1}^{t+}) = \tau_i, \tag{2.25}$$

where τ_i is the local truncation error. Our final local truncation error result follows from Lemmas 1 through 5.

THEOREM 6. *Let $y \in C^8[I]$ and $b(x)$, $c(x)$, $d(x)$ be Lipschitz continuous. Then*

$$\tau_i = O(h^4).$$

In order to implement (2.25) we set $\tau_i = 0$ and let v_i be an approximation to y_i. After substitution of the appropriate definitions and rearrangement we have a set of equations of the form

$$m_{i,i-2}v_{i-2} + m_{i,i-1}v_{i-1} + m_{i,i}v_i + m_{i,i+1}v_{i+1} + m_{i,i+2}v_{i+2} = r_i, \qquad i = 2, \cdots, N-1, \tag{2.26}$$

where the matrix elements $m_{i,j}$ and the right-hand side vector elements r_i are given in the Appendix, in terms of the coefficients of (1.1) and h.

We turn now to the treatment of the derivative boundary conditions (1.2). We cannot use (2.3) because it introduces too many image points. Use of (2.7) and (2.8) will yield a fourth order approximation, but we found from numerical experiments that the truncation error (Lemma 2) leads to a global error larger than that due to the interior discretization alone. We thus use the approximation to the fifth derivative (2.15) to improve the truncation error of (2.7) and (2.8) to fifth order without expanding the mesh star:

$$y^{(1)}(0) = y_1'^{-} - \frac{h^4}{20}y_2^{(v)} + O(h^5), \tag{2.27}$$

$$y^{(1)}(1) = y'^{+}_{N} - \frac{h^4}{20} y^{(v)}_{N-1} + O(h^5). \tag{2.28}$$

Both of these approximations are written out in full in the Appendix. As is common in finite difference methods, these two equations are used to eliminate the image points v_0 and v_{N+1} from the two interior equations (2.26) that contain them, yielding the desired pentadiagonal matrix. We note that even though (2.27) and (2.28) are fifth order accurate, the size of the truncation error is still comparable to that of the interior approximation (for reasonable h). This is due to their one-sided nature, and appears to be unavoidable. However, the error term is proportional to the sixth derivative, so (2.23) can be used to correct (2.27) and (2.28) to sixth order, but this probably would not help much.

The other commonly encountered Dirichlet boundary condition is obtained by replacing the first derivative in (1.2) by the second derivative. The procedure and comments above still apply, except that the order of accuracy is reduced by one. That is, (2.15) is used to correct (2.9) and (2.10) to fourth order accuracy, and (2.23) is used in addition for fifth order.

3. Accuracy and stability of the methods. In this section we compare the local truncation error and stability of the OCI and RE methods. Even though $y \in C^8[I]$ is sufficient for $O(h^4)$ accuracy for both methods, we assume that $y \in C^9[I]$ so that the coefficient of the h^4 error term is well defined. The results of the previous section yield the local truncation error for the OCI method (suppressing the subscript i):

$$\tau_{\mathrm{OCI}} = -\frac{h^4}{720} y^{(8)} - \frac{h^4 d}{60} y^{(7)} - \frac{h^4 c}{90} y^{(6)} - \frac{h^4 b}{30} y^{(5)} - \frac{h^2}{6}\sigma - \frac{h^2 d}{4}\rho + O(h^5). \tag{3.1}$$

We can simplify ρ and σ by noting (from Assumption (c)) that $b_{i+1} + b_{i-1} = 2b_i + O(h)$, etc. Thus, eliminating ρ and σ, we write (3.1) as

$$\begin{aligned} \tau_{\mathrm{OCI}} = h^4 \Bigg[& -\frac{1}{720} y^{(8)} - \frac{d}{360} y^{(7)} + \frac{c}{360} y^{(6)} + \frac{\Delta c}{72h} y^{(5)} - \frac{b}{180} y^{(5)} \\ & + \frac{d}{288}\left(-3\frac{\Delta d}{h} y^{(5)} + 2cy^{(5)} + \frac{\Delta c}{h} y^{(4)} + 4by^{(4)} + 2\frac{\Delta b}{h} y^{(3)} \right) \Bigg] + O(h^5), \end{aligned} \tag{3.2}$$

where $\Delta b = \Delta b_i = b_{i+1} - b_{i-1}$, etc.

For the RE method (see Keller [10]), we substitute (2.1), (2.2), (2.5), (2.6) into (1.1) and solve the resulting system twice, once on a h-grid and once on a $\frac{1}{2}h$-grid. The two solutions are then averaged in such a way that the $O(h^2)$ errors cancel. That is, if $v(h)$ is the approximate solution on the h-grid, we take

$$v = (4v(\tfrac{1}{2}h) - v(h))/3. \tag{3.3}$$

The resulting local truncation error is easily seen to be

$$\tau_{\mathrm{RE}} = h^4 \left[-\frac{1}{320} y^{(8)} - \frac{d}{160} y^{(7)} - \frac{c}{1440} y^{(6)} - \frac{b}{480} y^{(5)} \right] + O(h^6). \tag{3.4}$$

We note that if $b = c = 0$ and $d = \text{const.}$, then

$$\tau_{\mathrm{OCI}} = \tfrac{4}{9}\tau_{\mathrm{RE}}. \tag{3.5}$$

Otherwise, we cannot make a direct comparison but it is evident that if b, c, and d are large and variable, τ_{OCI} may be many times larger than τ_{RE}.

We now consider the spatial stability of the two methods. A method is considered to be spatially stable if the numerical solution exhibits the same qualitative behavior as the exact solution. In particular, instability is characterized by oscillatory solutions or a growth rate of the opposite sign as the true solution. We use the test problem

$$y^{(4)} + dy^{(3)} = 0, \qquad d = \text{const.} > 0, \tag{3.6}$$

which has solutions of the form

$$y = K_1 e^{-dx} + K_2 x^2 + K_3 x + K_4, \tag{3.7}$$

where the K_i's depend on the boundary conditions. We assume that the numerical solution varies like

$$v_i \sim \kappa^i \tag{3.8}$$

and substitute it into the difference equations. For the OCI method, we have (from the Appendix)

$$\begin{aligned}(1+\tfrac{1}{2}R+\tfrac{1}{12}R^2)\kappa^4 - (4+R+\tfrac{1}{3}R^2)\kappa^3 + (6+\tfrac{1}{2}R^2)\kappa^2 \\ - (4-R+\tfrac{1}{3}R^2)\kappa + (1-\tfrac{1}{2}R+\tfrac{1}{12}R^2) = 0,\end{aligned} \tag{3.9}$$

where we have multiplied through by κ^{2-i} and have defined the cell Reynolds number $R = dh$. The above can be factored to yield

$$\left(\kappa - 1 + \frac{12R}{12+6R+R^2}\right)(\kappa-1)^3 = 0. \tag{3.10}$$

The four roots of (3.10) are

$$\kappa_1 = \kappa_{\mathrm{OCI}} = 1 - \frac{12R}{12+6R+R^2}, \qquad \kappa_2 = \kappa_3 = \kappa_4 = 1. \tag{3.11}$$

Clearly, the κ_i's correspond to the solutions in (3.7). For second order differencing the same procedure yields

$$\kappa_{\mathrm{SO}} = \kappa_1 = 1 - \frac{2R}{2+R}, \qquad \kappa_2 = \kappa_3 = \kappa_4 = 1. \tag{3.12}$$

If we ignore the last three κ_i's, then the RE method applied to (3.8) using (3.12) yields

$$v_i = \frac{4}{3}\left(1 - \frac{2R}{4+R}\right)^{2i} - \frac{1}{3}\left(1 - \frac{2R}{2+R}\right)^{i}$$

or

$$\kappa_1 = \left[\frac{4}{3}\left(1 - \frac{2R}{4+R}\right)^{2i} - \frac{1}{3}\left(1 - \frac{2R}{2+R}\right)^{i}\right]^{1/i}. \tag{3.13}$$

Taking $i = 1$ we define

$$\kappa_{\mathrm{RE}} = \frac{4}{3}\left(1 - \frac{2R}{4+R}\right)^{2} - \frac{1}{3}\left(1 - \frac{2R}{2+R}\right). \tag{3.14}$$

The exact solution corresponding to κ_{OCI} and κ_{RE} is found by letting $K_2 = K_3 = K_4 = 0$, $K_1 = 1$, and can be written

$$y_i = e^{-Ri}. \tag{3.15}$$

We thus want to compare κ_{OCI}, κ_{RE} and e^{-R}. These three quantities plus κ_{SO} are plotted in Fig. 1. As shown there, centered second order differencing has the well known cell Reynolds number limitation of $R \leqq 2$ due to oscillations. Richardson extrapolation relieves that limitation but introduces another at $R = 12 + 8\sqrt{3} \simeq 25.86$ where the numerical solution (3.8) begins to grow instead of continuing to decay. The OCI method has no cell Reynolds number limitation in the sense that the numerical solution to (3.6) (with the appropriate boundary conditions) will monotonically decay for all R. However, it is noted that neither of the two methods will be accurate beyond the minimum of their respective κ vs. R curve (these minima occur at about $R = 3.14$ for the RE method, and $R = 3.46$ for the OCI method). But, we see that the behavior of the RE method may be fortuitous: The form of κ_{RE} may depend strongly on (1.1). In particular, we would expect that for some problems with a large cell Reynolds number ($R > 2$) the oscillations from the second order method will carry over through the RE method, instead of canceling out as they do for (3.6). On the other hand, the OCI method does not rely on cancellation of oscillations so the stability results obtained here should be more universal. Thus, we expect the OCI method to be, in general, more "robust."

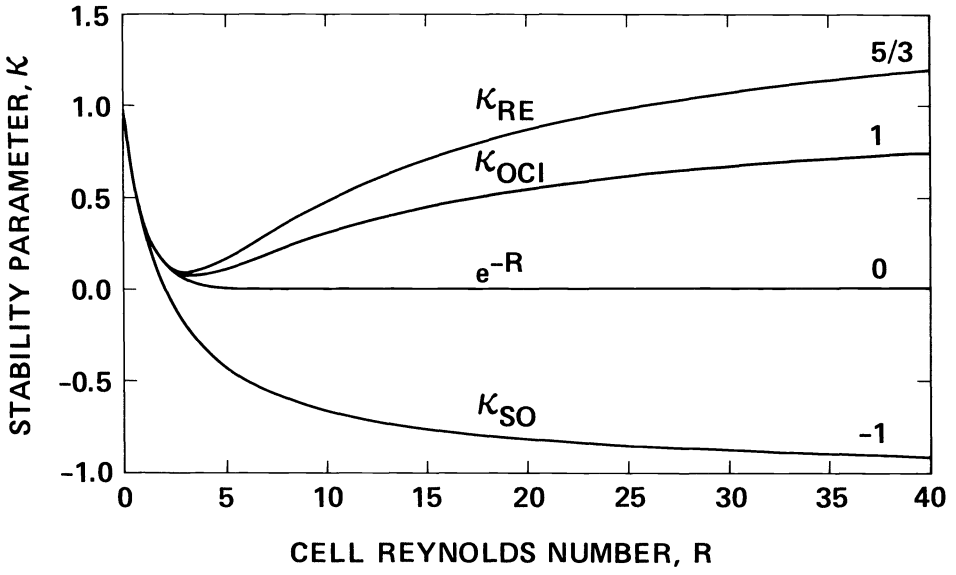


FIG. 1. κ *vs.* R *for the* OCI, RE *and* SO *methods. Values of* κ *as* $R \to \infty$ *are shown at the right.*

4. Numerical tests. Here we compare the OCI and RE methods in terms of computational efficiency and accuracy for three sample problems.

Shown in Table 1 is a comparison of the amount of work required to solve (1.1), (1.2) for fixed N. The number of coefficient evaluations for the RE method depends on whether the evaluated coefficients can be saved from the first pass through the second order method to the second. For many applications this is not possible or practical, which results in $15N$ coefficient evaluations. If these evaluations require few multiplications, then the total arithmetic for the two methods is very comparable. However, if the coefficients are expensive to compute (for example, requiring external function calls) then the RE method may require up to two or three times the arithmetic as the OCI method.

TABLE 1
Work comparison for the OCI, RE *and* SO *methods.*

	OCI	RE	SO
Coefficient evaluations	$5N$	$10N$–$15N$	$5N$
Multiplications to construct difference approximation	$44N$	$18N$	$6N$
Multiplications and divisions to solve algebraic equations	$10N$	$30N$	$10N$

We now consider some sample problems. The following differential equations with the given boundary conditions were solved using both methods

(A) $$y^{(4)}+x^5y^{(3)}-7x^4y^{(2)}-7x^3y^{(1)}-10x^2y=5040x^6+10x^{12},$$
$$y(0)=y^{(1)}(0)=0,\quad y(1)=1,\quad y^{(1)}(1)=10;$$

(B) $$y^{(4)}-3\pi\sin(2\pi x)y^{(3)}+6\pi^2\cos(2\pi x)y^{(2)}-16\pi^4y=4\pi^4,$$
$$y(0)=y^{(1)}(0)=y(1)=y^{(1)}(1)=0;$$

(C) $$y^{(4)}+2\alpha(x-\tfrac{1}{2})y^{(3)}+12\alpha y^{(2)}+12\alpha^2(x-\tfrac{1}{2})y^{(1)}+12\alpha^2y=0,\qquad \alpha=22,$$
$$y(0)=e^{-\alpha/4},\quad y^{(1)}(0)=\alpha e^{-\alpha/4},\quad y(1)=e^{-\alpha/4},\quad y^{(1)}(1)=-\alpha e^{-\alpha/4}.$$

The solutions are

(A) $$y=x^{10},$$

(B) $$y=\sin^2(\pi x),$$

(C) $$y=e^{-\alpha(x-1/2)^2},\qquad \alpha=22.$$

All computations were performed on an IBM 3033 in double precision. Table 2 lists the maximum error for the two methods and for second order differencing, for each problem and three different N. The numbers in parentheses refer to the exponents, for example, $1.3(-3)$ means 1.3×10^{-3}. The results from (A) and (B) indicate that the first two methods are globally fourth order accurate and have comparable absolute accuracy. Problem (C) is somewhat pathological in that the differential equation is homogeneous and the boundary conditions are very small; but it has an $O(1)$ solution. In other

TABLE 2
Accuracy comparison for the OCI, RE *and* SO *methods.*

		Error, max-norm		
Problem	N	OCI	RE	SO
A	11	1.3 (−3)	1.8 (−3)	1.0 (−1)
	21	6.5 (−5)	1.2 (−4)	2.7 (−2)
	41	3.1 (−6)	7.4 (−6)	6.8 (−3)
B	11	6.2 (−3)	1.6 (−2)	3.0 (−1)
	21	4.9 (−4)	8.0 (−4)	6.4 (−2)
	41	3.0 (−5)	4.8 (−5)	1.5 (−2)
C	21	2.4 (−2)	3.3 (−1)	8.2 (−1)
	41	1.3 (−3)	6.7 (−2)	4.5 (−1)
	81	8.0 (−5)	6.8 (−3)	1.6 (−1)

words, it is "nearly" singular. Both methods can solve this problem for any α and any desired degree of accuracy provided N is sufficiently large. However, the OCI method can solve (C) to within 2.4 per cent with 21 grid points while the RE method requires about 57 for the same accuracy. For large N the former is about 100 times as accurate as the latter. The RE method ran into trouble with $N=21$ even though the maximum cell Reynolds number was only about one. As one would expect, the OCI method is favored even more if the problem parameter "α" is larger (that is, the problem is closer to being singular), while the methods are comparable for smaller α. The conclusion is that (as we conjectured in § 3) for some problems the RE method will not work properly if the second order differencing does not result in at least qualitatively correct solutions, while the OCI method does not require this. Furthermore, an examination of the local truncation error cannot be used reliably to predict the relative accuracy of the methods. For problems (A) and (B), the maximum of τ_{OCI} is about four times larger than the maximum of τ_{RE}, and even for (C), the former is somewhat greater than the latter.

We note a characteristic of the present implementation of the OCI method that was alluded to earlier. Many problems were solved with a wide variety of N in order to gain experience with the method. In most cases it was observed that the order was actually a bit greater than four, while the RE method gave very consistent fourth order convergence. This is because the truncation errors due to the boundary and interior discretizations are about the same size, but the former is fifth order while the latter is fourth order accurate.

5. Summary. We have presented here an implementation of the OCI method for solving general linear fourth order ordinary differential equations, and have compared it to standard second order differencing with Richardson extrapolation. In particular, for some problems involving large and variable coefficients, the latter will have a smaller local truncation error. However, the numerical examples showed that this does not necessarily lead to a smaller global error; it still remains to find a strong relationship between the local and global truncation errors in order to explain the observed behavior. An operation count showed that if the coefficients are expensive to evaluate, then the OCI method will require less arithmetic for a fixed number of grid points. Also, this method will be preferred for problems where the cell Reynolds number is not small and for problems where centered second order differencing does not yield at least qualitatively correct solutions. For high cell Reynolds number problems neither method is good (from Fig. 1, we expect very diffusive-looking solutions and $O(1)$ errors). For this case, it would be very worthwhile to develop a method of exponential type analogous to the schemes considered in [7] for second order operators.

We have not considered non-Dirichlet boundary conditions or nonlinear differential equations, but these are straightforward extensions of the work presented here. For the former, third order derivatives are introduced into the boundary conditions. The differential equation is required to be satisfied on the boundaries, and an additional image point is introduced. This allows us to approximate the boundary conditions in the same manner as the differential equation derivatives (that is, using (2.3), (2.4) and (2.17)). Since the noncentered approximations (2.7)-(2.10) are not used, we expect the method to perform even better than for Dirichlet conditions. The behavior of centered second order differencing (and thus RE) does not depend, in general, on the type of boundary condition imposed. The OCI method for non-Dirichlet boundary conditions assumes that (1.1) is defined at $x=-h$ and $x=1+h$. The RE method does not require (1.1) to be defined outside of $[0, 1]$, which may be helpful for some problems.

Nonlinear differential equations are solved most easily by quasilinearization. We require existence and certain smoothness of the Frechet derivatives of f. The coefficients of (1.1) then become functions of the derivatives of y at the previous iteration as well as of x. These derivatives must be evaluated with fourth order accuracy, thus, (2.1) through (2.19) should be used. Recent results with a system of nonlinear differential equations (solved one at a time and coupled by iteration) have shown that the overall efficiency of the OCI method and ease of its implementation carry over to more complex problems.

Appendix. The matrix elements (2.26) for the OCI method are given here in terms of the coefficients of (1.1) and the mesh spacing h.

$$m_{i,i-2}=1-\frac{h}{2}d_i+\Big[-\frac{h}{12}(3d_{i-1}-2d_i-d_{i+1})+\frac{h^2}{72}(11c_{i-1}-4c_i-c_{i+1})$$
$$-\frac{h^3}{72}(3b_{i-1}-4b_i+b_{i+1})+z_i(-3d_{i-1}-d_{i+1}+2hc_{i-1}-h^2b_{i-1})\Big],$$

$$m_{i,i-1}=-4+hd_i+h^2c_i-\frac{h^3}{2}b_i+\Big[\frac{h}{6}(5d_{i-1}-2d_i-3d_{i+1})+\frac{h^2}{18}(-5c_{i-1}-2c_i+c_{i+1})$$
$$+\frac{h^3}{36}(-5b_{i-1}+2b_i+3b_{i+1})+\frac{h^4}{6}a_{i-1}$$
$$+2z_i(5d_{i-1}+3d_{i+1}-2hc_{i-1}+h^3a_{i-1})\Big],$$

$$m_{i,i}=6-2h^2c_i+h^4a_i+\Big[h(d_{i+1}-d_{i-1})+\frac{h^2}{12}(c_{i-1}+4c_i+c_{i+1})+\frac{h^3}{4}(b_{i-1}-b_{i+1})$$
$$-\frac{h^4}{3}a_i+z_i(-12(d_{i-1}+d_{i+1})+2h(c_{i-1}-c_{i+1})+h^2(b_{i-1}+b_{i+1}))\Big],$$

$$m_{i,i+1}=-4-hd_i+h^2c_i+\frac{h^3}{2}b_i+\Big[\frac{h}{6}(3d_{i-1}+2d_i-5d_{i+1})+\frac{h^2}{18}(c_{i-1}-2c_i-5c_{i+1})$$
$$+\frac{h^3}{36}(-3b_{i-1}-2b_i+5b_{i+1})+\frac{h^4}{6}a_{i+1}$$
$$+2z_i(3d_{i-1}+5d_{i+1}+2hc_{i+1}-h^3a_{i+1})\Big],$$

$$m_{i,i+2}=1+\frac{h}{2}d_i+\Big[-\frac{h}{12}(d_{i-1}+2d_i-3d_{i+1})+\frac{h^2}{72}(-c_{i-1}-4c_i+11c_{i+1})$$
$$+\frac{h^3}{72}(b_{i-1}-4b_i+3b_{i+1})+z_i(-d_{i-1}-3d_{i+1}-2hc_{i+1}-h^2b_{i+1})\Big],$$

$$r_i=h^4g_i+\Big[\frac{h^4}{6}(g_{i-1}+2g_i+g_{i+1})-2h^3z_i(g_{i+1}-g_{i-1})\Big],$$

where

$$z_i=\frac{h^2}{96}(d_{i-1}-4d_i+d_{i+1}) \quad \text{and} \quad 2\leqq i\leqq N-1.$$

We note that centered second order differencing is recovered if everything within the square brackets is deleted. The matrix elements for the nonderivative boundary conditions are

$$m_{1,1}=1, \qquad r_1=B_2,$$
$$m_{N,N}=1, \qquad r_N=B_4.$$

The approximations using (2.27) and (2.28) to the derivative boundary conditions are

$$\begin{aligned} v_0\left[-3+\frac{3h}{20}(3d_1+d_3-2hc_1+h^2b_1)\right] \\ +v_1\left[-10-\frac{3h}{10}(5d_1+3d_3-2hc_1+h^3a_1)\right] \\ +v_2\left[18+\frac{3h}{20}(12(d_1+d_3)+2h(c_3-c_1)-h^2(b_1+b_3))\right] \\ +v_3\left[-6-\frac{3h}{10}(3d_1+5d_3+2hc_3-h^3a_3)\right] \\ +v_4\left[1+\frac{3h}{20}(d_1+3d_3+2hc_3+h^2b_3)\right] \\ =12hB_1+\frac{3}{10}h^4(g_3-g_1), \end{aligned}$$

$$\begin{aligned} v_{N-3}\left[-1+\frac{3h}{20}(3d_{N-2}+d_N-2hc_{N-2}+h^2b_{N-2})\right] \\ +v_{N-2}\left[6-\frac{3h}{10}(5d_{N-2}+3d_N-2hc_{N-2}+h^3a_{N-2})\right] \\ +v_{N-1}\left[-18+\frac{3h}{20}(12(d_{N-2}+d_N)+2h(c_N-c_{N-2})-h^2(b_{N-2}+b_N))\right] \\ +v_N\left[10-\frac{3h}{10}(3d_{N-2}+5d_N+2hc_N-h^3a_N)\right] \\ +v_{N+1}\left[3+\frac{3h}{20}(d_{N-2}+3d_N+2hc_N+h^2b_N)\right] \\ =12hB_3+\frac{3}{10}h^4(g_N-g_{N-2}). \end{aligned}$$

Acknowledgment. The author would like to thank James M. McDonough for helpful comments at an early stage of this work.

REFERENCES

[1] M. R. OSBORNE, *Minimising truncation error in finite difference approximations to ordinary differential equations*, Math. Comp., 21 (1967), pp. 133-145.

[2] E. J. DOEDEL, *The construction of finite difference approximations to ordinary differential equations*, SIAM J. Numer. Anal., 15 (1978), pp. 450-465.

[3] R. E. LYNCH AND J. R. RICE, *A high order difference method for differential equations*, Math. Comp., 34 (1980), pp. 333-372.

[4] B. SWARTZ, *Compact, implicit difference schemes for a differential equation's side conditions*, Math. Comp., 35 (1980), pp. 733-746.

[5] M. CIMENT, S. H. LEVENTHAL AND B. C. WEINBERG, *The operator compact implicit method for parabolic equations*, J. Comp. Phys., 28 (1978), pp. 135-166.

[6] A. E. BERGER, J. M. SOLOMON, M. CIMENT, S. H. LEVENTHAL AND B. C. WEINBERG, *Generalized* OCI *schemes for boundary layer problems*, Math. Comp., 35 (1980), pp. 695-731.

[7] S. H. LEVENTHAL, *An operator compact implicit method of exponential type*, J. Comp. Phys., 46 (1982) pp. 138-165.

[8] R. S. STEPLEMAN, *Tridiagonal fourth order approximations to general two-point nonlinear boundary value problems with mixed boundary conditions*, Math. Comp., 30 (1976), pp. 92-103.

[9] C. P. KATTI, *Five-diagonal sixth order methods for two-point boundary value problem involving fourth order differential equations*, Math. Comp., 35 (1980), pp. 1177-1179.

[10] H. B. KELLER, *Numerical Methods for Two-Point Boundary Value Problems*, Blaisdell, Waltham, MA, 1968.

SIAM J. SCI. STAT. COMPUT.
Vol. 7, No. 4, October 1986

012

AN INVESTIGATION INTO THE STABILITY PROPERTIES OF SECOND DERIVATIVE METHODS USING PERFECT SQUARE ITERATION MATRICES*

C. A. ADDISON† AND P. M. HANSON†

Abstract. This paper investigates the absolute stability properties of numerical methods for initial value O.D.E.'s based on second derivative formulae where the regular iteration matrix, which is of the form $(I-\beta_0 hJ-\gamma_0 h^2J^2)$, is approximated by a perfect square matrix of the form $(aI-bhJ)^2$. For the problem $y'(t)=\lambda y(t)$, $y(t_0)$ specified, it is shown that the region of stability depends not only upon the underlying implicit formula, but also upon the way in which y' and y'' are determined, the choice of a and b, the number of iterations, and the type of predictor formula used. In some cases, the method has a bounded region of stability while the underlying formula is A-stable. The stability regions of methods using different values of a and b, with a varying number of iterations are presented.

Key words. stiff ordinary differential equations, second derivative formulae, absolute stability

AMS(MOS) subject classification. 65L05

1. Introduction. Second derivative and related formulae (such as the blended formulae of Skeel and Kong [12]) have been put forward frequently as candidate formulae for numerical methods intended to solve stiff initial value ordinary differential equations (O.D.E.'s) (see for example, Enright [6], Skeel and Kong [12], and England [5]). The general form of a second derivative formula is

$$\mathbf{y}_i=\sum_{j=1}^{k}\alpha_j\mathbf{y}_{i-j}+h\sum_{j=0}^{k}\beta_j\mathbf{y}'_{i-j}+h^2\sum_{j=0}^{k}\gamma_j\mathbf{y}''_{i-j}. \tag{1.1}$$

To be suitable for stiff O.D.E.'s, the formula must at least be stable at infinity, and hence must be implicit (see Dahlquist [4]), with $\gamma_0\neq 0$. The implicit nature of the formulae means that a system of nonlinear equations must be solved on each step. This is done using a Newton-like iteration, with an explicit formula used as a predictor to start the iteration. On the autonomous problem

$$\mathbf{y}'(t)=\mathbf{f}(\mathbf{y}(t)),\qquad \mathbf{y}(t_0)=\mathbf{y}_0, \tag{1.2}$$

each iteration involves solving a system of linear equations with a matrix of the form $(I-\beta_0 hJ-\gamma_0 h^2J^2)$, where J is an approximation to the Jacobian $\partial\mathbf{f}/\partial\mathbf{y}$. Forming J^2 is often undesirable, for example, when J is sparse J^2 may be dense. One way to avoid forming J^2 is to approximate $(I-\beta_0 hJ-\gamma_0 h^2J^2)$ by a perfect square matrix $(aI-bhJ)^2$, where a and b are real (an approximation with $a=1$ was first suggested by Skeel and Kong [12]), and then to solve two systems of linear equations with the same matrix $(aI-bhJ)$ in each case.

The purpose of this paper is to investigate the stability properties of second derivative methods that use a perfect square approximation to the iteration matrix.

The iteration scheme for second derivative formulae is discussed in detail in § 2. In § 3 the absolute stability properties of the methods are analysed. (A method is absolutely stable for a given step-size h and a given differential equation if the change due to a perturbation of size δ in one of the mesh values $\mathbf{y}_i$ is no larger than $K\delta$ in

* Received by the editors November 8, 1983, and in revised form May 20, 1985. This work was supported in part by the Natural Sciences and Engineering Research Council of Canada.

† Chr. Michelson Institute, Department of Science and Technology, Fantoftvegen 38, N-5036 Fantoft, Bergen, Norway.

all subsequent values $\mathbf{y}_j, j > i$, where K does not depend on j. For the differential equation $y' = \lambda y$, where λ is a complex constant, the region of absolute stability is that set of values of $h\lambda$ for which the method is absolutely stable. See Gear [8].) It is shown that the choice of predictor, the values of a and b, and the number of iterations carried out, all affect the stability properties of the methods. Indeed, in certain cases, the stability region associated with the method using a perfect square approximation is bounded, while the underlying implicit formula is A-stable. (A formula is A-stable if the region of absolute stability contains the entire left half $h\lambda$-plane.) Section 4 considers the rate of convergence of the methods for linear problems. Several illustrative examples of the stability regions that result when different predictors and different values of a and b are used with a varying number of iterations are presented in § 5. Practical implications and conclusions are discussed in § 6.

2. The iteration scheme. Consider the second derivative formulae

$$\text{(2.1)} \qquad \textit{Predictor}: \ \mathbf{y}_i^{(0)} = \sum_{j=1}^{k} \hat{\alpha}_j \mathbf{y}_{i-j} + h \sum_{j=1}^{k} \hat{\beta}_j \mathbf{y}'_{i-j} + h^2 \sum_{j=1}^{k} \hat{\gamma}_j \mathbf{y}''_{i-j},$$

$$\text{(2.2)} \qquad \textit{Corrector}: \ \mathbf{y}_i = \sum_{j=1}^{k} \alpha_j \mathbf{y}_{i-j} + h \sum_{j=0}^{k} \beta_j \mathbf{y}'_{i-j} + h^2 \sum_{j=0}^{k} \gamma_j \mathbf{y}''_{i-j},$$

applied to the initial value O.D.E.

$$\text{(2.3)} \qquad \mathbf{y}'(t) = \mathbf{f}(\mathbf{y}(t)), \quad t \in [t_0, t_f], \quad \mathbf{y}(t_0) = \mathbf{y}_0.$$

The solution of the corrector satisfies the nonlinear system of equations

$$\text{(2.4a)} \qquad \mathbf{F}(\mathbf{y}) = \mathbf{0},$$

where

$$\text{(2.4b)} \qquad \mathbf{F}(\mathbf{y}) = \mathbf{y} - h\beta_0 \mathbf{f}(\mathbf{y}) - h^2 \gamma_0 \frac{\partial \mathbf{f}}{\partial \mathbf{y}}(\mathbf{y})\mathbf{f}(\mathbf{y}) - \mathbf{g}_{i-1},$$

with

$$\text{(2.4c)} \qquad \mathbf{g}_{i-1} = \sum_{j=1}^{k} \alpha_j \mathbf{y}_{i-j} + h \sum_{j=1}^{k} \beta_j \mathbf{y}'_{i-j} + h^2 \sum_{j=1}^{k} \gamma_j \mathbf{y}''_{i-j}.$$

Applying Newton's method to (2.4) leads to the iteration scheme

$$\text{(2.5)} \qquad W^{(r)} \boldsymbol{\delta}^{(r)} = -\mathbf{F}(\mathbf{y}^{(r-1)}), \qquad r = 1, 2, \cdots, \qquad \mathbf{y}^{(0)} = \mathbf{y}_i^{(0)},$$

where

$$\boldsymbol{\delta}^{(r)} = \mathbf{y}^{(r)} - \mathbf{y}^{(r-1)},$$

$$W^{(r)} = \mathbf{F}'(\mathbf{y}^{(r-1)}) = I - \beta_0 h \frac{\partial \mathbf{f}}{\partial \mathbf{y}}(\mathbf{y}^{(r-1)}) - \gamma_0 h^2 \left\{ \frac{\partial \mathbf{f}}{\partial \mathbf{y}}(\mathbf{y}^{(r-1)}) \right\}^2 - \gamma_0 h^2 \frac{\partial^2 \mathbf{f}}{\partial \mathbf{y}^2}(\mathbf{y}^{(r-1)})\mathbf{f}(\mathbf{y}^{(r-1)}).$$

In practice a modified Newton's method is usually considered with $W^{(r)}$ replaced by

$$\text{(2.6)} \qquad W = I - \beta_0 h J - \gamma_0 h^2 J^2,$$

in which J is an approximation to the Jacobian matrix $(\partial \mathbf{f}/\partial \mathbf{y})(\mathbf{y}^{(r-1)})$, and the term involving $(\partial^2 \mathbf{f}/\partial \mathbf{y}^2)(\mathbf{y}^{(r-1)})$ is ignored, see Liniger and Willoughby [10] and Sacks-Davis [11].

Convergence of the iteration scheme (2.5) with $W^{(r)}$ replaced by W is seldom a problem and the simplification leads to substantial reductions in cost. To illustrate this, SECDER (see Addison [1]) was run on the nonlinear STIFF DETEST problems (see Enright and Pryce [7]) using three different iteration schemes. The first was the iteration scheme (2.5) with $W^{(r)}$. The analytic Jacobian was used and the $\partial^2\mathbf{f}/\partial\mathbf{y}^2$ was found using differences. The second version used $W^{(r)}$, ignoring the $\partial^2\mathbf{f}/\partial\mathbf{y}^2$ term. The final version used W in place of $W^{(r)}$. The W matrix was only re-evaluated when the step-size or order changed or when the rate of convergence was deemed inadequate.

Of these three versions it was found that the version using $W^{(r)}$ including the $\partial^2\mathbf{f}/\partial\mathbf{y}^2$ term and the version using $W^{(r)}$ neglecting $\partial^2\mathbf{f}/\partial\mathbf{y}^2$ required roughly the same number of steps for each test problem but the first version required about twice as much execution time as the second. The second version frequently required fewer steps than the third version, with W, but was almost always slower, typically requiring one-and-a-half times as much execution time. Thus the savings made in computation time by using W can be substantial, but there are still difficulties and explicitly forming W is undesirable in many situations.

The calculation of W involves forming J^2. This requires $O(n^3)$ operations, and the advantage of a matrix J with special structure, such as sparseness, is usually lost. Modifications of the iteration scheme which avoid forming J^2 and allow full exploitation of any special structure have been suggested. For example, Enright [6] factors W into $-\gamma_0(rI-hJ)(\bar{r}I-hJ)$ where $r=-(\beta_0/2\gamma_0)+i(-(\beta_0/2\gamma_0)^2-(1/\gamma_0))^{1/2}$, and uses complex arithmetic. This modification has the disadvantage of requiring twice as much storage and being up to four times slower than real arithmetic (see Addison [2]). An alternative modification, suggested by Skeel and Kong [12] in the context of blended linear multistep methods, is the approximation of the iteration matrix W by a perfect square matrix $(I-chJ)^2$, where c is real and is chosen to minimize the rate of convergence of the iteration scheme applied to the test equation

$$y'(t)=\lambda y(t), \tag{2.7}$$

$$\operatorname{Re}(\lambda)\leqq 0, \qquad y(t_0)=y_0.$$

A third modification, and the one considered here, is the approximation of W by a perfect square matrix $(aI-bhJ)^2$, where a and b are chosen to yield iteration schemes with the best possible absolute stability properties.

The notation $M(P, C, a, b, m)$ will be used to denote the method resulting from a predictor formula P, and a corrector formula C, of type (2.1) and (2.2) respectively, applied using the perfect square iteration scheme,

$$\begin{aligned}(aI-bhJ)^2\boldsymbol{\delta}^{(r)} &= -\mathbf{y}^{(r-1)}+\beta_0 h\mathbf{f}(\mathbf{y}^{(r-1)})+\gamma_0 h^2\frac{\partial\mathbf{f}}{\partial\mathbf{y}}(\mathbf{y}^{(r-1)})\mathbf{f}(\mathbf{y}^{(r-1)})+\mathbf{g}_{i-1},\\ \mathbf{y}^{(0)} &= \mathbf{y}_i^{(0)}, \qquad r=1,2,\cdots,m,\end{aligned} \tag{2.8}$$

in place of (2.5), with $\mathbf{y}_i=\mathbf{y}^{(m)}$. In the following analysis, m is assumed fixed. This simplifies the analysis and still provides useful insight since in practice the same number of iterations is taken on several successive steps. Three cases will be considered:

Case 1. in which $\mathbf{y}_i', \mathbf{y}_i''$ are chosen using the differential equation, that is

$$\mathbf{y}_i'=\mathbf{f}(\mathbf{y}_i), \qquad \mathbf{y}_i''=\frac{\partial\mathbf{f}}{\partial\mathbf{y}}(\mathbf{y}_i)\mathbf{f}(\mathbf{y}_i);$$

Case 2. in which $\mathbf{y}_i''$ is chosen to satisfy the corrector formula (2.2) with

$$\mathbf{y}_i' = \mathbf{f}(\mathbf{y}_i);$$

Case 3. in which $\mathbf{y}_i'$ is chosen to satisfy the corrector formula (2.2) with

$$\mathbf{y}_i'' = \frac{\partial \mathbf{f}}{\partial \mathbf{y}}(\mathbf{y}_i)\mathbf{y}_i'.$$

$M_I(P, C, a, b, m)$, $I = 1, 2, 3$ will be used to distinguish the cases. It should be noted that in Cases 2 and 3, $\mathbf{y}_i, \mathbf{y}_i', \mathbf{y}_i''$ satisfy the corrector formula. This is not true in general for Case 1. Also, Case 3 does not lead to an efficient implementation because of the need to "invert" $\beta_0 hI + \gamma_0 h^2 (\partial \mathbf{f}/\partial \mathbf{y})(\mathbf{y}^{(m)})$ at every step.

3. Absolute stability properties of $M_I(P, C, a, b, m)$. When the linear test equation (2.7) is considered, the iteration scheme (2.8) can be written as

$$(a - bh\lambda)^2 y_i^{(r)} = \nu(h\lambda) y_i^{(r-1)} + g_{i-1}, \qquad r = 1, 2, \cdots, m \tag{3.1}$$

where

$$\nu(h\lambda) = (a - bh\lambda)^2 - (1 - \beta_0 h\lambda - \gamma_0 h^2\lambda^2) = (a^2 - 1) + (\beta_0 - 2ab)h\lambda + (\gamma_0 + b^2)h^2\lambda^2,$$

is the difference between the perfect square iteration matrix and W. If $ab > 0$, then $(a - bh\lambda) \neq 0$ and the rate of convergence of the iteration is determined by $\theta(a, b) = (a - bh\lambda)^{-2}\nu(h\lambda)$.

For the three above cases, the resulting iteration schemes are as follows:

Case 1.

$$y_i^{(0)} = \sum_{j=1}^{k} (\hat{\alpha}_j + h\lambda\hat{\beta}_j + h^2\lambda^2\hat{\gamma}_j) y_{i-j}, \tag{3.2a}$$

$$(a - bh\lambda)^2 y_i^{(r)} = \nu(h\lambda) y_i^{(r-1)} + \sum_{j=1}^{k} (\alpha_j + h\lambda\beta_j + h^2\lambda^2\gamma_j) y_{i-j}, \qquad r = 1, 2, \cdots, m. \tag{3.2b}$$

Case 2. Here

$$y_i' = \lambda y_i, \tag{3.3a}$$

and in order that the corrector formula (2.2) be satisfied,

$$h^2\gamma_0 y_i'' = y_i - h\beta_0 y_i' - g_{i-1}. \tag{3.3b}$$

Since g_{i-1} can be written as $(a - bh\lambda)^2 y_i^{(m)} - \nu(h\lambda) y_i^{(m-1)}$, (3.3b) can be rewritten as

$$y_i'' = \lambda^2 y_i^{(m)} - \frac{\nu(h\lambda)}{h^2\gamma_0}(y_i^{(m)} - y_i^{(m-1)}), \tag{3.3c}$$

and hence the following iteration scheme is obtained:

$$y_i^{(0)} = \sum_{j=1}^{k} (\hat{\alpha}_j + h\lambda\hat{\beta}_j + h^2\lambda^2\hat{\gamma}_j) y_{i-1} - \nu(h\lambda) \sum_{j=1}^{k} \frac{\hat{\gamma}_j}{\gamma_0}(y_{i-j}^{(m)} - y_{i-j}^{(m-1)}), \tag{3.4a}$$

$$\begin{aligned}(a - bh\lambda)^2 y_i^{(r)} &= \nu(h\lambda) y_i^{(r-1)} + \sum_{j=1}^{k} (\alpha_j + h\lambda\beta_j + h^2\lambda^2\gamma_j) y_{i-j} \\ &\quad - \nu(h\lambda) \sum_{j=1}^{k} \frac{\gamma_j}{\gamma_0}(y_{i-j}^{(m)} - y_{i-j}^{(m-1)}), \qquad r = 1, 2, \cdots, m.\end{aligned} \tag{3.4b}$$

Case 3. In this case

$$(h\beta_0+h^2\gamma_0\lambda)y_i'=y_i-g_{i-1}, \tag{3.5a}$$

$$y_i''=\lambda y_i' \tag{3.5b}$$

provided $\beta_0+h\lambda\gamma_0\neq 0$. Substituting for g_{i-1} as in Case 2, (3.5) can be rewritten as

$$y_i'=\lambda y_i-(h\beta_0+h^2\gamma_0\lambda)^{-1}\nu(h\lambda)(y_i^{(m)}-y_i^{(m-1)}), \tag{3.6a}$$

$$y_i''=\lambda^2 y_i-(h\beta_0+h^2\gamma_0\lambda)^{-1}\nu(h\lambda)\lambda(y_i^{(m)}-y_i^{(m-1)}), \tag{3.6b}$$

and the iteration scheme is

$$y_i^{(0)}=\sum_{j=1}^{k}(\hat{\alpha}_j+h\lambda\hat{\beta}_j+h^2\lambda^2\hat{\gamma}_j)y_{i-j}-\nu(h\lambda)\sum_{j=1}^{k}\frac{(\hat{\beta}_j+h\lambda\hat{\gamma}_j)}{(\beta_0+h\lambda\gamma_0)}(y_{i-j}^{(m)}-y_{i-j}^{(m-1)}), \tag{3.7a}$$

$$\begin{aligned}(a-bh\lambda)^2y_i^{(r)}&=\nu(h\lambda)y_i^{(r-1)}+\sum_{j=1}^{k}(\alpha_j+h\lambda\beta_j+h^2\lambda^2\gamma_j)y_{i-j}\\&\quad-\nu(h\lambda)\sum_{j=1}^{k}\frac{(\beta_j+h\lambda\gamma_j)}{(\beta_0+h\lambda\gamma_0)}(y_{i-j}^{(m)}-y_{i-j}^{(m-1)}),\qquad r=1,2,\cdots,m.\end{aligned} \tag{3.7b}$$

If the matrix W is used instead of the perfect square approximation, the three cases are identical.

Using the notation

$$\mu_j(h\lambda)=\alpha_j+h\lambda\beta_j+h^2\lambda^2\gamma_j,\qquad \hat{\mu}_j(h\lambda)=\hat{\alpha}_j+h\lambda\hat{\beta}_j+h^2\lambda^2\hat{\gamma}_j,$$

$$\alpha(\omega)=\sum_{j=0}^{k}\alpha_j\omega^{k-j},\qquad \hat{\alpha}(\omega)=\sum_{j=1}^{k}\hat{\alpha}_j\omega^{k-j},$$

$$\beta(\omega)=\sum_{j=0}^{k}\beta_j\omega^{k-j},\qquad \hat{\beta}(\omega)=\sum_{j=1}^{k}\hat{\beta}_j\omega^{k-j},$$

$$\gamma(\omega)=\sum_{j=0}^{k}\gamma_j\omega^{k-j},\qquad \hat{\gamma}(\omega)=\sum_{j=1}^{k}\hat{\gamma}_j\omega^{k-j},$$

with $\alpha_0=-1$, the following stability polynomials are established.

CASE 1: THEOREM 1. *If the second derivative formulae* (2.1) *and* (2.2) *are applied to the linear scalar differential equation* $y'(t)=\lambda y(t)$, $y(t_0)=y_0$, λ *complex, using the perfect square iteration scheme with m iterations on each step and with the derivative values* y_i', y_i'' *chosen to satisfy the differential equation* (*Case* 1), *then the absolute stability properties of* $y_i^{(m)}$ *are determined by the moduli of the zeros of the stability polynomial*

$$\begin{aligned}Q_1(\omega;h\lambda)&=(a-bh\lambda)^{2m}\omega^k-\sum_{r=0}^{m-1}\nu(h\lambda)^{m-1-r}(a-bh\lambda)^{2r}\sum_{j=1}^{k}\mu_j(h\lambda)\omega^{k-j}\\&\quad-\nu(h\lambda)^m\sum_{j=1}^{k}\hat{\mu}_j(h\lambda)\omega^{k-j}.\end{aligned} \tag{3.8}$$

Proof. Let $\mathbf{x}_i=(y_i^{(m)},y_i^{(m-1)},\cdots,y_i^{(1)},y_i^{(0)},y_{i-1},y_{i-2},\cdots,y_{i-k+1})^T$.

Then $\mathbf{x}_i$ contains the information that results from the ith step, together with all information from previous steps that is required to advance from t_i to t_{i+1}. It can be verified that $\mathbf{x}_i$ satisfies the following system of equations:

$$T\mathbf{x}_i=S\mathbf{x}_{i-1},\qquad \mathbf{x}_0 \text{ given}, \tag{3.9}$$

with

$$S=\left[\begin{array}{c|ccc|ccc|c}
\multicolumn{4}{c|}{\overleftarrow{\qquad m+1 \qquad}} & \multicolumn{3}{c|}{\overleftarrow{\qquad k-1 \qquad}} & \\
\mu_1(h\lambda) & 0 & \cdots & 0 & \mu_2(h\lambda) & \cdots & \mu_k(h\lambda) & \uparrow \\
\vdots & \vdots & & \vdots & \vdots & & \vdots & \\
\mu_1(h\lambda) & 0 & \cdots & 0 & \mu_2(h\lambda) & \cdots & \mu_k(h\lambda) & m+1 \\
\hat{\mu}_1(h\lambda) & 0 & \cdots & 0 & \hat{\mu}_2(h\lambda) & \cdots & \hat{\mu}_k(h\lambda) & \downarrow \\
\hline
1 & 0 & \cdots & 0 & 0 & \cdots & 0 & \uparrow \\
0 & 0 & \cdots & 0 & 1 & & 0 & \\
\vdots & \vdots & & \vdots & & \ddots & \vdots & k-1 \\
0 & 0 & \cdots & 0 & 0 & & 1 & \downarrow
\end{array}\right]$$

and

$$T=\left[\begin{array}{cccc|cc|c}
\multicolumn{4}{c|}{\overleftarrow{\qquad m+1 \qquad}} & \multicolumn{2}{c|}{\overleftarrow{\quad k-1 \quad}} & \\
(a-bh\lambda)^2 & -\nu(h\lambda) & 0 & 0 & 0 & 0 & \uparrow \\
 & (a-bh\lambda)^2 & -\nu(h\lambda) & & & & m+1 \\
0 & 0 & 0 & 1 & 0 & 0 & \downarrow \\
\hline
0 & & & 0 & 1 & 0 & \uparrow \\
 & & & & & & k-1 \\
0 & & & 0 & 0 & 1 & \downarrow
\end{array}\right]$$

The first m equations represent the m iterations of the corrector, equation $(m+1)$ represents the predictor step, and the remaining $k-1$ equations are identity equations included to match across steps.

The method is stable if and only if the eigenvalues of the generalized eigenvalue problem,

$$(S-\omega T)\mathbf{z}=\mathbf{0}, \tag{3.10}$$

have modulus less than or equal to one with only simple eigenvalues of modulus one, see Gear [8].

Let

$$\mathbf{z}=(z_1^{(m)}, z_1^{(m-1)}, \cdots, z_1^{(1)}, z_1^{(0)}, z_2, z_3, \cdots, z_k)^T,$$

where $z_i = z_i^{(m)}$, $i>1$, then, from (3.10),

$$(\mu_1(h\lambda)-\omega(a-bh\lambda)^2)z_1^{(m)}+\omega\nu(h\lambda)z_1^{(m-1)}+\sum_{j=2}^{k}\mu_j(h\lambda)z_j=0, \tag{3.11a}$$

$$\mu_1(h\lambda)z_1^{(m)}-\omega(a-bh\lambda)^2z_1^{(m-p)}+\omega\nu(h\lambda)z_1^{(m-p-1)}+\sum_{j=2}^{k}\mu_j(h\lambda)z_j=0, \quad p=1,2,\cdots,m-1, \tag{3.11b}$$

$$\hat{\mu}_1(h\lambda)z_1^{(m)}-\omega z_1^{(0)}+\sum_{j=2}^{k}\hat{\mu}_j(h\lambda)z_j=0, \tag{3.11c}$$

$$z_1^{(m)}-\omega z_2=0, \tag{3.11d}$$

$$z_{j-1} - \omega z_j = 0, \qquad j = 3, 4, \cdots, k. \tag{3.11e}$$

Hence, from (3.11e) and (3.11d),

$$z_j = \omega^{k-j} z_k, \qquad j = 2, 3, \cdots, k, \tag{3.12a}$$

$$z_1^{(m)} = \omega^{k-1} z_k, \tag{3.12b}$$

and from (3.11c) and (3.11b),

$$\omega z_1^{(0)} = \left\{ \sum_{j=1}^{k} \hat{\mu}_j(h\lambda)\omega^{k-j} \right\} z_k, \tag{3.12c}$$

and

$$\omega(a - bh\lambda)^2 z_1^{(m-p)} = \omega\nu(h\lambda) z_1^{(m-p-1)} + \sum_{j=1}^{k} \mu_j(h\lambda)\omega^{k-j} z_k, \tag{3.12d}$$

$$p = 1, 2, \cdots, m-1.$$

From (3.12c) and (3.12d), multiplying by powers of $(a - bh\lambda)^2$ and combining terms,

$$\omega(a - bh\lambda)^{2p} z_1^{(p)} = \left[\nu(h\lambda)^p \sum_{j=1}^{k} \hat{\mu}_j(h\lambda) + \sum_{i=0}^{p-1} \nu(h\lambda)^{p-1-i}(a - bh\lambda)^{2i} \cdot \left\{ \sum_{j=1}^{k} \mu_j(h\lambda)\omega^{k-j} \right\} \right] z_k, \tag{3.12e}$$

$$p = 1, 2, \cdots, m-1$$

and hence (3.11a) multiplied by $(a - bh\lambda)^{2(m-1)}$ becomes

$$\left[\nu(h\lambda)^m \sum_{j=1}^{k} \hat{\mu}_j(h\lambda)\omega^{k-j} + \sum_{i=0}^{m-1} \nu(h\lambda)^{m-1-i}(a - bh\lambda)^{2i} \cdot \left\{ \sum_{j=1}^{k} \mu_j(h\lambda)\omega^{k-j} \right\} - (a - bh\lambda)^{2m}\omega^k \right] z_k = 0. \tag{3.13}$$

From (3.13), either $Q_1(\omega; h\lambda) = 0$, where $Q_1(\omega; h\lambda)$ is given in (3.8), or else $z_k = 0$, which implies $\omega = 0$ since $\mathbf{z} = \mathbf{0}$ is not a valid eigenvector. The matrix T has full rank $m + k$, and since S has rank at most k, eigenproblem (3.10) possesses at least m eigenvalues equal to zero.

CASE 2: THEOREM 2. *The stability polynomial for Case* 2 *is*

$$\begin{aligned} Q_2(\omega; h\lambda) = {} & (a - bh\lambda)^{2m}\omega^k\gamma(\omega) - \sum_{r=0}^{m-1} \nu(h\lambda)^{m-1-r}(a - bh\lambda)^{2r} \\ & \times \omega^k\{(\alpha(\omega) + h\lambda\beta(\omega))\gamma_0 - (\alpha_0 + h\lambda\beta_0)\gamma(\omega)\} \\ & - \nu(h\lambda)^m\{(\hat{\alpha}(\omega) + h\lambda\hat{\beta}(\omega))\gamma(\omega) - (\alpha(\omega) + h\lambda\beta(\omega))\hat{\gamma}(\omega)\}. \end{aligned} \tag{3.14}$$

CASE 3: THEOREM 3. *The stability polynomial for Case* 3 *is*

$$\begin{aligned} Q_3(\omega; h\lambda) = {} & (a - bh\lambda)^{2m}\omega^k(\beta(\omega) + h\lambda\gamma(\omega)) - \sum_{r=0}^{m-1} \nu(h\lambda)^{m-1-r}(a - bh\lambda)^{2r} \\ & \times \omega^k\{\alpha(\omega)(\beta_0 + h\lambda\gamma_0) - \alpha_0(\beta(\omega) + h\lambda\gamma(\omega))\} \\ & - \nu(h\lambda)^m\{\hat{\alpha}(\omega)(\beta(\omega) + h\lambda\gamma(\omega)) - \alpha(\omega)(\hat{\beta}(\omega) + h\lambda\hat{\gamma}(\omega))\}. \end{aligned} \tag{3.15}$$

The proofs for Theorems 2 and 3 are essentially the same as for Theorem 1, with $\mathbf{x}_i = (y_i^{(m)}, y_i^{(m-1)}, \cdots, y_i^{(0)}, y_{i-1}, y_{i-2}, \cdots, y_{i-k+1}, y_{i-1}^{(m-1)}, y_{i-2}^{(m-1)}, \cdots, y_{i-k+1}^{(m-1)})^T$.

The matrices T and S are now of size $m-1+2k$, with T of full rank and S of rank at most $2k$.

If the perfect square iteration schemes converge, that is, if $|\theta(a, b)|<1$, then it can be shown in all three cases that

$$\lim_{m\to\infty}\frac{\mu_0(h\lambda)Q_I(\omega; h\lambda)}{(a-bh\lambda)^{2m}}=(\alpha(\omega)+h\lambda\beta(\omega)+h^2\lambda^2\gamma(\omega))\rho_I(\omega; h\lambda)$$

with

$$\rho_I(\omega; h\lambda)=\begin{cases}1, & I=1,\\ \gamma_0\omega^k, & I=2,\\ (\beta_0+h\lambda\gamma_0)\omega^k, & I=3.\end{cases}$$

Notice that $(\alpha(\omega)+h\lambda\beta(\omega)+h^2\lambda^2\gamma(\omega))$ is just the stability polynomial for the corrector formula (2.2), and so if the iteration schemes converge then the absolute stability properties of the methods are identical, in the limit, to those of the underlying corrector formula.

The stability polynomials given in the above theorems can be used to determine whether or not the methods are stable at the origin and at infinity.

The methods are stable at the origin (zero-stable) if the roots of $Q_I(\omega; 0)$ are less than or equal to one in modulus, with only simple roots of modulus one.

$$Q_I(\omega; 0)=\begin{cases}a^{2m}\omega^k-\sum_{r=0}^{m-1}(a^2-1)^{m-1-r}a^{2r}(\alpha(\omega)-\alpha_0\omega^k) & I=1,\\ \quad -(a^2-1)^m\hat{\alpha}(\omega), & \\ a^{2m}\omega^k\gamma(\omega)-\sum_{r=0}^{m-1}(a^2-1)^{m-1-r}a^{2r}\omega^k(\alpha(\omega)\gamma_0-\alpha_0\gamma(\omega)) & I=2,\\ \quad -(a^2-1)^m(\hat{\alpha}(\omega)\gamma(\omega)-\alpha(\omega)\hat{\gamma}(\omega)), & \\ a^{2m}\omega^k\beta(\omega)-\sum_{r=0}^{m-1}(a^2-1)^{m-1-r}a^{2r}\omega^k(\alpha(\omega)\beta_0-\alpha_0\beta(\omega)) & I=3,\\ \quad -(a^2-1)^m(\hat{\alpha}(\omega)\beta(\omega)-\alpha(\omega)\hat{\beta}(\omega)) & \end{cases}\tag{3.16}$$

and in the case $a=1$

$$Q_I(\omega; 0)=\begin{cases}-\alpha(\omega), & I=1,\\ -\alpha(\omega)\omega^k\gamma_0, & I=2,\\ -\alpha(\omega)\omega^k\beta_0, & I=3,\end{cases}$$

where the methods are zero-stable if and only if the underlying second derivative formula (2.2) is zero-stable.

As $h\lambda\to\infty$, the stability polynomial $Q_I(\omega; h\lambda)$ tends to the term involving the largest power of $h\lambda$, say $R_I(\omega)(h\lambda)^N$, with $N\leqq 2m+2$ for $I=1$ and $N\leqq 2m+1$ for $I=2, 3$. $R_I(\omega)$ is of degree k for $I=1$ and of degree $2k$ for $I=2, 3$. Stability at infinity is determined by the polynomial $R_I(\omega)$; the methods are stable at infinity if and only if the roots of $R_I(\omega)$ are less than one. If the leading coefficient of $R_I(\omega)$ is zero, then $R_I(\omega)$ has a root at infinity and the method is unstable at infinity. Tables 1, 2 and 3 summarize the various cases that arise, giving the polynomials whose roots determine the stability at infinity for three types of predictor:

TABLE 1
Case 1.

$\hat{\gamma}_j \neq 0$ for some j	$\gamma_0 + b^2 \neq 0$				unstable
	$\gamma_0 + b^2 = 0$	$m > 2$			$\gamma(\omega)$
		$m = 2$	$\beta_0 - 2ab \neq 0$		$r_1(\omega)$
			$\beta_0 - 2ab = 0$		$\gamma(\omega)$
		$m = 1$	$\beta_0 - 2ab \neq 0$		unstable
			$\beta_0 - 2ab = 0$	$a^2 - 1 \neq 0$	$r_2(\omega)$
				$a^2 - 1 = 0$	$\gamma(\omega)$
$\hat{\gamma}_j - 0 \quad \forall j$	$\gamma_0 + b^2 \neq 0$				unstable
$\hat{\beta}_j \neq 0$ for some j	$\gamma_0 + b^2 = 0$	$m \geqq 2$			$\gamma(\omega)$
		$m = 1$	$\beta_0 - 2ab \neq 0$		$r_3(\omega)$
			$\beta_0 - 2ab = 0$		$\gamma(\omega)$
$\hat{\gamma}_j = \hat{\beta}_j = 0 \quad \forall j$	$\gamma_0 + b^2 \neq 0$				$r_4(\omega)$
$\hat{\alpha}_j \neq 0$ for some j	$\gamma_0 + b^2 = 0$				$\gamma(\omega)$

where

$$r_1(\omega) = -\gamma_0\gamma(\omega) + (\beta_0 - 2ab)^2\hat{\gamma}(\omega),$$

$$r_2(\omega) = \gamma(\omega) + (a^2 - 1)\hat{\gamma}(\omega),$$

$$r_3(\omega) = \gamma(\omega) + (\beta_0 - 2ab)\hat{\beta}(\omega),$$

$$r_4(\omega) = \gamma(\omega) + \left\{\frac{\gamma_0(\gamma_0 + b^2)^m}{(\gamma_0 + b^2)^m - b^{2m}}\right\}(\hat{\alpha}(\omega) - \omega^k),$$

TABLE 2
Case 2.

$\hat{\gamma}_j \neq 0$ for some j	$\gamma_0 + b^2 \neq 0$	$\hat{\gamma}(\omega)\beta(\omega) - \hat{\beta}(\omega)\gamma(\omega) \neq 0$			unstable
		$\hat{\gamma}(\omega)\beta(\omega) - \hat{\beta}(\omega)\gamma(\omega) = 0$	$\hat{\gamma}(\omega)\alpha(\omega) - \hat{\alpha}(\omega)\gamma(\omega) \neq 0$		$r_5(\omega)$
			$\hat{\gamma}(\omega)\alpha(\omega) - \hat{\alpha}(\omega)\gamma(\omega) = 0$		$\gamma(\omega)$
	$\gamma_0 + b^2 = 0$	$m \geqq 2$			$\gamma(\omega)$
		$m = 1$	$\hat{\gamma}(\omega)\beta(\omega) - \hat{\beta}(\omega)\gamma(\omega) \neq 0$	$\beta_0 - 2ab \neq 0$	$r_6(\omega)$
				$\beta_0 - 2ab = 0$	$\gamma(\omega)$
			$\hat{\gamma}(\omega)\beta(\omega) - \hat{\beta}(\omega)\gamma(\omega) = 0$		$\gamma(\omega)$
$\hat{\gamma}_j = 0 \quad \forall j$	$\gamma_0 + b^2 \neq 0$				unstable
$\hat{\beta}_j \neq 0$ for some j	$\gamma_0 + b^2 = 0$	$m \geqq 2$			$\gamma(\omega)$
		$m = 1$	$\beta_0 - 2ab \neq 0$		$r_7(\omega)\gamma(\omega)$
			$\beta_0 - 2ab = 0$		$\gamma(\omega)$
$\hat{\gamma}_j = \hat{\beta}_j = 0 \quad \forall j$	$\gamma_0 + b^2 \neq 0$				$r_8(\omega)\gamma(\omega)$
$\hat{\alpha}_j \neq 0$ for some j	$\gamma_0 + b^2 = 0$				$\gamma(\omega)$

where

$$r_5(\omega) = \gamma(\omega)\omega^k + \frac{(\gamma_0 + b^2)^m}{b^{2m}}\{\hat{\gamma}(\omega)\alpha(\omega) - \hat{\alpha}(\omega)\gamma(\omega)\},$$

$$r_6(\omega) = \gamma(\omega)\omega^k + \frac{(\beta_0 - 2ab)}{b^2}\{\hat{\gamma}(\omega)\beta(\omega) - \hat{\beta}(\omega)\gamma(\omega)\},$$

$$r_7(\omega) = \gamma_0\omega^k + (\beta_0 - 2ab)\hat{\beta}(\omega),$$

$$r_8(\omega) = \omega^k - \frac{(\gamma_0 + b^2)^m}{b^{2m}}\hat{\alpha}(\omega).$$

TABLE 3
Case 3.

$\hat{\gamma}_j \neq 0$ for some j	$\gamma_0 + b^2 \neq 0$	$r_5(\omega)$
	$\gamma_0 + b^2 = 0$	$\gamma(\omega)$
$\hat{\gamma}_j = 0 \quad \forall j$	$\gamma_0 + b^2 \neq 0$	$r_8(\omega)\gamma(\omega)$
$\hat{\beta}_j \neq 0$ for some j	$\gamma_0 + b^2 = 0$	$\gamma(\omega)$
$\hat{\gamma}_j = \hat{\beta}_j = 0 \quad \forall j$	$\gamma_0 + b^2 \neq 0$	$r_8(\omega)\gamma(\omega)$
$\hat{\alpha}_j \neq 0$ for some j	$\gamma_0 + b^2 = 0$	$\gamma(\omega)$

Stability at infinity of the underlying corrector formula (2.2), is determined by the polynomial $\gamma(\omega)$. From Table 3, Case 3, for all three types of predictor, it can be seen that when $\gamma_0 + b^2 = 0$ the methods are stable at infinity if the underlying corrector formula is stable at infinity. When $\gamma_0 + b^2 \neq 0$ stability at infinity depends on $r_5(\omega)$ or $r_8(\omega)\gamma(\omega)$, and as m increases both of these polynomials tend to $\omega^k\gamma(\omega)$ if $\gamma_0 < 0$ and $-\gamma_0 < 2b^2$. The condition $\gamma_0 < 0$ is satisfied if the corrector formula is to yield a unique solution for stiff O.D.E.'s, and the condition $-\gamma_0 < 2b^2$ is satisfied if $|\theta(a, b)| < 1$ as $h\lambda \to \infty$, so $r_5(\omega)$ and $r_8(\omega)\gamma(\omega)$ converge to $\omega^k\gamma(\omega)$ in all practical cases.

However, for Cases 1 and 2 stability at infinity is very dependent on the type of predictor. With Case 1, Table 1, when $\gamma_0 + b^2 \neq 0$, the methods with predictors involving past first or second derivative values are unstable at infinity for any finite m. Therefore, the stability regions are bounded and these methods are not A-stable for any finite number of iterations, even if the underlying corrector formula is A-stable. However, if the predictor involves only past approximations to the solution, stability at infinity depends on $r_4(\omega)$, and as m increases $r_4(\omega)$ converges to $\gamma(\omega)$ when $\gamma_0 < 0$, $-\gamma_0 < 2b^2$. When $\gamma_0 + b^2 = 0$, stability at infinity of Case 1 methods with predictors involving first or second derivative values is determined by $\gamma(\omega)$ for $m \geqq 2$ or $m \geqq 3$ respectively. For smaller m, stability at infinity is determined by $\gamma(\omega)$ only for certain choices of a, and so again the method may be unstable at infinity even if the corrector formula is stable at infinity. When only past approximations to the solution are used in the predictor, and still assuming $\gamma_0 + b^2 = 0$, stability at infinity is determined by $\gamma(\omega)$ for any values of m and a.

The situation is very similar for Case 2, Table 2. If the predictor involves derivatives and $\gamma_0 + b^2 \neq 0$, the methods will be unstable at infinity, except in the special case when $\hat{\gamma}(\omega)\beta(\omega) - \hat{\beta}(\omega)\gamma(\omega) = 0$. When $\gamma_0 + b^2 = 0$, stability at infinity is determined by $\gamma(\omega)$ for $m \geqq 2$ and for some special cases with $m = 1$. If the predictor involves just past approximations to the solution, stability at infinity is determined by $r_8(\omega)\gamma(\omega)$ when $\gamma_0 + b^2 \neq 0$, and by $\gamma(\omega)$ when $\gamma_0 + b^2 = 0$. As m increases $r_8(\omega)\gamma(\omega)$ converges to $\omega^k\gamma(\omega)$.

4. Convergence of the iteration scheme. It is important that the values of a and b chosen on the grounds of good stability should not give rise to poor rates of convergence. To assess the rate of convergence of a perfect square iteration scheme $M_I(P, C, a, b, m)$, we consider the maximum rate for the linear test equation (2.7).

For a given $h\lambda$ with $\mathrm{Re}(h\lambda) \leqq 0$, the iteration scheme (2.8) is convergent if $|\theta(a, b)| < 1$. Furthermore, by the maximum modulus theorem, since $\theta(a, b)$ is analytic

for $\mathrm{Re}(h\lambda) \leqq 0$, the $\max_{\mathrm{Re}(h\lambda)\leqq 0} |\theta(a, b)|$, denoted $\phi(a, b)$, is attained for $\mathrm{Re}(h\lambda) = 0$, and

$$\phi(a, b) = \max_{-\infty<\xi<\infty} \left| \frac{(a^2-1)+(\beta_0-2ab)i\xi-(\gamma_0+b^2)\xi^2}{(a-bi\xi)^2} \right|,$$

or equivalently,

$$\phi(a, b)^2 = \max_{-\infty<\xi<\infty} \frac{((a^2-1)-(\gamma_0+b^2)\xi^2)^2+(\beta_0-2ab)^2\xi^2}{(a^2+b^2\xi^2)^2}. \tag{4.1}$$

Let

$$\Phi^2(a, b, \xi) = \frac{((a^2-1)-(\gamma_0+b^2)\xi^2)^2+(\beta_0-2ab)^2\xi^2}{(a^2+b^2\xi^2)^2}.$$

Differentiation with respect to ξ yields extrema at

$$\xi = 0, \quad \xi^2 = -\frac{b^2}{a^2}, \quad \xi^2 = \infty, \quad \xi^2 = \frac{a^2((\beta_0-2ab)^2-2(a^2-1)(\gamma_0+b^2))-2b^2(a^2-1)}{b^2((\beta_0-2ab)^2-2(a^2-1)(\gamma_0+b^2))-2(\gamma_0+b^2)^2},$$

the last denoted $\hat{\xi}^2$. Only real ξ are of interest and so the extrema $\xi^2 = -b^2/a^2$ and $\hat{\xi}^2 < 0$ are not relevant. Thus,

$$\phi(a, b)^2 = \begin{cases} \max\left\{\dfrac{(a^2-1)^2}{a^4}, \dfrac{(\gamma_0+b^2)^2}{b^4}\right\}, & \text{if } \hat{\xi}^2 < 0, \\ \max\left\{\dfrac{(a^2-1)^2}{a^4}, \dfrac{(\gamma_0+b^2)^2}{b^4}, \Phi^2(a, b, \hat{\xi})\right\}, & \text{if } \hat{\xi}^2 \geqq 0. \end{cases} \tag{4.2}$$

Therefore $\phi(a, b)$ provides some indication of the convergence properties associated with using a and b. As a consequence the values of a and b should be such that $\phi(a, b)$ is small, so that good rates of convergence are achieved at least for linear problems.

5. Numerical results. To illustrate some of the results of §§ 3 and 4, the following second derivative formulae are considered:

$$C^{(3)}: y_i = y_{i-1} + \frac{h}{3}(2y'_i + y'_{i-1}) - \frac{h^2}{6} y''_i, \tag{5.1a}$$

$$C^{(4)}: y_i = y_{i-1} + \frac{h}{48}(29y'_i + 20y'_{i-1} - y'_{i-2}) - \frac{h^2}{8} y''_i. \tag{5.1b}$$

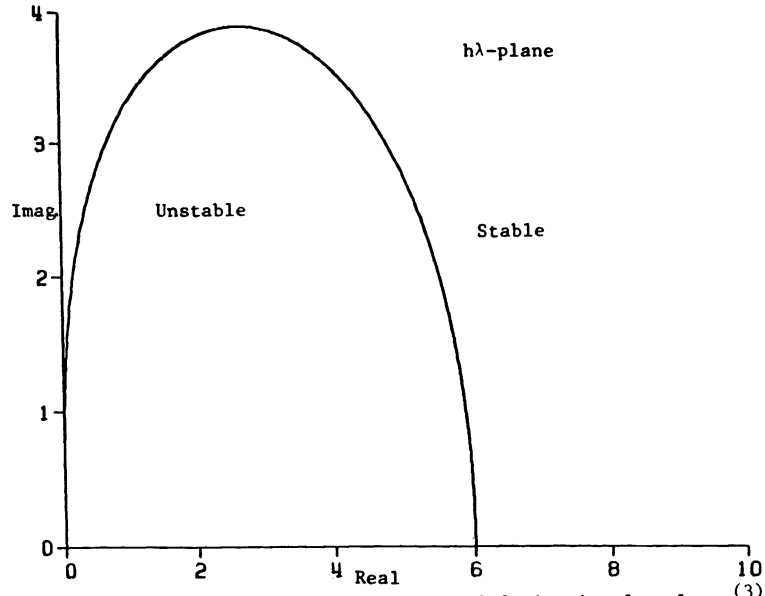
$C^{(3)}$ and $C^{(4)}$ are respectively the third and fourth order formulae derived by Enright [6]. Both formulae are A-stable and their regions of absolute stability are shown in Fig. 1. (As stability regions are symmetric about the real axis, only the half-regions with $\mathrm{Im}(h\lambda) \geqq 0$ are shown in the figures.)

Three types of predictors are considered:

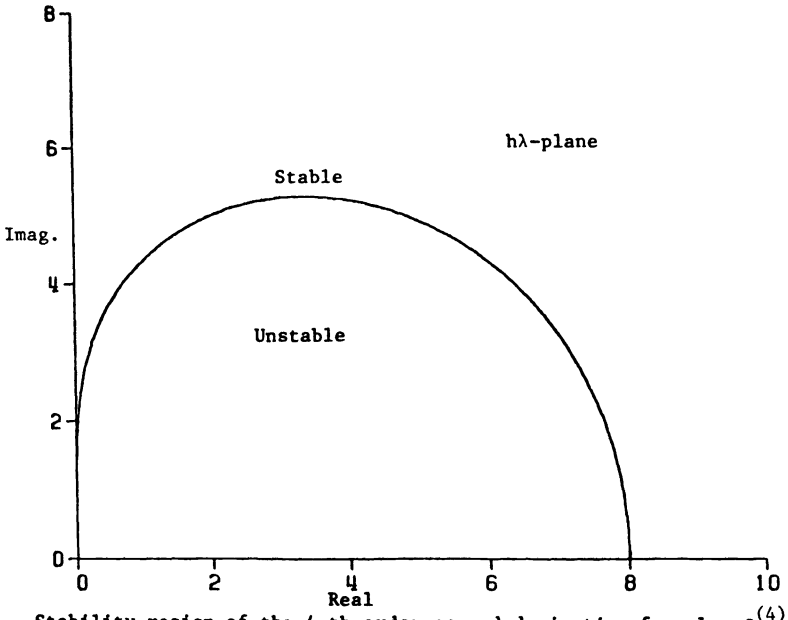
1) the third and fourth order explicit second derivative formulae of the Enright type:

$$P_1^{(3)}: y_i^{(0)} = y_{i-1} + \frac{h}{3}(2y'_{i-1} + y'_{i-2}) + \frac{5}{6} h^2 y''_{i-1}, \tag{5.2a}$$

$$P_1^{(4)}: y_i^{(0)} = y_{i-1} + \frac{h}{48}(11y'_{i-1} + 44y'_{i-2} - 7y'_{i-3}) + \frac{9}{8} h^2 y''_{i-1}; \tag{5.2b}$$



Stability region of the 3-rd order second derivative formula, $C^{(3)}$.



Stability region of the 4-th order second derivative formula, $C^{(4)}$.

FIG. 1

2) the third and fourth order Adams-Bashforth formulae:

$$P_2^{(3)}: y_i^{(0)} = y_{i-1} + \frac{h}{12}(23y'_{i-1} - 16y'_{i-2} + 5y'_{i-3}), \tag{5.3a}$$

$$P_2^{(4)}: y_i^{(0)} = y_{i-1} + \frac{h}{24}(55y'_{i-1} - 59y'_{i-2} + 37y'_{i-3} - 9y'_{i-4}); \tag{5.3b}$$

3) the third and fourth order explicit formulae based only on past y_i values:

$$P_3^{(3)}: y_i^{(0)} = 4y_{i-1} - 6y_{i-2} + 4y_{i-3} - y_{i-4}, \tag{5.4a}$$

$$P_3^{(4)}: y_i^{(0)} = 5y_{i-1} - 10y_{i-2} + 10y_{i-3} - 5y_{i-4} + y_{i-5}. \tag{5.4b}$$

It should be noted that the predictor has to have the same order as the corrector if $a \neq 1$, so as to maintain the order of the method.

Since Enright's formulae (5.1) only involve the second derivative y''_i, the stability regions for Cases 1 and 2 are identical whenever the predictor does not involve second derivatives, that is for the P_2 and P_3 predictors (5.3) and (5.4).

For each method $M_I(C, P, a, b, 1)$ the values of a and b which give rise to the "best" region of stability were obtained. In the case of unbounded regions this was done by minimizing the horizontal distance d, from the imaginary axis to the boundary of the stability region in the left half plane, and maximizing the angle α for which the method is $A(\alpha)$-stable, subject to the constraint of zero stability. The optimization was performed using the code NLPQL (see Hock [9]) which implements a nonlinear programming method with quadratic or linear least squares subproblems. For methods with bounded regions, that is Cases 1 and 2 with predictors P_1 and P_2, the "largest" region was obtained by maximizing d subject to the constraint of zero stability, and maintaining a connected region.

The values of a and b are given in Table 4 together with the values of d and $\pi/2 - \alpha$ where applicable, and the values of $\phi(a, b)$. The results for $m = 2$ are also given. Figures 2-6 illustrate some of the regions of absolute stability.

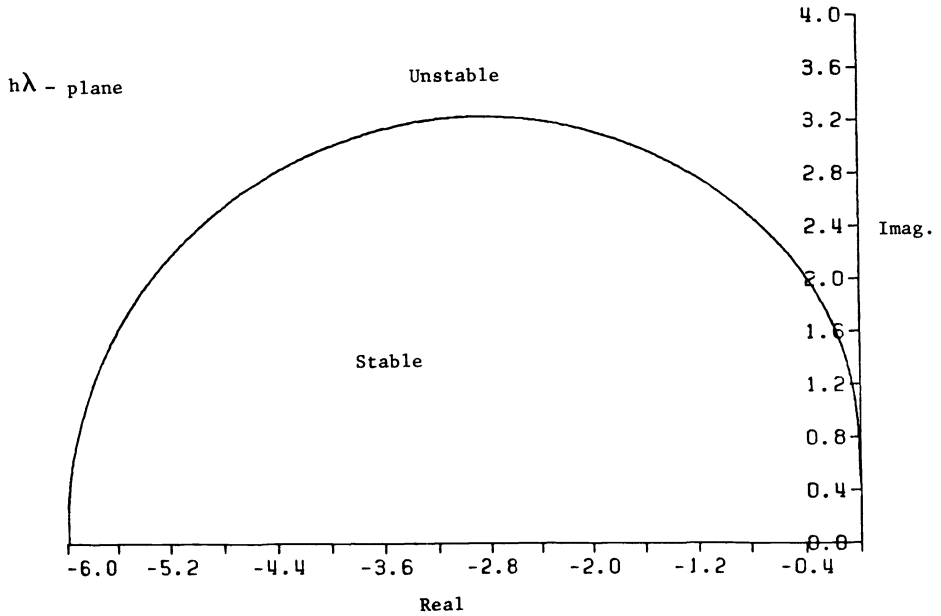


FIG. 2. *Stability region of* $M_1(P_1^{(3)}, C^{(3)}, \beta_0/2b, (-\gamma_0)^{1/2}, 1)$.

TABLE 4

Corrector	Case	Predictor	a	b	m	d	$\pi/2-\alpha$	$\phi(a,b)$	Fig.
$C^{(3)}$	1	$P_1^{(3)}$	$\beta_0/2b$	$(-\gamma_0)^{1/2}$	1	5.9753	Bounded	0.5	2
					2	A-stable			
$C^{(3)}$	2	$P_1^{(3)}$	0.9129	$(-\gamma_0)^{1/2}$	1	A-stable		0.2000	
					2	A-stable			
$C^{(3)}$	1&2	$P_2^{(3)}$	0.7253	$(-\gamma_0)^{1/2}$	1	0.1748	0.1210	0.9007	3
					2	A-stable			
$C^{(3)}$	1&2	$P_3^{(3)}$	0.9354	0.3968	1	0.2705	0.0971	0.1429	
					2	A-stable			
$C^{(3)}$	3	$P_1^{(3)}$	0.8509	0.4064	1	0.0730	0.0400	0.3812	
					2	A-stable			
$C^{(3)}$	3	$P_2^{(3)}$	0.9130	0.3812	1	0.0225	0.0211	0.1996	
					2	A-stable			
$C^{(3)}$	3	$P_3^{(3)}$	0.9253	0.3953	1	0.0778	0.0569	0.1681	4
					2	A-stable			
$C^{(4)}$	1	$P_1^{(4)}$	$\beta_0/2b$	$(-\gamma_0)^{1/2}$	1	2.6953	Bounded	0.3698	
					2	A-stable			
$C^{(4)}$	2	$P_1^{(4)}$	$(9/10)^{1/2}$	$(-\gamma_0)^{1/2}$	1	7.9540	Bounded	0.1111	
					2	0.0008	0.0005		
$C^{(4)}$	1&2	$P_2^{(4)}$	0.8600	$(-\gamma_0)^{1/2}$	1	1.7530	0.4594	0.3521	5
					2	A-stable			
$C^{(4)}$	1&2	$P_3^{(4)}$	0.9682	0.3502	1	3.6697	0.6337	0.0860	
					2	0.0279	0.0151		
$C^{(4)}$	3	$P_1^{(4)}$	0.9015	0.3434	1	0.3143	0.1366	0.2305	6
					2	A-stable			
$C^{(4)}$	3	$P_2^{(4)}$	0.9541	0.3448	1	0.8129	0.2207	0.0984	
					2	0.0005	0.0004		
$C^{(4)}$	3	$P_3^{(4)}$	0.9271	0.3482	1	0.4546	0.2008	0.1635	
					2	A-stable			

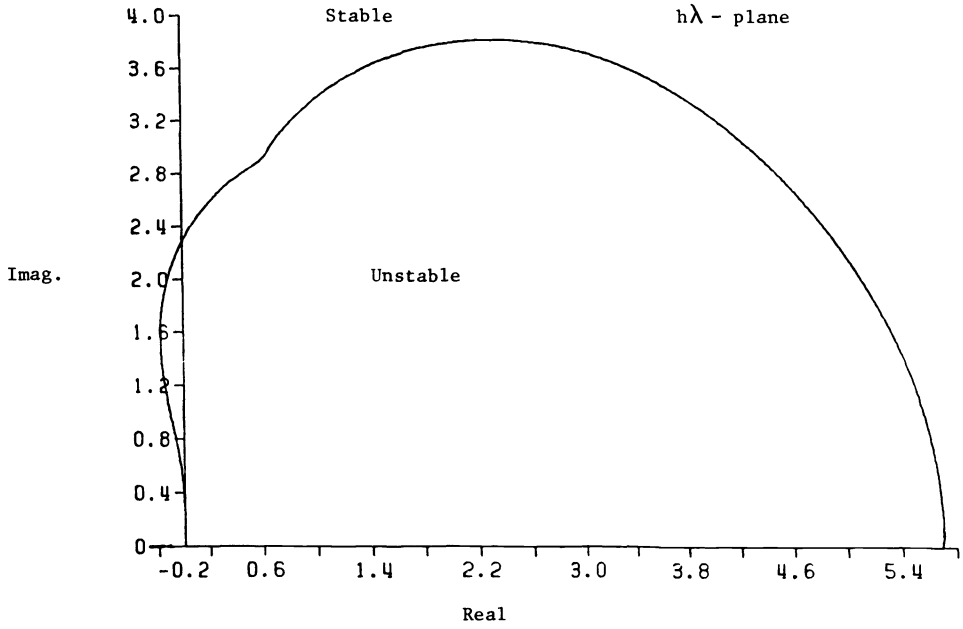


FIG. 3. *Stability region of* $M_{1\&2}(P_2^{(3)}, C^{(3)}, 0.7253, (-\gamma_0)^{1/2}, 1)$.

6. Conclusions. The purpose of this paper is to demonstrate the effect that certain implementation strategies can have on the stability properties of second derivative formulae. These and related formulae are of interest mainly because of their good stability and accuracy properties. A major disadvantage of these, as mentioned at the beginning of the paper, is the cost associated with using them. It has been shown analytically that one suggestion for an improvement in efficiency, namely the use of

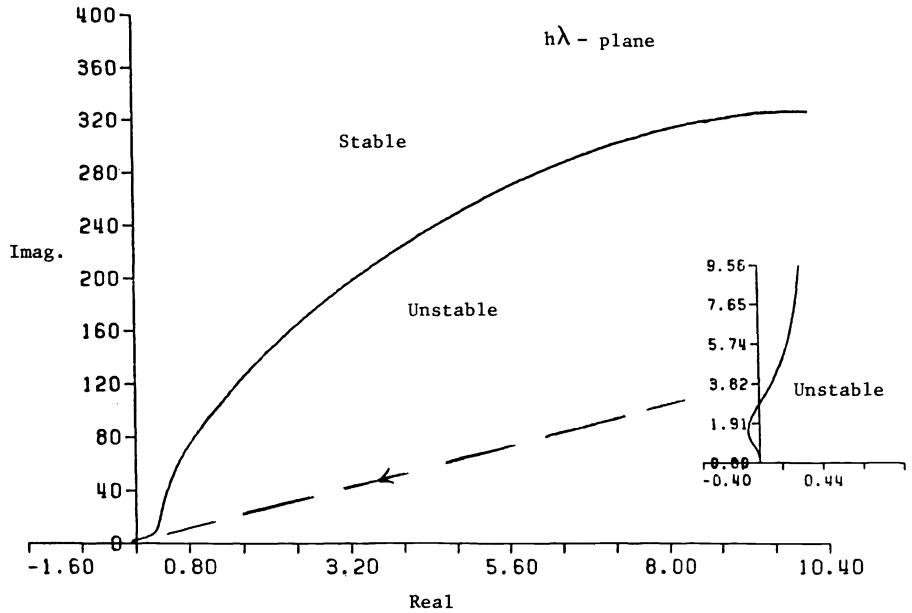


FIG. 4. *Stability region of* $M_3(P_3^{(3)}, C^{(3)}, 0.9253, 0.3953, 1)$.

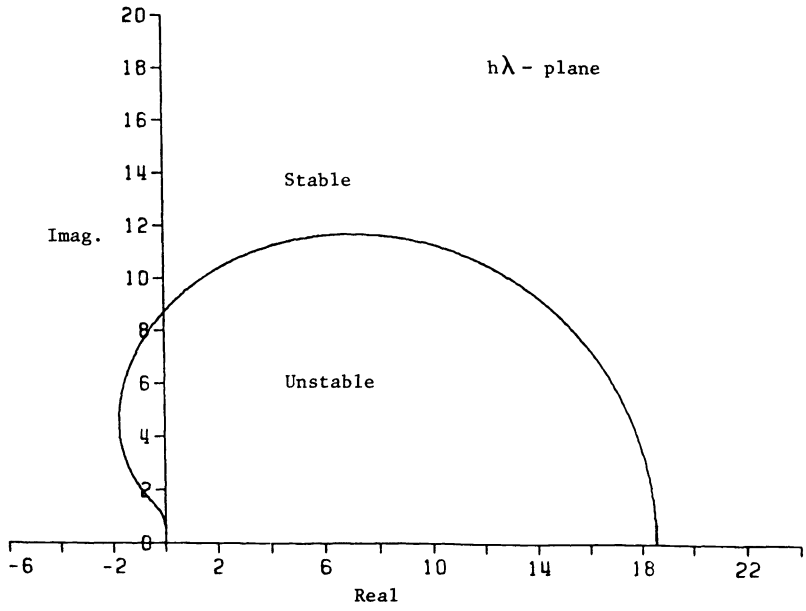


FIG. 5. *Stability region of* $M_{1\&2}(P_2^{(4)}, C^{(4)}, 0.8600, (-\gamma_0)^{1/2}, 1)$.

perfect square iteration matrices to approximate the regular iteration matrices, can have devastating effects on the stability properties of the resulting method. If the a and b used in the perfect square iteration matrix are not chosen carefully, the resulting method may possess a bounded stability region irrespective of the number of iterations performed. The test examples illustrate the variations that can arise even when a and b are chosen carefully—chosen in an attempt to obtain the best stability region possible

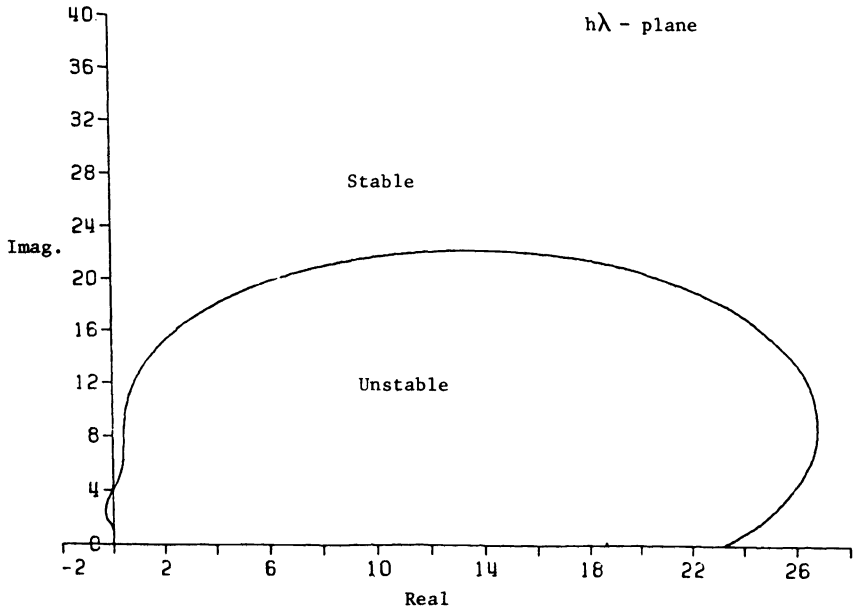


FIG. 6. *Stability region of* $M_3(P_1^{(4)}, C^{(4)}, 0.9015, 0.3434, 1)$.

after one iteration. This is because the choice of predictor and the way in which the past derivative and second derivative values are obtained (either from the differential equation or somehow from the formula) also affect the stability. Usually, but not always, using the formula to obtain an estimate of the derivatives leads to a method with stability properties equivalent or superior to those of a method where the differential equation is used. The comparison of the results from Cases 1 and 2 provides some weak support for this, but the main support comes in comparing the results obtained from Case 3 with those from Case 1. In Case 1 with predictor types 1 and 2 (which involve past derivative values) reasonable stability properties can only be obtained with $b = (-\gamma_0)^{1/2}$. Any perturbation of b from this value results in a method with a bounded stability region. The results for Case 3, on the other hand, vary continuously with the values of a and b in a neighbourhood about the reported values, with the stability regions for Case 3 generally as good as or superior to those of Case 1.

Unfortunately, even in Case 3, A-stability is usually not attained until at least two (and in some instances three) iterations are performed. This is important because it suggests that instability could become a major problem (in the best of situations) if several successive steps of the integration are taken with only one iteration per step. Two iterations of the perfect square approach require $4n^2$ real operations plus 4 function evaluations, whereas one iteration of the complex arithmetic approach requires the equivalent of at most $4n^2$ real operations plus 2 function evaluations (see Addison [2]). Consequently, if a minimum of two iterations is required with the perfect square approach for stability reasons, the cost of performing the iterations, independent of matrix factorizations, may be more expensive than using complex arithmetic. On the other hand, the perfect square approach has more modest storage requirements and so may be preferable for this reason.

When dealing with true second derivative formulae, the situation is even more pessimistic than it might at first appear, since Case 3 does not lead to an efficient implementation with these formulae. Case 3 is really only relevant when considering blended formulae as implemented in Skeel and Kong [12], or the hybrid formulae of England [5], in that with both of these formula types it is possible to obtain efficient implementations akin to Case 3. Further details of the stability properties of blended formulae can be found in Addison and Hanson [3].

The intention in this paper is not to suggest that perfect square iteration matrices are inadequate approximations to the general quadratic iteration matrix, but rather that extreme caution be taken when using them. The theoretical results required to compute the stability regions of any particular implementation of second derivative formulae have been presented, and aspects of the implementation which might be worth altering to improve the stability region, if found inadequate, have been suggested. Moreover, in instances, such as with the blended formulae, in which an implementation akin to Case 3 is practical, it is recommended that it be adopted. Finally, if the second derivative code is to be used in applications in which A-stability is critical, then it is recommended that the use of complex arithmetic to solve the system of equations on each step be thoroughly examined before opting for a perfect square iteration matrix.

Acknowledgments. The authors would like to thank Drs. R. D. Skeel and I. Gladwell for their very helpful comments and suggestions.

REFERENCES

[1] C. A. ADDISON, *Implementing a stiff method based upon the second derivative formulas*, Dept. of Computer Sci. Tech. Rep. No. 130, University of Toronto, 1980.

[2] C. A. ADDISON, *A comparison of several formulas on lightly damped, oscillatory problems*, this Journal, 5 (1984), pp. 920–936.

[3] C. A. ADDISON AND P. M. HANSON, *An investigation into the stability properties of blended formulae*, Dept. of Comput. Sci. Tech. Rep. No. 85-13, University of Alberta, 1985.

[4] G. DAHLQUIST, *A special stability property for linear multistep methods*, BIT, 3 (1963), pp. 27–43.

[5] R. ENGLAND, *Some hybrid implicit stiffly stable methods for ordinary differential equations*, Proc. Third IIMAS Workshop held at Cocoyoc, Mexico, J. P. Hennart, ed., Lecture Notes in Mathematics 909, Springer-Verlag, New York, 1981, pp. 147–158.

[6] W. H. ENRIGHT, *Optimal second derivative methods for stiff systems*, in Stiff Differential Systems, R. A. Willoughby, ed., Plenum Press, New York, 1974, pp. 95–111.

[7] W. H. ENRIGHT AND J. D. PRYCE, *Two* FORTRAN *packages for assessing initial value methods*, Dept. of Comput. Sci. Tech. Rep. No. 167/83, Univ. Toronto, 1983.

[8] C. W. GEAR, *Numerical Initial Value Problems in Ordinary Differential Equations*, Prentice-Hall, Englewood Cliffs, New Jersey, 1971.

[9] W. HOCK, *Test examples for non-linear programming codes*, Lecture Notes in Economics and Mathematical Systems 187, Springer-Verlag, New York, 1981.

[10] W. LINIGER AND R. WILLOUGHBY, *Efficient numerical integration of stiff systems of ordinary differential equations*, Tech. Report RC-1970, IBM T. J. Watson Research Center, Yorktown Heights, New York, 1970.

[11] R. SACKS-DAVIS, *Fixed leading coefficient implementation of* S.D.*-formulas for stiff* O.D.E.'s, ACM Trans. Math. Software, 6 (1980), pp. 540–562.

[12] R. D. SKEEL AND A. K. KONG, *Blended linear multistep methods*, ACM Trans. Math. Software, 3 (1977), pp. 326–345.

SIAM J. SCI. STAT. COMPUT.
Vol. 7, No. 4, October 1986

013

CONTINUATION AND LOCAL PERTURBATION FOR MULTIPLE BIFURCATIONS*

E. L. ALLGOWER† AND C.-S. CHIEN‡

Abstract. We describe numerical methods for the detection of multiple bifurcations on solution paths of certain gradient maps, and for effecting the branching off via appropriate local perturbations. Our model problems are quasi-linear elliptic boundary value problems, and their discretizations via finite differences and finite elements. Sample numerical experiments are reported.

Key words. continuation methods, multiple bifurcation, gradient maps, quasi-linear elliptic boundary value problems

AMS(MOS) subject classifications. 65H10, 65N99, 58C15

1. Introduction. In this paper we describe methods for the numerical approximation of bifurcation curves which satisfy equations of the form

$$F(u, \lambda) = 0, \tag{1.1}$$

where $F: B_1 \times \mathbb{R} \to B_2$, $u \in B_1$, $\lambda \in \mathbb{R}$ is a smooth mapping with $0 \in B_2$ a regular value, B_1 and B_2 are two real Banach spaces, and $\mathbb{R}$ is the real line.

If the assumption that 0 is a regular value is not satisfied, that is, the Frechet derivative DF is not surjective, then $F^{-1}(0)$ may contain bifurcation points.

Equation (1.1) arises in many different areas, for example, homotopy methods and nonlinear eigenvalue problems (see [2], [6]). We will only discuss the latter in this paper. In this case, B_1 and B_2 are infinite-dimensional spaces of functions with certain smoothness conditions.

To solve a nonlinear eigenvalue problem numerically by the continuation method, it is necessary to first discretize the differential equation, for example, by a finite difference method or a finite element method. In both cases, (1.1) is approximated by a finite-dimensional problem

$$H(x, \lambda) = 0 \tag{1.2}$$

where $H: \mathbb{R}^N \times \mathbb{R} \to \mathbb{R}^N$, $x \in \mathbb{R}^N$, $\lambda \in \mathbb{R}$ is a smooth mapping. Viewing λ as the continuation parameter, the continuation algorithm can be implemented to follow the solution curves of (1.2). Of special interest here are singular points such as turning points and bifurcation points. Various techniques for the numerical treatment of bifurcations have been developed and numerically tested (see, for example, [11]-[13], [15]-[17], [24]). Among the methods for effecting a change of branches, perturbation is perhaps the simplest one, since it requires no extra computation when incorporated with Newton iterations and it may still succeed where branches are tangent. Most numerical experiments in the literature have been restricted to simple bifurcation where one secondary curve branches off the primary curve. To the writers' knowledge, few experiments have been published concerning multiple bifurcation points.

In [24] Rheinboldt proposed sophisticated methods to handle multiple bifurcations, but the method does not seem to have been implemented or tested. Kearfott

* Received by the editors March 6, 1985, and in revised form August 15, 1985.

† Department of Mathematics, Colorado State University, Fort Collins, Colorado 80523.

‡ Department of Applied Mathematics, National Chung-Hsing University, Taichung, Taiwan, Republic of China.

[15] gave a numerical report of double bifurcation by determining tangent directions there. In the latter method, large amounts of computation are necessary, particularly when the dimension becomes large.

Our main point here is that local perturbation techniques can still be used to handle multiple bifurcations in the above cases. The theoretical foundation is based on a version of a generalized Sard's theorem. The choice of perturbation vectors plays a key role in this method.

This paper is organized as follows. In § 2 we briefly outline a predictor-corrector type continuation method. A criterion for testing for simple and multiple bifurcations is also given there. The application of perturbation to continuation methods is given in § 3. In § 4 we give details concerning numerical implementation. In § 5 sample numerical experiments are reported on nonlinear eigenvalue problems of the form

$$\begin{aligned} &\Delta u + \lambda f(u) = 0 \qquad \text{in } \Omega, \\ &u = 0 \qquad \text{on } \partial\Omega, \end{aligned} \tag{1.3}$$

where Ω is a compact domain in $\mathbb{R}^2$ with piecewise smooth boundary and $f: \mathbb{R}^1 \to \mathbb{R}^1$ is a smooth, odd map. Both finite difference methods and finite element methods have been used. Perturbation techniques have been successfully used to handle simple and multiple bifurcation in our numerical experiments.

2. The numerical continuation method.

2.1. Basic theory. Let H be defined as in (1.2) with zero as a regular value in $\mathbb{R}^N$. It follows from the implicit function theorem that $H^{-1}(0)$ is a 1-dimensional manifold which is the disjoint union of smooth curves $c(s)$ which are diffeomorphic to either $\mathbb{R}^1$ or S^1. We denote c by

$$c = \{y(s) = (x(s), \lambda(s)) \mid H(y(s)) = 0,\ s \in I\}. \tag{2.1}$$

Here I is any interval in $\mathbb{R}^1$. We assume a parametrization via arc length is available on c. By differentiating the equation $H(y(s)) = 0$ with respect to s, we obtain

$$DH(y(s)) \cdot \dot{y}(s) = 0, \tag{2.2}$$

where $\dot{y}(s) = (\dot{x}(s), \dot{\lambda}(s))^T$ denotes a tangent vector to c at $y(s)$ and $DH(y(s)) = (D_x H(y(s)), D_\lambda H(y(s)))$ is the $N \times (N+1)$ Jacobian matrix of rank N. It follows from (2.2) that the augmented Jacobian matrix

$$A(y(s)) = \begin{pmatrix} DH(y(s)) \\ \dot{y}(s)^T \end{pmatrix} \tag{2.3}$$

is nonsingular for all $s \in I$. If an orientation is given, and a starting point $y(0) = (x(0), \lambda(0))$ is known, then one can numerically trace c by solving the Davidenko initial value problem

$$\begin{aligned} &DH(y(s)) \cdot \dot{y}(s) = 0, \\ &\|\dot{y}(s)\| = 1, \\ &y(0) = (x(0), \lambda(0)). \end{aligned} \tag{D}$$

2.2. Predictor-corrector type continuation method. Let $y_i = (x_i, \lambda_i) \in \mathbb{R}^{N+1}$ be a point which has been accepted as an approximating point for c. A new point $z_{i+1,1}$ is predicted either by interpolation [29] or an Adams-Bashforth predictor [6], [12]. The Euler predictor [2], [4]

$$z_{i+1,1} = y_i + \delta_i \cdot u_i \tag{2.4}$$

is the simplest one which has been used. Here $\delta_i > 0$ is the step length, and u_i is the unit tangent vector at y_i, which is obtained either by solving the linear system (see [2], [4], [26])

$$A(y_i) \cdot u_i = \begin{pmatrix} \bar{0} \\ 1 \end{pmatrix} \tag{2.5}$$

or

$$\begin{pmatrix} DH(y_i) \\ e_k^T \end{pmatrix} \cdot u_i = \begin{pmatrix} \bar{0} \\ 1 \end{pmatrix}, \tag{2.6}$$

where e_k is the kth standard basis vector of $\mathbb{R}^{N+1}$ such that

$$|e_k^T \cdot u_{i-1}| = \max\{|e_j^T \cdot u_{i-1}|, j = 1, \cdots, N+1\}.$$

To maintain orientation and control the local curvature, we also require

$$u_i \cdot u_{i-1} > 1 - \alpha > 0 \quad \text{for some } \alpha \in (0, 1). \tag{2.7}$$

Either direct methods or iterative methods can be used to solve (2.5) or (2.6).

The accuracy of approximation to the solution curve must generally be improved via a corrector process. This may be done by choosing a hyperplane which is orthogonal to $\dot{y}(s)$ at $z_{i+1,1}$ and performing Newton iterations constrained to the hyperplane. In practice, the modified Newton's method with constraint

$$\begin{pmatrix} DH(z_{i+1,1}) \\ \gamma^T \end{pmatrix} w_j = \begin{pmatrix} -H(z_{i+1,j}) \\ 0 \end{pmatrix}, \qquad j = 1, 2, 3, \cdots, \tag{2.8}$$

is solved. Two possible choices for the constraint vector γ are:

(i) the current unit tangent vector,

(ii) the standard basis e_k of $\mathbb{R}^{N+1}$ defined in (2.6).

One can solve (2.8) by the same methods for solving (2.5) and (2.6).

If y_i lies sufficiently near c, then the Newton process (2.8) will also converge for step size $h_i > 0$ sufficiently small. This criterion, as well as others concerning curvature and distances are the factors which govern the step size selection process. For more details concerning predictor-corrector continuation methods, see, for example, [2], [4], [6].

2.3. Testing for bifurcation points. As was discussed in § 1, if 0 is not a regular value of H (or correspondingly of F), then one may encounter points on c where DH has a positive rank deficiency. Let us distinguish turning points and bifurcation points. The turning points of c are the critical points of $\lambda(s)$, that is $\dot{\lambda} = 0$. However, unless $DH(y(s))$ also has a positive rank deficiency, where $\dot{\lambda} = 0$, we shall not call a turning point a bifurcation point. A necessary condition for a bifurcation to occur at a point $y(s^*) \in c$ is that $DH(y(s^*))$ should have a positive rank deficiency, or what is equivalent, the augmented Jacobian

$$A(y(s^*)) = \begin{pmatrix} DH(y(s^*)) \\ \dot{y}(s^*)^T \end{pmatrix}$$

should be singular. By a familiar theorem of Crandall and Rabinowitz [10], a sufficient condition for a bifurcation to occur at $y(s^*)$ is that the rank deficiency of $DH(y(s^*))$ should be odd, or what is correspondingly equivalent, det $A(y(s))$ changes sign on all sufficiently small neighborhoods of $y(s^*)$. For even rank deficiency, bifurcation does not necessarily occur. M. S. Berger [7] has given a sufficient condition for bifurcation

to occur at points $y(s^*) \in c$ where $DH(y(s^*))$ has an arbitrary rank deficiency: If F is an odd gradient map with respect to u and $DF(u(s^*), \lambda(s^*))$ has rank deficiency m, then at least m branches bifurcate from c at $(u(s^*), \lambda(s^*))$. In the finite-dimensional context, in order for Berger's result to apply, $D_x H$ should be symmetric and $D_x H(-x, \lambda) = -D_x H(x, \lambda)$ should hold.

Thus, the task of numerically detecting possible bifurcations from a solution curve $c \subset H^{-1}(0)$ reduces to monitoring the rank of $DH(y(s))$ or of $A(y(s))$. Perhaps the simplest criterion to monitor for bifurcation is whether

$$\det A(y_i) \det A(y_{i+1}) < 0 \tag{2.9}$$

for two accepted consecutive approximations y_i and y_{i+1}. Criterion (2.9) or some variation thereof has been employed by several authors (for example, [2], [6]-[11], [16]). Since in general a reduction of $DH(y_i)$ is necessary to obtain the unit tangent u_i, the evaluations in (2.9) are performed at very little additional cost. By the Crandall-Rabinowitz result, at least one "odd" bifurcation from c occurs between y_i and y_{i+1}. Assuming that y_i and y_{i+1} are sufficiently near one another so that only one bifurcation from c occurs between them, it is not certain whether this might be a multiple bifurcation of odd multiplicity.

Kearfott [15] has used the monitoring of the condition number of $DH(y_i)$ via singular value decompositions to detect possible bifurcations. At points $y_i \in c$ where the condition number of $DH(y_i)$ warrants, the functional $\|H(y)\|_2^2$ is minimized subject to the constraint $\|y - y_i\| = \delta$ to obtain potential bifurcating directions. The potential bifurcating directions are then used to obtain predictors in the continuation method attempt to locate a bifurcating branch. Although this method appears to be generally successful and is fairly exhaustive, it appears to be too costly to be applied when the dimension N is large. On the other hand, when H represents a discretization of an operator equation, it is generally necessary to select a large N in order to obtain a reasonably good truncation error.

Since we have assumed that $D_x H$ is symmetric, we will exploit this fact to detect the singular points of $A(y(s))$ while a reduction process is done to determine the unit tangent $u(s)$ to the curve c at $y(s)$. To begin our discussion, let us write $DH(y) = (D_x H(y) D_\lambda H(y))$. If we perform a series of row operations upon $DH(y)$ to reduce it to upper triangular form, this series may be represented by a nonsingular matrix $P(y)$ so that

$$P(y)(DH(y)) = (P(y) D_x H(y) P(y) D_\lambda H(y)) = \begin{pmatrix} h_{11} & \cdots & h_{1N} & c_1 \\ 0 & & h_{NN} & c_N \end{pmatrix}. \tag{2.10}$$

To obtain a unit tangent, it is necessary to solve $(DH)w = 0$ or equivalently,

$$(P(D_x H) P(D_\lambda H)) w = 0 \tag{2.11}$$

(where we have now suppressed the y-dependence). From (2.10) and the symmetry of $D_x H$, we have

$$P(D_x H) P^T = \begin{pmatrix} h_{11} & 0 \\ 0 & h_{NN} \end{pmatrix}. \tag{2.12}$$

(If diag $(D_x H) = 0$, we make a preliminary transformation $(I + e_1 e_j^T)(D_x H)(I + e_1 e_j^T)^T$ for any j with $(D_x H)_{1j} \neq 0$.) Hence, if we set

$$w = \begin{pmatrix} P^T & 0 \\ 0^T & 1 \end{pmatrix} v,$$

then from (2.11),

$$(P(D_xH)P^TP(D_\lambda H))v=0. \tag{2.13}$$

From (2.10), (2.12), (2.13) it follows that

$$\begin{pmatrix} h_{11} & & c_1 \\ & \ddots\ 0 & \vdots \\ 0 & h_{NN} & c_N \end{pmatrix}\begin{pmatrix} v_1 \\ \vdots \\ v_{N+1} \end{pmatrix}=0, \tag{2.14}$$

or equivalently,

$$h_{jj}v_j+c_jv_{N+1}=0 \quad \text{for } j=1,\cdots,N. \tag{2.15}$$

There are three basic cases we should distinguish in (2.14) or (2.15):

(i) $h_{jj}\neq 0$ for $j=1,\cdots,N$. In this case $\dim(\mathcal{N}(DH))=1$, that is, y is a regular point of H. The null space of $M=(P(D_xH)P^TP(D_\lambda H))$ is spanned by $v=(v_1,\cdots,v_{N+1})^T$ with $v_{N+1}=1$ and $v_j=-c_j/h_{jj}$, $j=1,\cdots,N$.

(ii) $h_{jj}\neq 0$ for $j\neq k$, $h_{kk}=0$ and $c_k\neq 0$ for exactly one $k\in\{1,\cdots,N\}$. In this case $\mathcal{N}(M)$ is spanned by $v=e_k\in\mathbb{R}^{N+1}$, where e_k is the kth standard unit vector. In this case, too, y is a regular point of H, but it is also a turning point relative to the variable λ.

(iii) After further row operations (if necessary) $M=(P(D_xH)P^TPD_\lambda H)$ is of the form that

$$h_{jj}\neq 0 \text{ only for } j=1,\cdots,N-m \text{ and } c_j\neq 0 \text{ only for } j=N-m+1. \tag{2.16}$$

Then it is easily seen that $\{e_{N-m+j}\}_{j=1}^{m+1}$ is a basis for $\mathcal{N}(M)$. If, for example, $m=1$, then y is a simple bifurcation point by the Crandall-Rabinowitz theorem. If $D_xH(x,\lambda)$ is odd in x, then by Berger's theorem at least m curves bifurcate from c at y.

It should be noted that in the course of traversing c numerically, it is rather unlikely that precisely such a point will be obtained where cases (ii) or (iii) occur. The detection of turning points is relatively easy, since $\dot{\lambda}$ changes sign at such a point. However, the detection of multiple bifurcation points is more difficult, and in general, also more interesting. To accomplish this, we will make use of Sylvester's Law of Inertia, which we now briefly recall for the reader.

Let A be a symmetric $N\times N$ matrix and let $p(A)$, $n(A)$ denote the number of positive and negative eigenvalues of A, respectively. The number $\sigma(A)=p(A)-n(A)$ is called the *signature* of A. Sylvester's Law of Inertia (see, for example, [20, p. 377]) states that $\sigma(PAP^T)=\sigma(A)$ and

$$p(PAP^T)=p(A) \tag{2.17}$$

whenever P is a nonsingular matrix.

Now we may apply (2.17), (2.12) and (2.16) to detect multiple bifurcations in the course of following a curve c via the above sketched numerical continuation process. To see this, suppose that y_{i-1}, y_i are successive approximations to points on $c=\{y(s)|s\in[0,S)\}$. Suppose that the current step size $h_i>0$ is sufficiently small so that only one singular point of $D_xH(y)$ lies on C between $y(s_{i-1})$ $(\approx y_i)$. Suppose that

$$|p(D_xH(y_{i-1}))-p(D_xH(y_i))|=m. \tag{2.18}$$

Then exactly m eigenvalues of $D_xH(y((s)))$ change sign at some point $y_*=y(s_*)$ between $y(s_{i-1})$ and $y(x_i)$, that is, at a point "between" y_{i-1} and y_i. By the above-mentioned theorem of Berger, if $D_xH(x,\lambda)$ is odd in x, then at least m branches

bifurcate from c at $y(s_*)$. Furthermore, the precise difference in signature (2.18) can be easily detected, since it is equal to the change in signature of the corresponding diagonalized matrices

$$|p(P_{i-1}D_xH(y_{i-1})P_{i-1}^T)-p(P_iD_xH(y_i)P_i^T)|. \tag{2.19}$$

It is possible to determine a basis of $\mathcal{N}(D_xH(y_*))$ by ordering the diagonal elements of

$$P_{i-1}D_xH(y_{i-1})P_{i-1}^T \quad \text{and} \quad P_iD_xH(y_i)P_i^T$$

in descending order in order to determine which eigenvalues change sign at y_* and then using (2.16) to determine a basis for $\mathcal{N}(M)$, and finally applying the transformation

$$\begin{pmatrix} P^T & 0 \\ 0^T & 1 \end{pmatrix}.$$

This basis can be used to determine bifurcating directions via a Lyapunov-Schmidt process as discussed in [24] or [28]. However, we shall not pursue this approach, since it is our aim to utilize local perturbations to effect branching at detected bifurcations.

3. Branching via local perturbations. The use of local perturbations to implement numerical branching was described in [11]. For completeness, we briefly review them. One of the results used is a generalized version of a theorem of Sard (see, for example, [1]).

THEOREM 3.1. *Let $V\subset\mathbb{R}^m$, $W\subset\mathbb{R}^p$ be nonempty open sets and let $\Phi: V\times W\to\mathbb{R}^n$ be a smooth map with $m\geqq n$. If 0 is a regular value of Φ, then for almost all $d\in W$, 0 is a regular value of the restricted map $\Phi_d(\cdot)=\Phi(\cdot, d)$.*

For our particular application of the above theorem, we set $m=N+1$ and $p=N$. Now suppose that $y(s^*)$ is a detected bifurcation point on the curve $c\subset H^{-1}(0)$. Let $U\subset\mathbb{R}^{N+1}$ be a bounded open neighborhood of $y(s^*)$. Let $f:\mathbb{R}^{N+1}\to\mathbb{R}$ be a smooth map such that $f(y)=0$ for $y\notin U$ and $f(y)>0$ for $y\in U$. Then the following result holds [11].

LEMMA 3.2. *For H, U, f defined as above, let $H_d:\mathbb{R}^{N+1}\to\mathbb{R}^N$ be defined by*

$$H_d(y)=H(y)+f(y)d. \tag{3.1}$$

Then $H_d(y)$ has 0 as a regular value for almost all $d\in\mathbb{R}^N-\{0\}$.

Let us suppose that $c\subset H^{-1}(0)$ has been numerically traversed until two successive approximations y_{i-1} and y_i have been obtained for which $|p(D_xH(y_{i-1}))-p(D_xH(y_i))|$ is an odd integer. Then by the Crandall-Rabinowitz theorem, there is a bifurcation point $y(s^*)$ on C between $y(s_{i-1})\approx y_{i-1}$ and $y(s_i)\approx y_i$. Let U be a bounded open set such that $y(s^*)\in U$ and $y_{i-1}, y_i\notin U$. Let f be a smooth map such that $f(y)=0$ for $y\notin U$ and $f(y)>0$ for $y\in U$. Then by Lemma 3.2 and the implicit function theorem there are for almost all $d\in\mathbb{R}^n-\{0\}$, smooth curves C_d^- and C_d^+ such that

(i) $C_d^-\cup C_d^+\subset H_d^{-1}(0)$,

(ii) $C_d^-\cap C_d^+=\varphi$,

(iii) $y(s_{i-1})\in C_d^-$ and $y(s_i)\in C_d^+$.

Figure 3.1 illustrates this case.

In the case of multiple bifurcations the task of obtaining all of the bifurcating curves will require the use of more than one type of perturbation when the local perturbation technique is used. We illustrate this first with a simple example.

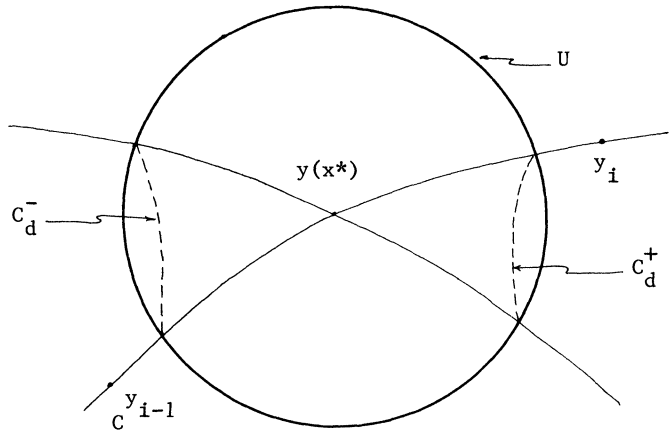


FIG. 3.1

Example 3.3. Let $H:\mathbb{R}^3\to\mathbb{R}^2$ be defined by

$$H(x_1, x_2, \lambda) = \begin{pmatrix} x_1^2-\lambda^2 \\ x_2^2-\lambda^2 \end{pmatrix}.$$

The solution curves for $H(x_1, x_2, \lambda)=0$ are the four lines parametrized by x_1

$$C^{\pm\pm} = \{(x_1, \pm x_1, \pm x_1) | x_1 \in \mathbb{R}\} \tag{3.2}$$

where the corresponding sign patterns are taken on both sides of (3.2).

Now let

$$H_d = H + d = \begin{pmatrix} x_1^2-\lambda^2+d_1 \\ x_2^2-\lambda^2+d_2 \end{pmatrix}.$$

Then the solution curves for $H_d(x_1, x_2, \lambda)=0$ vary according to d as follows. Suppose $d_1>0$ and $d_1>d_2$. Then the solution curves may be parametrized by x_1

$$C_d^{\pm\pm} = \{(x_1, \pm\sqrt{x_1^2+d_1-d_2}, \pm\sqrt{x_1^2+d_1}) | x_1 \in \mathbb{R}\}, \tag{3.3}$$

where again corresponding sign patterns are associated together. Notice that each of these curves crosses $x_1=0$, but cannot cross the planes $x_2=0$ or $\lambda=0$.

Continuing to follow this convention, we have that as

$$x_1\to\pm\infty, \quad C_c^{\pm\pm} \text{ is asymptotic to } (x_1, \pm|x_1|, \pm|x_1|).$$

(See Fig. 3.2.) Let us note that although the total set (3.3) approximates the total set (3.2) arbitrarily closely as $d\to 0$, it is nevertheless not possible to switch branches arbitrarily. Indeed, it is possible to switch between C^{++} and C^{--} via C_d^{++} and C_d^{--}, and between C^{+-} and C^{-+} via C_d^{+-} and C_d^{-+}, but it is not possible to switch between C^{+-} or C^{-+}, since on C_d^{++} or C_d^{--} the last two coordinates never vanish, and maintain the same sign. Similarly, it is possible to switch between C^{+-} and C^{-+} via C_d^{+-} and C_d^{-+}, but it is not possible to switch to C^{++} or C^{--}.

On the other hand, if $d_1<0$ and $d_2<0$, then the solution curves for $H_d(x_1, x_2, \lambda)=0$ may be parametrized by λ:

$$C_d^{\pm\pm} = \{(\pm\sqrt{\lambda^2-d_1}, \pm\sqrt{\lambda^2-d_2}, \lambda) | \lambda \in \mathbb{R}\}.$$

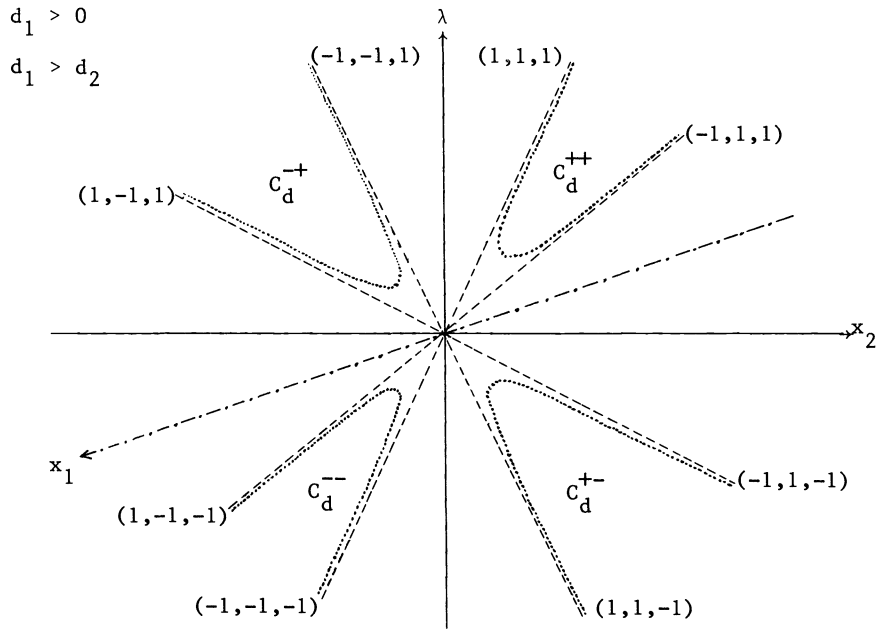


FIG. 3.2

Now as

$$\lambda \to \pm\infty, \quad C_d^{++} \text{ is asymptotic to } (\pm|\lambda|, \pm|\lambda|, \lambda).$$

This time the curves all cross $\lambda = 0$, but cannot cross $x_1 = 0$ or $x_2 = 0$.

It is now possible to switch between C^{++} and C^{+-} via C_d^{++} and C_d^{--} and between C^{--} and C^{-+} via C_d^{+-} and C_d^{-+}, but it is not possible to switch between C^{++} and C^{--} or between C^{+-} and C^{-+} via any perturbation with $d_1 < 0$ and $d_2 < 0$.

Now let us examine $D_x H$ on the various solution branches. Since

$$D_x H(x_1, x_2, \lambda) = \begin{pmatrix} 2x_1 & 0 \\ 0 & 2x_2 \end{pmatrix}$$

we have on C^{++}

$$D_x H(x_1, x_1, x_1) = \begin{pmatrix} 2x_1 & 0 \\ 0 & 2x_1 \end{pmatrix}$$

and

$$p(D_x H(x_1, x_1, x_1)) = \begin{cases} 2 & \text{if } x_1 > 0, \\ 0 & \text{if } x_1 < 0. \end{cases}$$

Similarly, on C^{+-}

$$p(D_x H(x_1, x_1, -x_1)) = \begin{cases} 2 & \text{if } x_1 > 0, \\ 0 & \text{if } x_1 < 0. \end{cases}$$

Also on C^{-+} and on C^{--}

$$D_x H(x_1, -x_1, \pm x_1) = \begin{pmatrix} 2x_1 & 0 \\ 0 & -2x_1 \end{pmatrix} \quad \text{and} \quad p(D_x H(x_1, -x_1, \pm x_1)) \equiv 1.$$

However, in the latter case, both eigenvalues have reversed their signs as x_1 passes through 0. Thus, this particular bifurcation would go undetected unless the positions of the elements in the diagonalized matrices

$$P(y_{i-1})D_xH(y_{i-1})P(y_i)^T \quad \text{and} \quad P(y_i)D_xH(y_i)P(y_i)^T$$

are also accounted for. We will return to this question below. Before doing this, let us remark upon the choice strategies for perturbations. As the above example shows and our numerical examples below also confirm this, the oscillation pattern of the components of the perturbation vector d, generally is mirrored in the oscillation of the components of the solution branch. This is again plainly seen in Fig. 3.2 for Example 3.3.

4. Practical implementation.

4.1. Solving the symmetric linear systems. Our primary objective here is the detection of multiple bifurcations and the implementation of branching. For this purpose it is often sufficient to discretize operator equations with a relatively coarse mesh so that the resulting matrix D_xH is of a size $N \leqq 500$. In this event it is reasonably practical to use an elimination process, or in some cases, a block elimination process (see, for example, [8], [9]).

Let us first discuss some particular matters of the elimination process discussed in § 3. To begin with, let us examine the elimination process when no pivoting is performed and suppose that $D_xH(y)_{ij} = h_{ij}(y)$. If $h_{11}(y) \neq 0$, then with $p_{1j} = -h_{1j}(y)/h_{11}(y)$

$$\prod_{j=2}^{N} (I + p_{1j}e_1e_j^T)(D_xH) \prod_{j=2}^{N} (I + p_{1j}e_je_1^T)$$

is of the form

$$\begin{pmatrix} h_{11}^{(1)} & 0 \\ 0 & H^{(2)} \end{pmatrix} \quad \text{where } H_{ij}^{(2)} := h_{ij}^{(2)},\ h^{(1)} = h_{11}. \tag{4.1}$$

This process may be analogously repeated if $h_{kk}^{(k)} \neq 0$ to obtain successively

$$\begin{pmatrix} h_{11}^{(1)} & & & 0 \\ & \ddots & & \\ & & h_{kk}^{(k)} & \\ 0 & & & H^{(k+1)} \end{pmatrix} \cdots \begin{pmatrix} h_1^{(1)} & & 0 \\ & \ddots & \\ 0 & & h_{NN}^{(N)} \end{pmatrix}. \tag{4.2}$$

By Sylvester's law of inertia, there is a one-to-one correspondence between the eigenvalues $\lambda_j(y(s))$ of $D_xH(y(s))$ and the elements $h_{jj}^{(j)}(y(s))$ such that

$$\lambda_j(y(s))h_{jj}^{(j)}(y(s)) \geqq 0 \tag{4.3}$$

and if equality holds, then both factors are zero. In particular, if as s (or y) varies, $h_{jj}^{(j)}(y(s))$ changes sign, then so does $\lambda_j(y(s))$. Thus, if no pivoting is performed, or if the same pivots are performed in performing the reduction processes for $D_xH(y_{i-1})$ and $D_xH(y_i)$, bifurcations are detected whenever

$$h_{jj}^{(j)}(y_{i-1})h_{jj}^{(j)}(y_i) \leqq 0 \tag{4.4}$$

holds for some $j \in \{1, \cdots, N\}$. Furthermore, the number of indices for which (4.4) holds generally determines the order of the bifurcation, assuming that a sufficiently small step size is used.

Since the reduction process (4.1)-(4.2) is a symmetric one, a pivoting amounts to permuting two rows and the corresponding columns. This means that only the diagonal

elements $h_{kk}^{(j)}$, $k \geqq j$ can be pivot elements. Thus, in monitoring (4.4) the diagonal elements should be correspondingly repermuted. If $h_{kk}^{(j)} = 0$ for $k \geqq j$, then the device mentioned in (2.12) may be employed.

4.2. Implementing perturbations. For a numerical implementation of the perturbation device it is not really necessary to produce a smooth map f as in Lemma 3.2. Suppose that in the process of traversing c with a tolerance $\varepsilon > 0$, two points y_{i-1} and y_i are obtained for which a bifurcation point between y_{i-1} and y_i is detected. A branching off from c is numerically achieved as follows:

(i) Choose a perturbation vector $d \in \mathbb{R}^N - \{0\}$ with $\|d\| \leqq \varepsilon/2$.

(ii) Replace the map H by $H_d = H + d$ and the tolerance ε by 2ε.

(iii) Using the starting point and vector y_{i-2}, u_{i-2}, and continuation process 2.2, trace the curve $c_d \subset H_d^{-1}(0)$ having $y_{i-2} \approx y_0 \in c_d$.

(iv) When a point $\tilde{y}$ is obtained such that

$$\min\{\|\tilde{y} - y_{i-1}\|, \|\tilde{y} - y_i\|\} > 2\|y_{i-1} - y_i\|,$$

then replace H_d by H and ε by $\varepsilon/2$.

(v) Perform the corrector process

$$\text{solve} \quad \begin{pmatrix} DH(\tilde{y}) \\ \tilde{u}^T \end{pmatrix} w_j = \begin{pmatrix} -H(z^{(j)}) \\ 0 \end{pmatrix}$$

set

$$z^{(j+1)} = z^{(j)} + w_j, \qquad z^{(0)} = \tilde{y}, \qquad j = 0, 1, \cdots$$

until

$$\|H(z^{(j+1)})\| < \varepsilon.$$

(vi) Resume the continuation process 2.2 for H, $\varepsilon > 0$ with starting point and vector $z^{(j+1)}$, $\tilde{u}$.

Remark. The implementation of the perturbation device in 4.2(iii) is very convenient to make, since with $H_d = H + d$, the same formulae for calculating DH still apply for calculating DH_d.

5. Numerical results. The above-described methods have been implemented and applied to a wide variety of problems. Since we are primarily concerned with illustrating the capability of handling multiple bifurcations, and to make this section somewhat brief, we will concentrate on discretizations of problems of the form

$$\begin{aligned} &\Delta u + \lambda f(u) = 0 \quad \text{on } \Omega = [0, 1]^2 \\ &u|_{\partial\Omega} = 0 \end{aligned} \tag{5.1}$$

where $f: \mathbb{R} \to \mathbb{R}$ is an odd smooth map. The eigenvalues and corresponding eigenfunctions of the linear problem

$$\begin{aligned} &\Delta u + \lambda u = 0 \quad \text{on } \Omega = [0, 1]^2, \\ &u|_{\partial\Omega} = 0 \end{aligned} \tag{5.2}$$

are known [18] to be

$$\lambda_{m,n} = (m^2 + n^2)\pi^2 \tag{5.3}$$

$$u_{m,n}(x, y) = \pm \sin m\pi x \sin n\pi y \quad \text{for } m, n = 1, 2, 3, \cdots. \tag{5.4}$$

If $m = n$, then $\lambda_{m,n}$ is a simple eigenvalue, whereas if $m \neq n$, then $\lambda_{m,n}$ is at least a double eigenvalue. Indeed, for $m^2+n^2=85$, we have solutions $(m, n) = (2, 9), (9, 2), (6, 7), (7, 6)$.

For the standard central difference approximation of the Laplacian using J interior mesh points on the x-axis and K interior mesh points on the y-axis, the eigenvalues and corresponding eigenvectors are (see, for example, [14])

$$\mu_{p,q} = 4\left[(J+1)^2 \sin^2 \frac{\pi}{2}\left(\frac{p}{J+1}\right) + (K+1)^2 \sin^2 \frac{\pi}{2}\left(\frac{q}{K+1}\right)\right] \tag{5.5}$$

for $1 \leqq p \leqq J$, $1 \leqq q \leqq K$.

$$U_{p,q}(x_j, y_k) = +\sin\left(\frac{jp\pi}{J+1}\right) \sin\left(\frac{kq\pi}{K+1}\right) \tag{5.6}$$

for

$$(x_j, y_k) = \left(\frac{j}{J+1}, \frac{k}{K+1}\right), \qquad 1 \leqq j, p \leqq J, 1 \leqq k, q \leqq k.$$

Now if $J = K = N$, then $\mu_{p,q} = \mu_{q,p}$ and the multiplicity of eigenvalues for the discrete problem mirrors that of the continuous problem. On the other hand, if $J \neq K$, then the eigenvalues (5.5) are generally simple, but those which correspond to multiple eigenvalues, that is, $p \neq q$ will be separated by a distance which is proportional to the truncation error, at least for $p \ll J$ and $q \ll K$. In any event, the central difference discretizations of (5.1) offer a rich source of multiple bifurcations or closely spaced simple bifurcations.

The particular example which we treat here is the two-dimensional plate buckling problem

$$\begin{aligned} &\Delta u + \lambda \sin u = 0 \quad \text{on } \Omega = [0, 1]^2, \\ &u|_{\partial\Omega} = 0. \end{aligned} \tag{5.7}$$

Our first discretization of (5.7) utilizes the central difference scheme with $J = K = 7$, for which the bifurcation points are given by (5.5). The resulting system of nonlinear equations is

$$\begin{pmatrix} A_N & I_N & & 0 \\ I_N & \ddots & \ddots & I_N \\ 0 & \ddots & I_N & A_N \end{pmatrix} \begin{pmatrix} u_1 \\ \vdots \\ u_N \end{pmatrix} + \frac{\lambda}{(N+1)^2} \begin{pmatrix} f(u_1) \\ \vdots \\ f(u_N) \end{pmatrix} = 0 \tag{5.8}$$

where $f(u) = \sin u$, I_N is the $N \times N$ identity matrix and A_N is the $N \times N$ tridiagonal matrix with

$$A_N = \begin{pmatrix} -4 & 1 & & 0 \\ 1 & \ddots & \ddots & 1 \\ 0 & \ddots & 1 & -4 \end{pmatrix}.$$

The solution set to (5.8) is followed throughout with accuracy tolerance $\varepsilon = 10^{-3}$. The first bifurcation point for (5.8) occurs at $\mu_{11} \approx 19.57$. Figure 5.1 illustrates the perturbed curve c_d where $d_i = 5 \times 10^{-4}$ and $f(u) = u$. In all figures, c is the curve of trivial solutions. In Figs. 5.2–5.3, the bifurcating solutions to the finite difference discretization with $f(u) = \sin u$ are portrayed. Figure 5.2 shows the solution c_d which bifurcates at $\mu_{1,1}$ with $d_i = 5 \times 10^{-4}$. Figure 5.4 shows the solutions bifurcating from

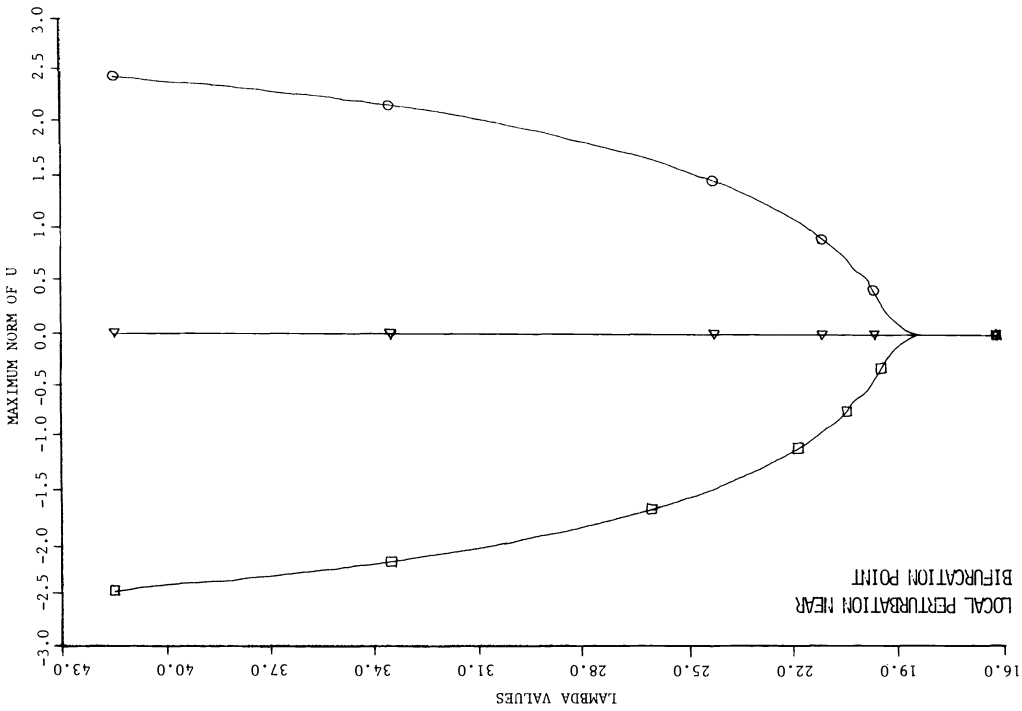


FIG. 5.2. *Solution curves of buckling problem.*

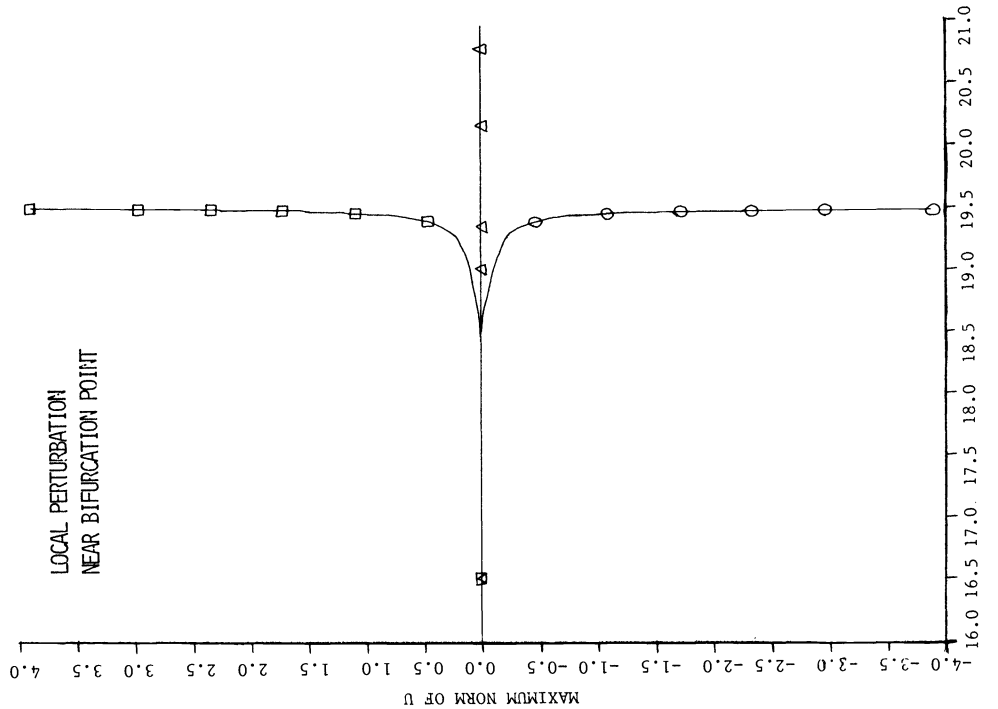


FIG. 5.1. *Solution curves of Dirichlet problem.*

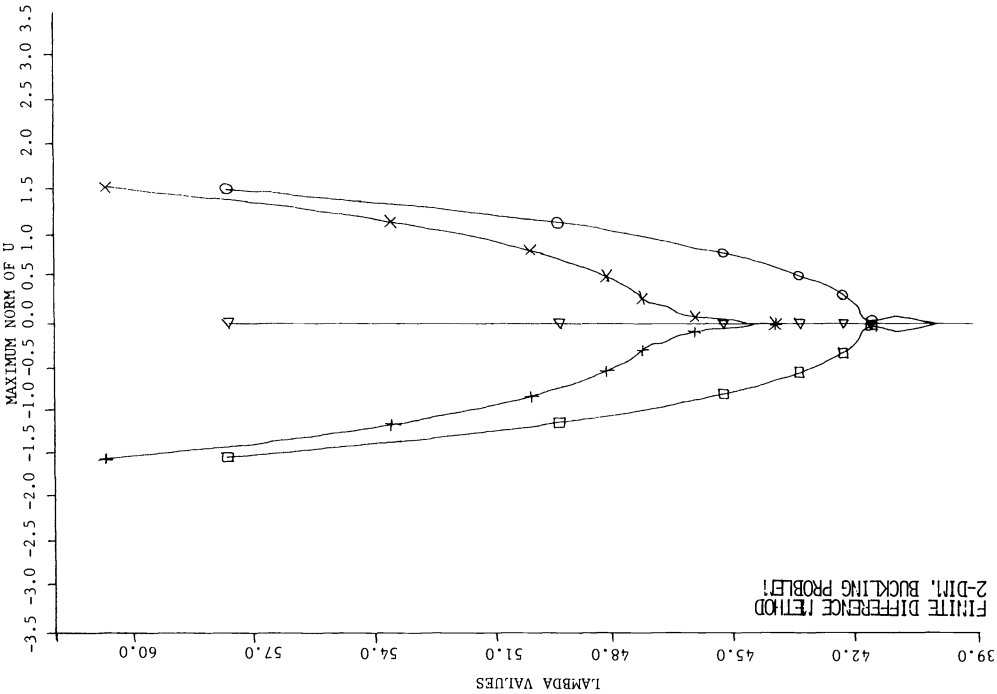


FIG. 5.4. *Double bifurcation has been separated.*

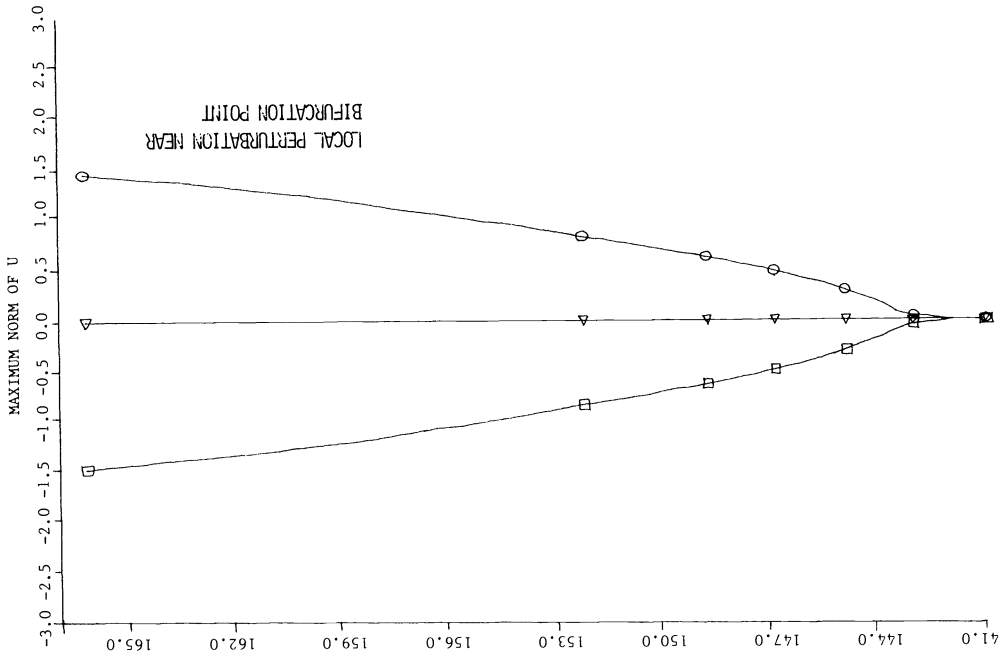


FIG. 5.3. *Solution curves of buckling problem.*

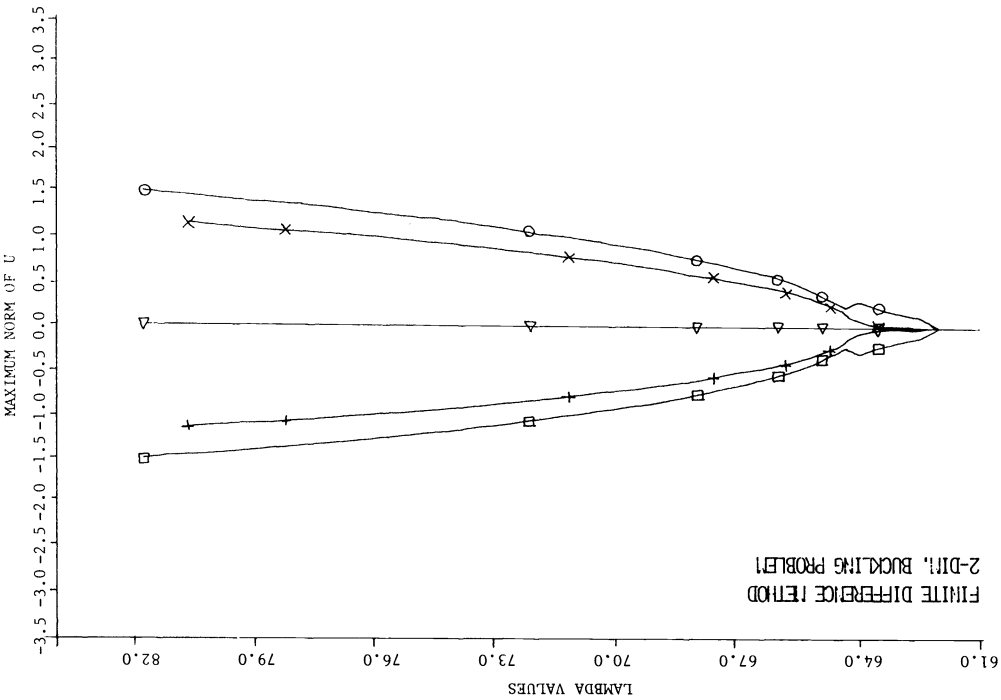


FIG. 5.6. *Perturbation on triple bifurcation point.*

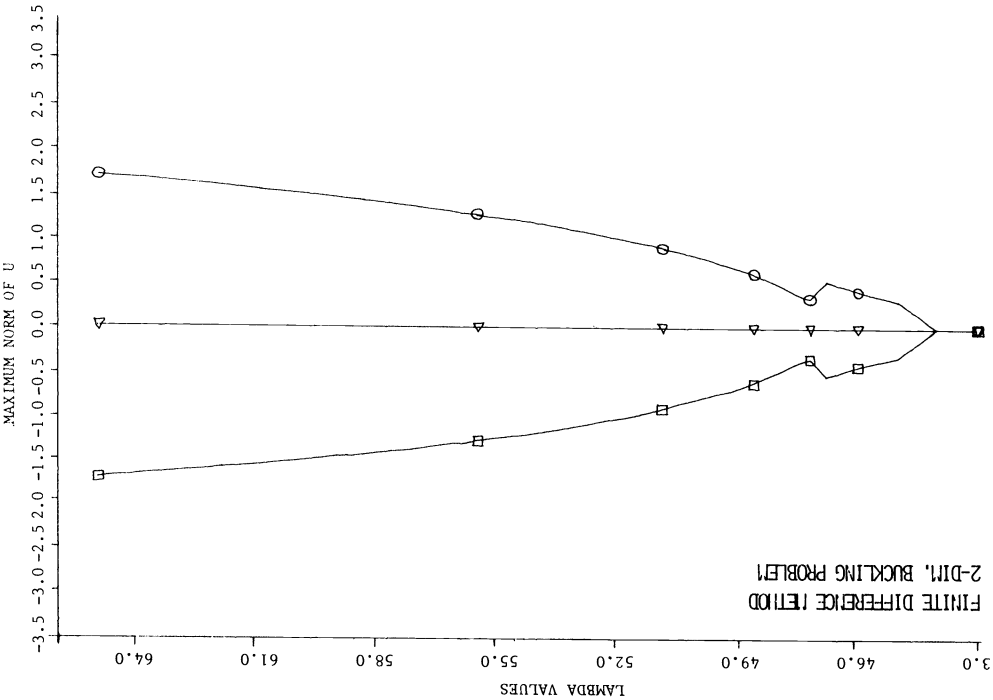


FIG. 5.5. *Perturbation near bifurcation point.*

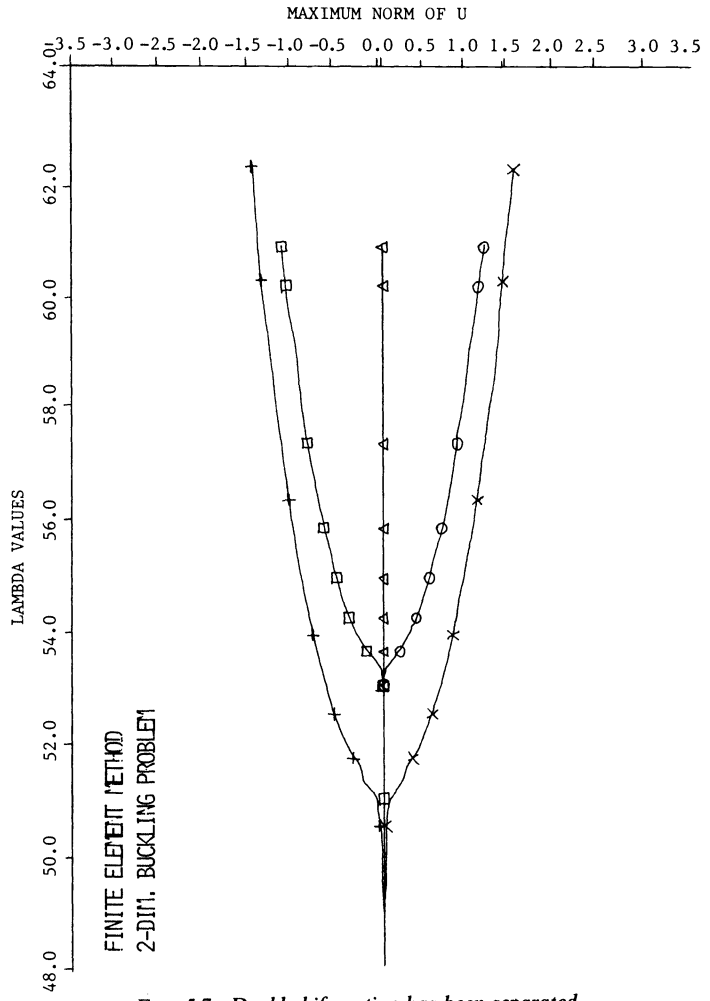


FIG. 5.7. *Double bifurcation has been separated.*

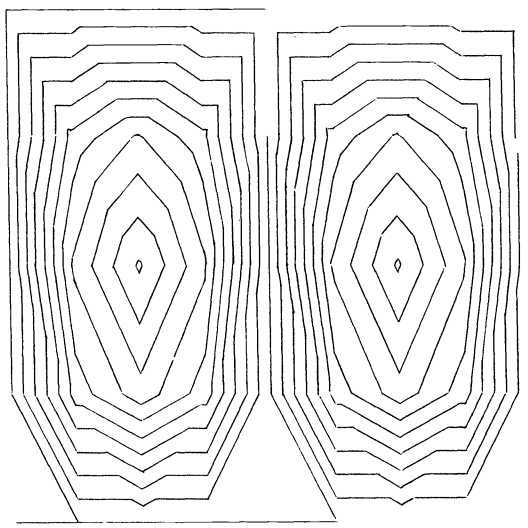


FIG. 5.8. *Contour of the buckling problem.* 21 *nodal points, lambda* = 51.23, *finite element method.*

$\mu_{1,2} \approx 41.7434$ and $u_{2,1} \approx 46.8627$ when $J = 7$ and $K = 3$. This illustrates how the double bifurcation corresponding to $\lambda_{1,2} = 5\pi^2$ has been separated via the nonsquare mesh. Figure 5.5 shows the solutions bifurcating at $\mu_{1,2} \approx 46.79$ when $J = 7 = K$. Both bifurcating solutions were obtained by choosing d so that $|d_i| \leqq 5 \times 10^{-4}$ and the components oscillated correspondingly to those of the bifurcating solutions. Figure 5.6 illustrates the bifurcating solutions which were obtained by different perturbations.

The buckling problem (5.7) was also discretized via finite elements. The domain $\Omega = [0, 1]^2$ was supplied with 32 elements and the 49 interior nodal points $(x_j, y_k) = (j/8, k/8)$, $j, k = 1, \cdots, 7$. Figure 5.7 shows the split-up bifurcation at $\nu_{1,2}$, $\nu_{2,1}$ when a nonsquare mesh with $J = 3$, $K = 7$ is used. Figures 5.8 and 5.9 show the corresponding contours of these solutions.

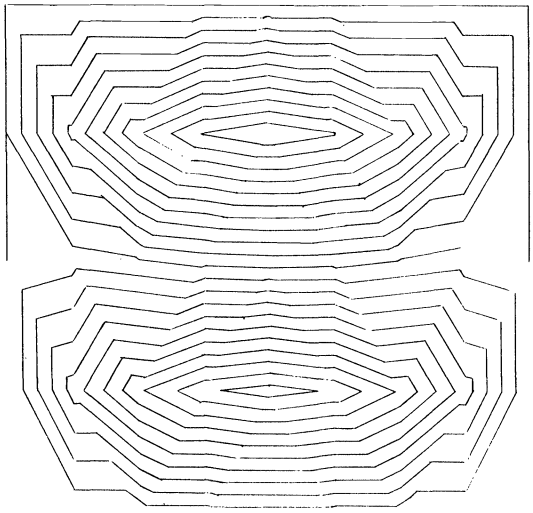


FIG. 5.9. *Contour of the buckling problem.* 21 *nodal points, lambda* = 53.44, *finite element method.*

The above numerical experiments were performed on the computing facilities (CDC CYBER 720, CYBER 205 and CYBER 170) at Colorado State University.

REFERENCES

[1] R. ABRAHAM AND J. ROBBIN, *Transversal Mappings and Flows*, W. A. Benjamin, New York, 1967.

[2] E. L. ALLGOWER, *A survey of homotopy methods for smooth mappings*, in Numerical Solutions of Nonlinear Equations, E. L. Allgower, K. Glashoff, H.-O. Peitgen, eds., Lecture Notes in Mathematics 878, Springer-Verlag, New York, 1981, pp. 1-29.

[3] E. L. ALLGOWER AND K. GEORG, *Homotopy method for approximating several solutions to a nonlinear system of equations*, in Numerical Solutions of Highly Nonlinear Problems, W. Foster, ed., North-Holland, Amsterdam, 1980, pp. 253-270.

[4] ———, *Simplicial and continuation methods for approximating fixed points and solutions to systems of equations*, SIAM Rev., 22 (1980), pp. 28-85.

[5] ———, *Relationships between deflation and global methods in the problem of approximating additional zeros of a system of nonlinear equations*, in Proc. NATO Advanced Research Institute on Homotopy Methods and Global Convergence, Vol. 13, Plenum Press, New York, pp. 31-42, 1983.

[6] ———, *Predictor-corrector and simplicial methods for approximating fixed points and zero points of nonlinear mapping*, in Mathematical Programming: The State of the Art, A. Bachem, M. Grotschel and B. Korte, eds., Springer-Verlag, New York, 1983, pp. 15-56.

[7] M. S. BERGER, *On one parameter families of real solutions of nonlinear operator equations*, Bull. Amer. Math. Soc., 75 (1969), pp. 456-459.

[8] T. F. CHAN AND H. B. KELLER, *Arc-length continuation and multigrid techniques for nonlinear elliptic eigenvalue problems*, this Journal, 3 (1982), pp. 173-194.

[9] T. F. CHAN AND Y. SAAD, *Iterative methods for solving bordered systems with applications to continuation methods*, this Journal, 6 (1985), pp. 438-451.

[10] M. G. CRANDALL AND P. H. RABINOWITZ, *Bifurcations from simple eigenvalues*, J. Funct. Anal., 8 (1971), pp. 321-340.

[11] K. GEORG, *On tracing an implicitly defined curve by quasi-Newton steps and calculating bifurcation by local perturbation*, this Journal, 2 (1981), pp. 35-50.

[12] K. GEORG, *Numerical integration of the Davidenko equation*, in Numerical Solutions of Nonlinear Equations, Lecture Notes in Mathematics 878, Springer-Verlag, New York, pp. 128-161.

[13] R. GLOWINSKI, H. B. KELLER AND L. REINHART, *Continuation-conjugate gradient methods for the least square solution of nonlinear boundary value problem*, this Journal, 6 (1985), pp. 793-832.

[14] E. ISAACSON AND H. B. KELLER, *Analysis of Numerical Methods*, John Wiley, New York, 1965.

[15] R. H. KEARFOTT, *Some general bifurcation techniques*, this Journal, 4 (1983), pp. 52-68.

[16] H. B. KELLER, *Numerical solution of bifurcation and nonlinear eigenvalue problems*, in Applications of Bifurcation Theory, P. H. Rabinowitz, ed., Academic Press, New York, 1977, pp. 348-359.

[17] T. KÜPPER, H. D. MITTELMANN AND H. WEBER, *Numerical Methods for Bifurcation Problems*, ISNM70, Birkhauser, Basel, 1984.

[18] J. R. KUTTLER AND V. G. SIGILLITO, *Eigenvalues of the Laplacian in two dimensions*, SIAM Rev., 26 (1984), pp. 163-193.

[19] J. W. MILNOR, *Topology from the Differentiable Viewpoint*, University Press of Virginia, Charlottesville, VA, 1969.

[20] L. MIRSKY, *An Introduction to Linear Algebra*, Clarendon Press, Oxford, 1963.

[21] A. R. MITCHELL AND R. WAIT, *The Finite Element Method in Partial Differential Equations*, John Wiley, New York, 1977.

[22] JEAN-CLAUDE PAUMIER, *Une methode numérique pour le calcul des points de retournement. Application a un problème aux limites non linéaire*, I. *Etude théorique et experimentation de la methode*, Numer. Math., 37 (1981), pp. 433-444.

[23] ———, *Une methode numérique pour le calcul des points de retournement. Application a un problème aux limites non-linéaire*, II. *Analyse numérique d'un problème aux limites*, Numer. Math., 37 (1981), pp. 445-452.

[24] W. C. RHEINBOLDT, *Numerical methods for a class of finite dimensional bifurcation problems*, SIAM J. Numer. Anal., 15 (1978), pp. 1-11.

[25] ———, *Solution fields of nonlinear equations and continuation methods*, SIAM J. Numer. Anal., 17 (1980), pp. 221-237.

[26] ———, *Numerical analysis continuation methods for nonlinear structural problems*, Computers and Structures, 13 (1981), pp. 103-113.

[27] W. C. RHEINBOLDT AND J. V. BURKHARDT, *Algorithm* 596: *A program for a locally parametrized continuation process*, ACM Trans. Math. Software, 9 (1983), pp. 236-241.

[28] I. STAKGOLD, *Branching of solutions of nonlinear equations*, SIAM Rev., 13 (1971), pp. 289-331.

[29] T. J. YPMA, *Following paths through turning points*, BIT, 22 (1982), pp. 368-383.

SIAM J. SCI. STAT. COMPUT.
Vol. 7, No. 4, October 1986

014

ON THE RECONCILIATION OF CLASHING BOUNDARY CONDITIONS IN CELL DISCRETIZATION*

JOHN GREENSTADT†

Abstract. In most physical problems, there are conservation laws which take the form of integral identities. Some of these are "flow balance" conditions across interfaces. In the process of discretization, some or all of the integral identities may be lost, thereby leading to violations of material or energy balance, etc. In the cell discretization (CD) method, it is possible to impose the necessary conservation laws across interfaces as part of the discretization process. However, the variational procedure, on which the CD method is based, can induce "natural interface conditions" which are inconsistent with the imposed conditions. The aim of this work is to derive the necessary "nullifier" to be added to the variational functional, so as to cancel automatically the unwanted induced condition. This nullifier is derived for the classical continuous formulation, and then for the CD formulation, where it is shown how the rank of the discrete nullifier can be reduced to a minimum. The limitations of the nullification procedure in the discrete context are discussed.

Key words. discretization, variational methods, partial differential equations, conservation laws, natural boundary conditions

AMS(MOS) subject classifications. 65N30, 65N35

1. Classical conservation laws. For most of the physical systems described by partial differential equations, there are so-called *conservation laws*; that is, there are quantities related to the dependent variables in the equations which bear relations to each other of a particular form. If x is a vector of independent variables in R^N, $S(x)$ is a source term and $J(x)$ is a vector current, then the typical form of a conservation law in a domain Ω is

$$\int_\Omega S\,d\Omega = \int_\Omega \nabla \cdot J\,d\Omega. \tag{1.1}$$

S may or may not contain time derivatives; in this treatment we shall consider the steady state only, wherein time derivatives do not enter. $\nabla \cdot J$ indicates the divergence of the vector J, and is equal to $\sum_i \partial J_i/\partial x_i$ in terms of components. Note that if (1.1) holds for any arbitrary subdomain of Ω, then the "local," or differential form of the conservation law holds, viz.,

$$S = \nabla \cdot J. \tag{1.2}$$

The second integral over the interior of Ω in (1.1) can be transformed into a surface integral over the boundary Γ of Ω by the application of the well-known *integral identity*

$$\int_\Omega \nabla \cdot J\,d\Omega = \int_\Gamma n \cdot J\,d\Gamma \tag{1.3}$$

which is variously called Gauss' theorem or Green's theorem. The vector n is the outward normal to the surface Γ. Equation (1.1) now becomes

$$\int_\Omega S\,d\Omega = \int_\Gamma J_n\,d\Gamma, \tag{1.4}$$

where J_n stands for the normal component of J given by $n \cdot J$.

* Received by the editors February 1, 1984, and in revised form August 6, 1985.

† Department of Applied Mathematics and Theoretical Physics, Cambridge University, Cambridge, England CB3 9EW.

Let us now partition the domain Ω into two subdomains, Ω_1 and Ω_2, separated by a common interface Γ_{12}. That part of Γ which belongs only to Ω_1 we shall denote by Γ_1, and the part of Γ which belongs only to Ω_2 we shall denote by Γ_2. Hence, the conservation law (1.4) "divides" into two separate laws:

$$\begin{aligned}\int_{\Omega_1} S_1\, d\Omega_1 &= \int_{\Gamma_1} J_{1n}\, d\Gamma_1 + \int_{\Gamma_{12}} J_{1n}\, d\Gamma_{12},\\ \int_{\Omega_2} S_2\, d\Omega_2 &= \int_{\Gamma_2} J_{2n}\, d\Gamma_2 + \int_{\Gamma_{12}} J_{2n}\, d\Gamma_{12}.\end{aligned} \tag{1.5}$$

When we add these two together, we should recover (1.4). In fact, using the obvious facts that

$$\int_{\Omega_1} S_1\, d\Omega_1 + \int_{\Omega_2} S_2\, d\Omega_2 = \int_{\Omega} S\, d\Omega \tag{1.6}$$

and

$$\int_{\Gamma_1} J_{1n}\, d\Gamma_1 + \int_{\Gamma_2} J_{2n}\, d\Gamma_2 = \int_{\Gamma} J_n\, d\Gamma \tag{1.7}$$

the sum of the two equations in (1.5) reduces to

$$\int_{\Omega} S\, d\Omega = \int_{\Gamma} J_n\, d\Gamma + \int_{\Gamma_{12}} (J_{1n} + J_{2n})\, d\Gamma_{12}. \tag{1.8}$$

Since the Ω's and Γ's are arbitrary (except, of course, that the Γ's are the boundaries of the Ω's), it is clear that, if (1.8) is to match (1.4), we must have

$$J_{1n} + J_{2n} = 0, \tag{1.9}$$

which shows that the currents must (in general) balance across an interface. (We have found it more convenient to change the sign of one of the normals at each interface. With this convention, (1.9) becomes

$$J_{1n} = J_{2n} \quad .) \tag{1.10}$$

The main reason we have gone through this well-known elementary derivation of the conservation across interfaces is to emphasize that it depends on the validity of the *analytic* identity (1.3). In the process of discretization, with which we are primarily concerned, the conservation across interfaces may be lost. Our aim is to investigate the conditions under which we may still enforce it (and in what form) as part and parcel of the discretization process.

When a partial differential equation is derivable from a variational formulation, the variational process itself may sometimes interfere with the validity of the interface conservation law. We shall show by example how this may occur. Let us therefore consider the functional $\Phi[u]$, which is the integral over Ω of an integrand F, the Lagrange function, which contains only the coordinate vector x, the dependent variable $u(x)$, and its gradient ∇u. We then have

$$\Phi[u] \equiv \int_{\Omega} F(x, u, \nabla u)\, d\Omega. \tag{1.11}$$

The standard procedure of the calculus of variations (see [1]) consists of adding a multiple ε of an arbitrary function ϕ to u, and calculating the change (to the first

order in ε) induced in Φ thereby. Thus, we replace u by $u+\varepsilon\phi(x)$, substitute this into $\Phi[u]$, and compute the derivative of Φ with respect to ε at $\varepsilon=0$. The result is ([1], p. 153)

$$\left(\frac{\partial\Phi}{\partial\varepsilon}\right)_{\varepsilon=0}=\int_\Omega\left(\frac{\partial F}{\partial u}-\nabla\cdot\frac{\partial F}{\partial\nabla u}\right)\phi\,d\Omega+\int_\Omega\nabla\cdot\left(\frac{\partial F}{\partial\nabla u}\,\phi\right)d\Omega, \tag{1.12}$$

which is the sum of the integrals of an Euler variational derivative and of a divergence expression. The latter can be transformed into an integral over the boundary Γ by means of (1.3), so that (1.12) becomes

$$\left(\frac{\partial\Phi}{\partial\varepsilon}\right)_{\varepsilon=0}=\int_\Omega\left(\frac{\partial F}{\partial u}-\nabla\cdot\frac{\partial F}{\partial\nabla u}\right)\phi\,d\Omega+\int_\Gamma\left(n\cdot\frac{\partial F}{\partial\nabla u}\right)\phi\,d\Gamma. \tag{1.13}$$

We are now in a position to write down the necessary condition for a stationary Φ by simply setting $\partial\Phi/\partial\varepsilon$ to zero. But first, for conciseness, we shall condense the notation by defining

$$\begin{aligned} A&\equiv\frac{\partial^2F}{\partial\nabla u\partial\nabla u}\equiv\left\{\frac{\partial^2F}{\partial u_i'\partial u_j'}\right\},\\ u_n'&\equiv\frac{\partial u}{\partial n}\equiv n\cdot A\cdot\nabla u,\\ \frac{\partial F}{\partial u_n'}&\equiv n\cdot\frac{\partial F}{\partial\nabla u}. \end{aligned} \tag{1.14}$$

A is a dyadic which corresponds to the Hessian matrix of F with respect to the u_i''s, and u_n' is the covariant normal derivative (sometimes called the coderivative) of u. We are defining $\partial F/\partial u_n'$ as a shorthand for the right-hand side of the last equation, but if F is quadratic in ∇u (the most important case), then the equation is really an identity. With this notation, the necessary condition for stationary Φ takes the form

$$\int_\Omega\left(\frac{\partial F}{\partial u}-\nabla\cdot\frac{\partial F}{\partial\nabla u}\right)\phi\,d\Omega+\int_\Gamma\frac{\partial F}{\partial u_n'}\,\phi\,d\Gamma=0. \tag{1.15}$$

The standard argument of the Calculus of Variations continues by making use of the relative arbitrariness of ϕ, and various lemmas based on it. In rough outline, we first assume that ϕ vanishes identically on Γ, and everywhere in Ω, except in a small neighborhood of an arbitrary point x. The basic argument (see [1]) of the Fundamental Lemma of the Calculus of Variations then shows that the first integrand in (1.15) must vanish at x, hence everywhere in Ω. Thus,

$$\frac{\partial F}{\partial u}-\nabla\cdot\frac{\partial F}{\partial\nabla u}=0 \tag{1.16}$$

at every point x in the interior of Ω and this is, of course, the Euler equation. Hence, the integral over Ω in (1.15) vanishes, and we are left with the integral over Γ. We then consider a variation ϕ whose support is contained in Γ alone, and which also vanishes everywhere on Γ except for a neighborhood of an arbitrary point s. The same lemma then shows that

$$\frac{\partial F}{\partial u_n'}=0 \tag{1.17}$$

holds at the point s on Γ and hence, everywhere on Γ. This is, of course, the natural, or *induced*, boundary condition.

When Ω is partitioned, as before, into two subdomains Ω_1 and Ω_2, the same variational process described above leads to the pointwise relation on the interface Γ_{12},

$$\frac{\partial F}{\partial u'_{1n}} = \frac{\partial F}{\partial u'_{2n}}, \tag{1.18}$$

provided $u(x)$ is assumed to be continuous across Γ_{12}. In selfadjoint problems, $\partial F/\partial u'_n$ may be readily identified with the normal component of the current, but in nonself-adjoint problems, this is generally not so. It may therefore be necessary to *impose* the interface constraint in some form.

In the case of boundary conditions, an *imposed* condition may clash with the *induced* condition, particularly if the imposed condition is of inhomogeneous Neumann or of mixed type. In the next section, we shall consider the reconciliation of such inconsistent boundary conditions before returning to the treatment of interface conservation relations within the framework of Cell Discretization (CD). In the latter, we may sometimes wish to impose the interface conditions corresponding to conservation relations, in case they are not induced by the variational procedure. In that case, we may have the task of reconciling conflicts which might arise with the interface conditions which are *thereby* induced. If this can be done, the conservation relations can be "built in" to the discrete version of the problem, while preserving its consistency and hence, well-posedness.

2. The nullification of induced boundary conditions. So far, we have said little of the boundary conditions which might be imposed on the dependent variable $u(x)$ in a given problem. Let us assume that such a condition has the form

$$U(u, u'_n) = R, \tag{2.1}$$

where u and u'_n are evaluated on the boundary Γ (and hence are functions of s). We need not assume that U is linear or homogeneous in u or u'_n, but only that it be differentiable with respect to either of them. The "inhomogeneous part," R, is assumed to be independent of u and u'_n.

In the most elementary variational treatment, when Dirichlet conditions are imposed, ϕ is constrained to vanish on the boundary, so as not to disturb the values of $u(s)$ there. For our purposes, it is convenient to incorporate our boundary condition as an a posteriori constraint, and to let the function ϕ, which appears in the variational procedure, be arbitrary in Ω and on Γ. We shall therefore add to the original functional in (1.5) a term containing a Γ-integration, so that we have

$$\Phi[u] \equiv \int_\Omega F(x, u, \nabla u)\, d\Omega - \int_\Gamma \lambda(s)(U(u, u'_n) - R)\, d\Gamma. \tag{2.2}$$

When we calculate the necessary condition for stationary Φ, corresponding to (1.10), we find

$$\int_\Omega \left(\frac{\partial F}{\partial u} - \nabla \cdot \frac{\partial F}{\partial \nabla u}\right)\phi\, d\Omega + \int_\Gamma \frac{\partial F}{\partial u'_n}\phi\, d\Gamma - \int_\Gamma \lambda\left(\frac{\partial U}{\partial u}\phi + \frac{\partial U}{\partial u'_n}\phi'_n\right) d\Gamma = 0. \tag{2.3}$$

As before, we shall assume the Euler equation to be satisfied, and are left with the Γ-integrations, which we regroup to obtain

$$\int_\Gamma \left(\frac{\partial F}{\partial u'_n} - \lambda\frac{\partial U}{\partial u}\right)\phi\, d\Gamma - \int_\Gamma \lambda \frac{\partial U}{\partial u'_n}\phi'_n\, d\Gamma = 0. \tag{2.4}$$

Since ϕ is assumed to be arbitrary, it is not hard to show that it can be chosen so that

its normal derivative vanishes everywhere on Γ, while the support of ϕ itself is an arbitrarily small neighborhood of some point on Γ. Likewise, it can be shown that a ϕ can be constructed which vanishes everywhere on the boundary, but the support of whose normal derivative is also an arbitrarily small neighborhood of some point on Γ. Hence, the integrands in both integrals in (2.4) must vanish individually. Rearranging factors, we thus have

$$\frac{\partial U}{\partial u}\lambda = \frac{\partial F}{\partial u'_n}, \qquad \frac{\partial U}{\partial u'_n}\lambda = 0. \tag{2.5}$$

This is an overdetermined system (of two equations) for the unknown λ. Hence, there is either no solution, or an infinity of them. In order to ensure that there is a solution, the right-hand side of (2.5) must satisfy the Fredholm condition, viz., it must be orthogonal to all left null-vectors of the coefficient matrix on the left. Since the latter is only a (2×1) matrix, there is only one such vector (up to an arbitrary factor). A convenient one is the vector L, given by

$$L = \begin{pmatrix} \dfrac{\partial U}{\partial u'_n} \\ \dfrac{-\partial U}{\partial u} \end{pmatrix}. \tag{2.6}$$

If we premultiply (2.5) by L^T, the left-hand side clearly vanishes identically; therefore, the product of L^T with the right-hand side must also vanish, so that we have

$$0 = -\frac{\partial U}{\partial u'_n}\frac{\partial F}{\partial u'_n}. \tag{2.7}$$

We note in passing that if (2.7) *is* satisfied, a solution for λ could be obtained by premultiplying (2.5) by the "other" vector $(\partial U/\partial u, \partial U/\partial u'_n)$ with the result

$$\lambda = \left\{\left(\frac{\partial U}{\partial u}\right)^2 + \left(\frac{\partial U}{\partial u'_n}\right)^2\right\}^{-1}\frac{\partial U}{\partial u}\frac{\partial F}{\partial u'_n}. \tag{2.8}$$

It is clear from (2.7) that, if U is "Dirichlet-like," i.e., if it does not contain u'_n, there is no problem, since the right-hand side also vanishes identically. On the other hand, if U is "Neumann-like," or "mixed," in that it *does* contain u'_n, then $\partial F/\partial u'_n$ must vanish. This is just the natural boundary condition. Only if this condition happens to be equivalent to (2.1) is the boundary value problem consistent; otherwise, there is a contradiction, and the problem is not well posed.

It is well known that, in order to rectify this contradiction automatically, additional surface integrals must be added to the functional Φ in (2.2). We shall give a somewhat more detailed treatment of this device than is usual, to make it easier to relate it to the Cell Discretization procedure. We add a term consisting of the Γ- integral of a function $N(u, u'_n)$, which we call a *nullifier*, since its purpose is to nullify the unwanted induced boundary conditions. Φ thus becomes

$$\Phi[u] \equiv \int_\Omega F(x, u, \nabla u)\, d\Omega - \int_\Gamma \lambda(s) U(u, u'_n)\, d\Gamma + \int_\Gamma N(u, u'_n)\, d\Gamma. \tag{2.9}$$

Note that the added term does not involve a constraint, as does the second term, nor does it contain an undetermined multiplier. Our aim is to determine the form of N so that the variationally formulated boundary value problem is consistent.

We therefore repeat the preceding steps of calculating the variation of Φ and setting the result to zero to obtain

$$\int_\Omega \left(\frac{\partial F}{\partial u} - \nabla \cdot \frac{\partial F}{\partial \nabla u}\right)\phi \, d\Omega + \int_\Gamma \frac{\partial F}{\partial u_n'}\phi \, d\Gamma - \int_\Gamma \lambda\left(\frac{\partial U}{\partial u}\phi + \frac{\partial U}{\partial u_n'}\phi_n'\right) d\Gamma + \int_\Gamma \left(\frac{\partial N}{\partial u}\phi + \frac{\partial N}{\partial u_n'}\phi_n'\right) d\Gamma = 0. \tag{2.10}$$

When the terms are regrouped, the result is

$$\frac{\partial U}{\partial u}\lambda = \frac{\partial N}{\partial u} + \frac{\partial F}{\partial u_n'}, \qquad \frac{\partial U}{\partial u_n'}\lambda = \frac{\partial N}{\partial u_n'}. \tag{2.11}$$

This is still an overdetermined system of two equations for the unknown λ. We therefore apply the same left null-vector L to (2.11), as before, to obtain the following consistency constraint on N:

$$\frac{\partial U}{\partial u_n'}\frac{\partial N}{\partial u} - \frac{\partial U}{\partial u}\frac{\partial N}{\partial u_n'} = -\frac{\partial U}{\partial u_n'}\frac{\partial F}{\partial u_n'}. \tag{2.12}$$

This is a linear, first-order p.d.e., for which the general solution can be found by a standard procedure (the method of characteristics) [2]. The result can be expressed as follows: The equation $U(u, u_n') = \tau$ is first solved for u_n', and the result expressed in terms of the inverse function H, viz.,

$$H(u, \tau) = H[u, U(u, u_n')] \equiv u_n'. \tag{2.13}$$

If we next denote $\partial F/\partial u_n'$ by $G(u, u_n')$, then the general solution of (2.11) is given by

$$N(u, u_n') = \int^u G\{s, H[s, U(u, u_n')]\}\, ds + \Lambda[U(u, u_n')]. \tag{2.14}$$

The first term is an indefinite integral with respect to the dummy variable s, and u is its upper limit. The second term consists of an arbitrary function Λ of the argument U.

We shall illustrate this procedure with an example. Let

$$U(u, u_n') \equiv Pu + Qu_n'. \tag{2.15}$$

The application of (2.14) yields (provided $Q \neq 0$)

$$N(u, u_n') = -\frac{1}{2}\frac{P}{Q}u^2 - uu_n' + \Lambda(U). \tag{2.16}$$

In [3], Λ was chosen to be a multiple of the square of U so that some attempt could be made to minimize the degradation of the positivity of Φ due to the addition of N. A convenient form of N was found to be

$$N(u, u_n') = -\frac{1}{2}\frac{P}{Q}(\mu PQ - 1)u^2 + (\mu PQ - 1)uu_n' + \frac{1}{2}\mu Q^2 u_n'^2, \tag{2.17}$$

and various choices were made for μ, based on the magnitudes and signs of P and Q. In [4] and [5], these nullifiers were applied in Neumann and mixed boundary value problems. The computed solutions to these problems were grossly incorrect without nullifiers, but correct with them. However, even though this "classical" nullifier works well enough, it is not really integrated with the basic technique of cell discretization, and amounts to an "overcorrection" of the functional (i.e., its rank is too high).

So far we have said nothing about the treatment of more domains than one, and have not really considered true interface conditions, rather than merely boundary conditions. In fact, under fairly general circumstances, the interface correction problem can be reduced to the boundary correction problem which we have just solved in undiscretized form. To ascertain when this can be done, let us consider a more general interface condition, involving cells Ω_1 and Ω_2 with interface Γ_{12}, of the form:

$$U(u_1, u'_{1n}, u_2, u'_{2n}) = R. \tag{2.18}$$

The nullifier for this imposed condition will then have the general form

$$N \equiv N(u_1, u'_{1n}, u_2, u'_{2n}), \tag{2.19}$$

so that those parts of the functional Φ for this problem, which have to do with Ω_1, Ω_2 and Γ_{12} are as follows:

$$\begin{aligned}\Phi[u_1, u_2] \equiv & \int_{\Omega_1} F_1(x, u_1, \nabla u_1)\, d\Omega_1 + \int_{\Omega_2} F_2(x, u_2, \nabla u_2)\, d\Omega_2 \\ & - \int_{\Gamma_{12}} \lambda(s)(U(u_1, u'_1, u_2, u'_2) - R)\, d\Gamma_{12} \\ & + \int_{\Gamma_{12}} N(u_1, u'_1, u_2, u'_2)\, d\Gamma_{12}.\end{aligned} \tag{2.20}$$

When we repeat the procedure previously described for obtaining the equations for λ, we get

$$\begin{pmatrix} \dfrac{\partial U}{\partial u_1} \\ \dfrac{\partial U}{\partial u'_1} \\ \dfrac{\partial U}{\partial u_2} \\ \dfrac{\partial U}{\partial u'_2} \end{pmatrix} \lambda = \begin{pmatrix} \dfrac{\partial N}{\partial u_1} + \dfrac{\partial F_1}{\partial u'_1} \\ \dfrac{\partial N}{\partial u'_1} \\ \dfrac{\partial N}{\partial u_2} + \dfrac{\partial F_2}{\partial u'_2} \\ \dfrac{\partial N}{\partial u'_2} \end{pmatrix}. \tag{2.21}$$

The coefficient matrix in (2.21) has *three* left null-vectors, since its rank is only unity. These vectors are not unique, but are determined only up to independent linear combinations. However, a convenient choice of the three vectors with useful symmetry (which we shall combine into a *matrix* L) is the following:

$$L \equiv \begin{pmatrix} \dfrac{\partial U}{\partial u'_1} & 0 & 0 \\ -\dfrac{\partial U}{\partial u_1} & 0 & \dfrac{\partial U}{\partial u'_2} \\ 0 & \dfrac{\partial U}{\partial u'_2} & 0 \\ 0 & -\dfrac{\partial U}{\partial u_2} & -\dfrac{\partial U}{\partial u'_1} \end{pmatrix}. \tag{2.22}$$

Applying the Fredholm condition for consistency again, the result of premultiplying

the right-hand side of (2.21) by L^T must be zero. This yields the following system of constraints on N:

$$\begin{aligned}
&\frac{\partial U}{\partial u_1'}\frac{\partial N}{\partial u_1}-\frac{\partial U}{\partial u_1}\frac{\partial N}{\partial u_1'}=-\frac{\partial U}{\partial u_1'}\frac{\partial F_1}{\partial u_1'},\\
&\frac{\partial U}{\partial u_2'}\frac{\partial N}{\partial u_2}-\frac{\partial U}{\partial u_2}\frac{\partial U}{\partial u_2'}=-\frac{\partial U}{\partial u_2'}\frac{\partial F_2}{\partial u_2'},\\
&\frac{\partial U}{\partial u_2'}\frac{\partial N}{\partial u_1'}-\frac{\partial U}{\partial u_1'}\frac{\partial N}{\partial u_2'}=0.
\end{aligned} \tag{2.23}$$

We must now determine whether the system (2.23) is consistent, and does in fact have at least one solution for N. The simplest way to do this is by replacing N by a new function M which satisfies the homogeneous equations corresponding to (2.23) with the right-hand sides set to zero. The inhomogeneous part of N is derived in a manner analogous to that described for the boundary correction, except that we now have *two* functions H_1 and H_2, defined by solving $U(u_1, u_1', u_2, u_2')=\tau$ first for u_1' and then for u_2'. The results are:

$$\begin{aligned}
&H_1[u_1, \tau, u_2, u_2']=H_1[u_1, U(u_1, u_1', u_2, u_2'), u_2, u_2']\equiv u_1',\\
&H_2[u_2, \tau, u_1, u_1']=H_2[u_2, U(u_1, u_1', u_2, u_2'), u_1, u_1']\equiv u_2'.
\end{aligned} \tag{2.24}$$

We next define $G_1(u_1, u_1')\equiv \partial F_1/\partial u_1'$, and $G_2(u_2, u_2')\equiv \partial F_2/\partial u_2'$, and calculate the two functions $J_1(u_1, u_1', u_2, u_2')$ and $J_2(u_1, u_1', u_2, u_2')$ as indefinite integrals:

$$\begin{aligned}
&J_1\equiv\int^{u_1} G_1\{\sigma, H_1[\sigma, U(u_1, u_1', u_2, u_2'), u_2, u_2']\}\, d\sigma,\\
&J_2\equiv\int^{u_2} G_2\{\sigma, H_2[\sigma, U(u_1, u_1', u_2, u_2'), u_1, u_1']\}\, d\sigma.
\end{aligned} \tag{2.25}$$

We then assume N to have the form

$$N\equiv J_1+J_2+M. \tag{2.26}$$

Bearing in mind that the equations in (2.24) are *identities* in u_1, u_1', u_2 and u_2', it can be worked out that M satisfies the equations:

$$\begin{aligned}
&\frac{\partial U}{\partial u_1'}\frac{\partial M}{\partial u_1}-\frac{\partial U}{\partial u_1}\frac{\partial M}{\partial u_1'}=0,\\
&\frac{\partial U}{\partial u_2'}\frac{\partial M}{\partial u_2}-\frac{\partial U}{\partial u_2}\frac{\partial M}{\partial u_2'}=0,\\
&\frac{\partial U}{\partial u_2'}\frac{\partial M}{\partial u_1'}-\frac{\partial U}{\partial u_1'}\frac{\partial M}{\partial u_2'}=0,
\end{aligned} \tag{2.27}$$

which is a linear homogeneous system, and may be tested for consistency by a standard method [6]. The equations turn out in fact to be consistent, and to have a single general solution of the form

$$M(u_1, u_1', u_2, u_2')=\Lambda[U(u_1, u_1', u_2, u_2')], \tag{2.28}$$

where Λ is an arbitrary function of its single argument.

It thus appears that an interface nullifier can be found for the rather general class of imposed interface conditions shown in (2.18). Unfortunately, the nullifier is very

complicated to apply and, to a large extent, spoils the cellwise treatment of discretization described in [4]. However, if $U(u_1, u_1', u_2, u_2')$ happens to have the special form

$$U(u_1, u_1', u_2, u_2') \equiv U_1(u_1, u_1') - U_2(u_2, u_2'), \tag{2.29}$$

then N can be similarly split, and the part associated with each cell can be treated independently of the other, in the manner described for boundary corrections.

Fortunately, practically all imposed interface conditions in physical applications are of the form (2.29). Obviously, any *linear* interface constraint can be put in this form, but even most nonlinear constraints can as well, the best-known example being the very important Rankine-Hugoniot shock conditions [7]. Hence, we shall only concern ourselves further with interface constraints which decompose into paired boundary constraints.

It is thus only necessary to discuss the boundary correction for a single (typical) cell, with the understanding that other, contiguous cells might lie on the other sides of some or all of the segments of its boundary. Rather than describe the full apparatus of CD applied to many cells [4], we shall indicate briefly in § 6 how the presence of neighbors would be incorporated into the nullifier for the single cell.

3. Discretization procedure for linear problems in one cell. Because of its difficulty, we shall defer to a later report the discussion of general nonlinear problems—those for which $F(x, u, \nabla u)$ and $U(u, u_n')$ have an arbitrary functional form. Linear problems—those for which F is quadratic in u and ∇u, and U is linear in u and u_n'—are of course much simpler and better understood. Moreover, the treatment of linear problems will introduce various auxiliary quantities in a natural way, and generalizations of these quantities are useful in making headway on the nonlinear (or quasilinear) problems. As a further simplification, based on the discussion of "separable" interface conditions in the previous section, we lose little generality by restricting the discussion to one (typical) cell.

We shall therefore assume

$$F(x, u, \nabla u) \equiv \tfrac{1}{2}\nabla u \cdot A(x) \cdot \nabla u + \tfrac{1}{2}b(x)u^2 - c(x)u \tag{3.1}$$

and

$$U(u, u_n') \equiv P(s)u + Q(s)u_n'. \tag{3.2}$$

The quantities A, b and c may depend on the independent variable x, defined as lying inside the domain Ω, and P and Q may depend on the variable s, defined as lying on the boundary Γ. Note that $A(x)$ must be a diadic, but the rest of the coefficients are scalars.

The first step in the CD process (described in [4]) is to select a representation of $u(x)$ which has a predetermined functional form in a given cell, but depends on a finite set of M parameters $\{\theta_\mu;\ \mu = 1, \cdots, M\}$. Thus,

$$u(x) \sim \psi(x, \theta) \quad \text{in } \Omega. \tag{3.3}$$

For linear problems, the obvious and best choice is

$$\psi(x, \theta) = \sum_{\mu=1}^{M} \theta_\mu \phi_\mu(x). \tag{3.4}$$

We shall refer to the set of ϕ's as the *basis functions* in Ω. Since they are subject to our choice, we assume that they are linearly independent in Ω, though not necessarily

mutually orthogonal, and that, in the limit as $\mu \to \infty$, they form a Schauder basis in the Sobolev space $H^1(\Omega)$, within which ψ must be contained.

Substituting this representation of ψ into (1.11) with the F defined in (3.1), we obtain the *discretized* form of Φ,

$$\Phi = \tfrac{1}{2}\sum_{\mu}\sum_{v} S_{\mu v}\theta_\mu\theta_v - \sum_{\mu} T_\mu\theta_\mu \tag{3.5}$$

with appropriate ranges for μ and v. The coefficient arrays are defined by

$$S_{\mu v} \equiv \int_\Omega (\nabla\phi_\mu \cdot A \cdot \nabla\phi_v + b\phi_\mu\phi_v)\, d\Omega \tag{3.6}$$

and

$$T_\mu \equiv \int_\Omega c\phi_\mu\, d\Omega. \tag{3.7}$$

We must obviously include the boundary conditions somehow, as constraints on the θ's. In the *CD* method, this is done by means of the special type of collocation we refer to as *moment* collocation [4]. This type of collocation calls for a set of *weight functions* $\{w_\alpha(s);\ \alpha = 1, \cdots, L\}$, defined on Γ. The boundary condition (2.1) is then discretized by requiring only the finite set of conditions

$$\int_\Gamma w_\alpha(s)(U(\psi, \psi'_n) - R)\, d\Gamma = 0, \qquad \alpha = 1, \cdots, L \tag{3.8}$$

instead of the infinite set consisting of having (2.1) satisfied at every point on Γ.

Since we are in a position to choose the w's, we naturally assume that they are linearly independent on Γ, though not necessarily mutually orthogonal. We also assume that, in the limit as $L \to \infty$, they form a basis in some appropriate function space defined on Γ (this will be touched on later).

When we substitute for ψ according to (3.4), the discretized boundary condition (3.8) takes the form

$$\sum_{\mu=1}^{M} U_{\alpha\mu}\theta_\mu - W_\alpha = 0, \tag{3.9}$$

where

$$U_{\alpha\mu} \equiv \int_\Gamma w_\alpha(s)\left(P(s)\phi_\mu(s) + Q(s)\frac{\partial\phi_\mu}{\partial n}(s)\right) d\Gamma \tag{3.10}$$

and

$$W_\alpha \equiv \int_\Gamma w_\alpha(s)R(s)\, d\Gamma. \tag{3.11}$$

The sets of constraints (3.9) may be incorporated into Φ via a set $\{\lambda_\alpha\}$ of Lagrange multipliers as follows:

$$\Phi = \tfrac{1}{2}\sum_{\mu}\sum_{v} S_{\mu v}\theta_\mu\theta_v - \sum_{\mu} T_\mu\theta_\mu - \sum_{\alpha}\lambda_\alpha\left(\sum_{\mu} U_{\alpha\mu}\theta_\mu - W_\alpha\right). \tag{3.12}$$

The problem of minimizing Φ is thus a *constrained* minimization problem, instead of an *unconstrained* one. We shall later find a way of undoing this, so that we have to solve only an unconstrained minimization problem. The necessary conditions for a

stationary Φ are derived from the composite Φ by differentiating with respect to θ_μ, with the result

$$\frac{\partial \Phi}{\partial \theta_\mu} = \sum_v S_{\mu v}\theta_v - T_\mu - \sum_\alpha \lambda_\alpha U_{\alpha\mu} = 0, \tag{3.13}$$

which, combined with (3.9), constitute the requisite number of equations to solve for the θ's and λ's.

We are now confronted with the task of formulating a reasonable and natural analogue of the continuous nullifier derived in the previous section. A fundamental point was made in the first paper on the cell method [8] which is pertinent to our present problem. It was emphasized that the classical limiting process of analysis maps an infinity of discrete formulations of a given problem into a single continuous one (when this is properly done, the discrete version is said to be *consistent*, see [9]). Unfortunately, the reverse mapping, i.e., back from continuous to discrete, is one-to-many, where the "many" are infinite in number. This state of affairs makes the construction of any discretization scheme ambiguous in the extreme, so that it is necessary to fall back on such heuristic criteria as "naturalness" and "reasonableness," to reduce the number of possibilities to manageable size.

We shall proceed by first restoring the continuous expressions which defined the discrete quantities in (3.13), and then try to relate them to the classical continuous variational procedure outlined in § 2. Substituting the quantities defined in (3.6), (3.7) and (3.10) into (3.13), we obtain

$$\begin{aligned}\int_\Omega \nabla\phi_\mu \cdot A \cdot \nabla\left(\sum_v \phi_v\theta_v\right) d\Omega + \int_\Omega b\phi_\mu\left(\sum_v \phi_v\theta_v\right) d\Omega \\ - \int_\Omega c\phi_\mu \, d\Omega - \int_\Gamma \left(\sum_\alpha \lambda_\alpha w_\alpha(s)\right) U[\phi_\mu] \, d\Gamma = 0.\end{aligned} \tag{3.14}$$

$U[\phi_\mu]$ is an abbreviation for the contents of the bracket in (3.10). We next apply integration by parts and Gauss' theorem to the first integral in (3.14) to obtain

$$\begin{aligned}&\int_\Omega \nabla\phi_\mu \cdot A \cdot \nabla\left(\sum_v \phi_v\theta_v\right) d\Omega \\ &\quad = \int_\Gamma \phi_\mu n \cdot A \cdot \nabla\left(\sum_v \phi_v\theta_v\right) d\Gamma - \int_\Omega \phi_\mu \nabla \cdot \left\{A \cdot \nabla\left(\sum_v \phi_v\theta_v\right)\right\} d\Omega.\end{aligned} \tag{3.15}$$

Before incorporating this result into (3.14), we shall make more substitutions so as to render the result more concise. We use (3.4) and the definition in the second line of (1.14), and then rearrange the factors to obtain

$$\begin{aligned}\int_\Omega \{-\nabla \cdot (A \cdot \nabla\psi) + b\psi - c\}\phi_\mu \, d\Omega + \int_\Gamma \frac{\partial \psi}{\partial n}\phi_\mu \, d\Gamma \\ - \int_\Gamma \left(\sum_\alpha \lambda_\alpha w_\alpha(s)\right) U[\phi_\mu] \, d\Gamma = 0.\end{aligned} \tag{3.16}$$

When we compare (3.16) with its continuous analogue (2.3), we see that the sum over α represents the *discretized form of* $\lambda(s)$. In other words, $\sum_\alpha \lambda_\alpha w_\alpha(s)$ is the expansion of $\lambda(s)$ in terms of the weight functions used to define the moment collocation of the boundary condition, viz., (3.8). In fact, (3.14) could have been derived by simple differentiation of an appropriate functional. This functional is a linear version of (2.2),

which has the form

$$\Phi[\psi] \equiv \int_\Omega \left(\frac{1}{2}\nabla\psi \cdot A \cdot \nabla\psi + \frac{1}{2}b\psi^2 - c\psi\right) d\Omega - \int_\Gamma \left(\sum_\alpha \lambda_\alpha w_\alpha(s)\right)(U[\psi] - R)\, d\Gamma. \tag{3.17}$$

We would derive (3.14) simply by differentiating this Φ with respect to θ_μ.

It is easy to recognize the expression in the brace in (3.16) as the analogue of the variational derivative of F in (3.1). In the latter equation, ϕ stood for the *variation* or, in modern terms, the *test function.* In (3.16), ϕ_μ stands for a basis function, of course, but it plays the role of a "variation" or test function, in the discrete context. Our next step in the continuous case, as in the argument following (1.15), would be to use the arbitrariness of ϕ to show that the variational derivative should vanish in Ω. However, in the discrete case, *the ϕ's are not arbitrary.* If the ϕ's are chosen a priori without special regard for their satisfying either the Euler equation or the boundary condition, then no linear combination of them can be selected which will support an argument that would force the variational derivative to vanish pointwise. If such a combination did not vanish identically inside Ω, it would not in general vanish identically on Γ, and conversely.

What can we then substitute for the classical Fundamental Lemma of the Calculus of Variations in the framework of cell discretization? To arrive at a "reasonable" and "natural" heuristic solution, we must make somewhat of a detour. In the following section, we shall try to establish a convincing formal analogy between the continuous and the discrete versions of the variational problem, so as to facilitate a natural choice for the discrete form of the boundary correction.

4. Some analogies and interpretations. Our help comes from the matrices V and Z, introduced in [4] for the purpose of "pretransforming" $\{\theta_\mu\}$. The crucial assumption was made there that U is of *full rank* (or "nondegenerate"), i.e., that its rows are linearly independent. If, on the contrary, U is *degenerate*, the *entire* apparatus of CD *fails.* Accordingly, we assume that U is *not* degenerate, in which case matrices $\{V_{\mu\beta}\}$ and $\{Z_{\mu\tau}\}$ can readily be found (see [4]) with the properties:

$$\sum_\mu U_{\alpha\mu} V_{\mu\beta} = \delta_{\alpha\beta}, \qquad \sum_\mu U_{\alpha\mu} Z_{\mu\tau} = 0. \tag{4.1}$$

The ranges of the indices are: $\alpha, \beta = 1, \cdots, L$, $\mu = 1, \cdots, M$ and $\tau = 1, \cdots, N$, where $N \equiv M - L$. These relations, in matrix form, appear as follows:

$$UV = I, \qquad UZ = 0. \tag{4.2}$$

V and Z are not unique; V is a right-generalized inverse of U, and Z is a collection of linearly independent right null-vectors of U. By introducing the vector variables $\{\sigma_\alpha\}$ and $\{\rho_\tau\}$, the vector $\{\theta_\mu\}$ was expressed in [4] as the linear combination:

$$\theta_\mu = \sum_\alpha V_{\mu\alpha}\sigma_\alpha + \sum_\tau Z_{\mu\tau}\rho_\tau. \tag{4.3}$$

Equation (3.9) then becomes

$$\begin{aligned} \sum_\mu U_{\alpha\mu}\theta_\mu &= \sum_\beta \sum_\mu U_{\alpha\mu} V_{\mu\beta}\sigma_\beta + \sum_\tau \sum_\mu U_{\alpha\mu} Z_{\mu\tau}\rho_\tau \\ &= \sum_\beta \delta_{\alpha\beta}\sigma_\beta + \sum_\tau 0 \cdot \rho_\tau \\ &= \sigma_\alpha = W_\alpha. \end{aligned} \tag{4.4}$$

This remarkable simplification suggests that the V and Z matrices might also play a significant role in the continuous form of the representation ψ. We must first, however, develop further the analogy between the CD formulation of the boundary value problem, and the classical continuous formulation.

First of all, the matrices U, V and Z, and the functions $\{\phi_\mu(x)\}$ and $\{w_\alpha(s)\}$ introduced in [4] do not form a "complete" complement of quantities for discussing this question. We must also introduce the matrix Y, of order $N \times M$, with the properties:

$$YV = 0, \qquad YZ = I. \tag{4.5}$$

These relations complete a set, the other half of which is (4.2). It is clear, because (4.2) and (4.5) can be collected in the form

$$\begin{pmatrix} U \\ Y \end{pmatrix}(V \quad Z) = \begin{pmatrix} I & 0 \\ 0 & I \end{pmatrix}, \tag{4.6}$$

that Y is the "other part" of the inverse of (V, Z). Reversing the order of the factors in (4.6) yields the result

$$VU + ZY = I, \tag{4.7}$$

and because

$$(VU)(ZY) = V(UZ)Y = 0, \qquad (ZY)(VU) = Z(YV)U = 0, \tag{4.8}$$

the products (VU) and (ZY) can be interpreted as *projection* operators whose sum is the identity.

There are also certain *functions* missing, and they must be added to the ϕ's and the w's, so as to enable a symmetric, complete relationship to be set up between the discrete and continuous forms. We shall denote these functions by $\{\varepsilon_\tau(x)\}$ where τ, as before, runs from 1 to N. It is important to recall that we have assumed that the $\phi(x)$'s were linearly independent in Ω, and that the $w(s)$'s were linearly independent on Γ. Likewise, we assume now that the $\varepsilon(x)$'s are linearly independent in Ω, but do not have any a priori relationship to the ϕ's. We can then find sets of "dual" functions $\{\chi_\mu(x)\}$, $\{\gamma_\alpha(s)\}$ and $\{\zeta_\tau(x)\}$ which are biorthogonal to the ϕ's, the w's and the ε's, respectively, so that we have

$$\begin{aligned} &\int_\Omega \chi_\mu(x)\phi_\nu(x)\, d\Omega = \delta_{\mu\nu}, \\ &\int_\Gamma w_\alpha(s)\gamma_\beta(s)\, d\Gamma = \delta_{\alpha\beta}, \\ &\int_\Omega \varepsilon_\sigma(x)\zeta_\tau(x)\, d\Omega = \delta_{\sigma\tau}. \end{aligned} \tag{4.9}$$

Rewriting the definition (3.10) with emphasis on the operator nature of U, we have

$$U_{\alpha\mu} \equiv \int_\Gamma w_\alpha(s)\, U[\phi_\mu(x)](s)\, d\Gamma, \tag{4.10}$$

which suggests that we can also consider the effect of U as that of an integration involving a kernel $U(s, x)$ as follows:

$$Uf \equiv U[f(x)](s) \equiv \int_\Omega U(s, x) f(x)\, d\Omega \tag{4.11}$$

so that (4.10) becomes

$$U_{\alpha\mu} \equiv \int_\Gamma \int_\Omega w_\alpha(s) U(s, x) \phi_\mu(x)\, d\Omega\, d\Gamma. \tag{4.12}$$

This form, in turn, suggests the "inverse" bilinear expression for the kernel $U(s, x)$ in terms of the matrix elements $U_{\alpha\mu}$

$$U(s, x) = \sum_\alpha \sum_\mu \gamma_\alpha(s) U_{\alpha\mu} \chi_\mu(x). \tag{4.13}$$

Similarly,

$$Y(y, x) = \sum_\tau \sum_\mu \zeta_\tau(y) Y_{\tau\mu} \chi_\mu(x) \tag{4.14}$$

and for the "pseudoinverse" operators,

$$V(x, s) = \sum_\mu \sum_\alpha \phi_\mu(x) V_{\mu\alpha} w_\alpha(s) \tag{4.15}$$

and

$$Z(x, y) = \sum_\mu \sum_\tau \phi_\mu(x) Z_{\mu\tau} \varepsilon_\tau(y). \tag{4.16}$$

Corresponding to the discrete equation which is the first line of (4.1), we form the composite kernel $UV(s, t)$, which we derive formally by applying $U(s, x)$ to $V(x, t)$:

$$\begin{aligned} UV(s, t) &\equiv \int_\Omega U(s, x) V(x, t)\, d\Omega \\ &= \int_\Omega \sum_\alpha \sum_\mu \sum_\nu \sum_\beta \gamma_\alpha(s) U_{\alpha\mu} \chi_\mu(x) \phi_\nu(x) V_{\nu\beta} w_\beta(t)\, d\Omega \\ &= \sum_\alpha \sum_\mu \sum_\nu \sum_\beta \gamma_\alpha(s) U_{\alpha\mu} \delta_{\mu\nu} V_{\nu\beta} w_\beta(t) \\ &= \sum_\alpha \sum_\mu \sum_\beta \gamma_\alpha(s) U_{\alpha\mu} V_{\mu\beta} w_\beta(t) \\ &= \sum_\alpha \sum_\beta \gamma_\alpha(s) \delta_{\alpha\beta} w_\beta(t) \\ &= \sum_\alpha \gamma_\alpha(s) w_\alpha(t) \\ &= \Delta(s, t). \end{aligned} \tag{4.17}$$

We have made use of (4.9), (4.1). The notation $\Delta(s, t)$ indicates that the final sum is a truncated version of a formal series expansion for the delta-function, which follows, also formally, from the biorthogonality of the γ's and w's. Along similar lines, one can derive

$$\begin{aligned} UZ(s, y) &\equiv \int_\Omega U(s, x) Z(x, y)\, d\Omega = 0, \\ YV(y, s) &\equiv \int_\Omega Y(y, x) V(x, s)\, d\Omega = 0, \\ YZ(y, z) &\equiv \int_\Omega Y(y, x) Z(x, z)\, d\Omega = \Delta(y, z). \end{aligned} \tag{4.18}$$

Thus, the operator kernels U, Y, V and Z obey the same formal relations as their matrix counterparts in (4.6).

V is evidently a right-hand inverse of U, and Z a right-hand inverse of Y. We recall the fact that U is a *trace* operator, and thereby acts as a *projection* of a function $\psi(x)$, defined in Ω, into another function, defined only on Γ. We can therefore regard $V(x, s)$ as an *extension* operator, which imbeds functions defined on Γ into the set of those defined over Ω. The corresponding interpretations of Y and Z will be taken up in the next section.

It will be necessary to maintain a clear distinction between the boundary condition operator U and the more direct "restriction" trace operator E, which has the effect

$$Ef(x) \equiv f(s), \tag{4.19}$$

so that its matrix elements are

$$E_{\alpha\mu} = \int_\Gamma w_\alpha(s) E\phi_\mu(x)\, d\Gamma = \int_\Gamma w_\alpha(s)\phi_\mu(s)\, d\Gamma. \tag{4.20}$$

E is the trace operator which maps $f(x)$, defined in Ω, into the set of its limiting values on Γ; we are using the word "restriction" in this sense. We shall find E useful in helping to interpret the relationship between the continuous and discrete versions of the induced boundary condition. Its bilinear representation is similar to that of $U(s, x)$:

$$E(s, x) = \sum_\alpha \sum_\mu \gamma_\alpha(s) E_{\alpha\mu} \chi_\mu(x) \tag{4.21}$$

and it has a corresponding extension kernel $F(x, s)$

$$F(x, s) = \sum_\mu \sum_\alpha \phi_\mu(x) F_{\mu\alpha} w_\alpha(s) \tag{4.22}$$

with

$$\sum_\mu E_{\alpha\mu} F_{\mu\beta} = \delta_{\alpha\beta}. \tag{4.23}$$

5. The analogue of the classical variation. To complete our complement of special functions, we need add only two more. These are defined as follows:

$$\xi_\alpha(x) \equiv \sum_\mu \phi_\mu(x) V_{\mu\alpha}, \qquad \eta_\tau(x) \equiv \sum_\mu \phi_\mu(x) Z_{\mu\tau} \tag{5.1}$$

where α, τ and μ have the same ranges as before. These functions have the interesting properties

$$\begin{aligned} U\xi_\alpha(x) &= \int_\Omega U(s, x)\xi_\alpha(x)\, d\Omega \\ &= \int_\Omega \sum_\beta \sum_\mu \sum_\nu \gamma_\beta(s) U_{\beta\mu}\chi_\mu(x)\phi_\nu(x) V_{\nu\alpha}\, d\Omega \\ &= \sum_\beta \sum_\mu \sum_\nu \gamma_\beta(s) U_{\beta\mu}\delta_{\mu\nu} V_{\nu\alpha} \\ &= \sum_\beta \sum_\mu \gamma_\beta(s) U_{\beta\mu} V_{\mu\alpha} \\ &= \sum_\beta \gamma_\beta(s)\delta_{\beta\alpha} \\ &= \gamma_\alpha(s) \end{aligned} \tag{5.2}$$

and

$$
\begin{aligned}
U\eta_\tau(x) &= \int_\Omega U(s,x)\eta_\tau(x)\,d\Omega \\
&= \int_\Omega \sum_\beta \sum_\mu \sum_\nu \gamma_\beta(s) U_{\beta\mu}\chi_\mu(x)\phi_\nu(x)Z_{\nu\tau}\,d\Omega \\
&= \sum_\beta \sum_\mu \sum_\nu \gamma_\beta(s) U_{\beta\mu}\delta_{\mu\nu}Z_{\nu\tau} \\
&= \sum_\beta \sum_\mu \gamma_\beta(s) U_{\beta\mu}Z_{\mu\tau} \\
&= \sum_\beta \gamma_\beta(s)\cdot 0 \\
&= 0.
\end{aligned} \tag{5.3}
$$

Similarly, it may be shown that

$$
Y\xi_\alpha(x) = 0, \qquad Y\eta_\tau(x) = \zeta_\tau(y). \tag{5.4}
$$

It also follows from (4.17) and (4.18) (or it can be derived as above) that

$$
\xi_\alpha(x) = V\gamma_\alpha(s), \qquad \eta_\tau(x) = Z\zeta_\tau(y). \tag{5.5}
$$

We now have everything we need to interpret these relationships. Let us first suppose that U were simply the restriction operator E. In that case, (5.2) would have the meaning that $\gamma_\alpha(s)$ is the restriction of $\xi_\alpha(x)$ on Γ. But more interesting is the interpretation of (5.3): namely, that every $\eta_\tau(x)$ *vanishes* on Γ. This is what corresponds to selecting a variation, in the continuous context, which vanishes on the boundary, but not in the interior of Ω. The use of such a variation is implicit in the argument leading to (1.16). In the discrete case before us, we are not able to select an *arbitrary* variation, but only those which are linear combinations of the η's. We may, however, think of η_τ as "vanishing on Γ", even when the trace operator U has the more general form (3.2), instead of simply being E.

The "dual" of U is Y, in the following sense: Let us consider the set of functions into which Z maps all functions defined in Ω. The result of applying the trace operator U to any of these functions will yield zero, viz., the function which is identically zero on Γ. The reason for this is, of course, that $UZ \equiv 0$, as shown in (4.18). Thus, we may regard the "Z-extension" of any function as one with a null trace on Γ. In a similar sense, the "V-extension" of any function defined on Γ has vanishing Y-projection, or "Y-trace" in Ω, because of the fact that $YV \equiv 0$. In this sense, any function which is a V-extension "vanishes in Ω, but not on Γ" (i.e., its support is Γ). This appears to be the closest we can come, in the discrete context, to the classical variations whose supports can be specified arbitrarily, and which were used to derive (1.17).

In order to derive the variational equations corresponding to these two classes of discrete variations (viz., with support in Ω and on Γ, respectively), we need only postmultiply (3.16) by $Z_{\mu\tau}$ and $V_{\mu\alpha}$, in turn, and sum over μ to obtain

$$
\begin{aligned}
&\int_\Omega \{-\nabla\cdot(A\cdot\nabla\psi) + b\psi - c\}\eta_\tau\,d\Omega + \int_\Gamma \frac{\partial\psi}{\partial n}\eta_\tau\,d\Gamma \\
&\quad - \int_\Gamma \left(\sum_\beta \lambda_\beta w_\beta(s)\right) U\eta_\tau\,d\Gamma = 0
\end{aligned} \tag{5.6}
$$

and

$$\int_\Omega \{-\nabla\cdot(A\cdot\nabla\psi)+b\psi-c\}\xi_\alpha\,d\Omega+\int_\Gamma \frac{\partial\psi}{\partial n}\,\xi_\alpha\,d\Gamma - \int_\Gamma \left(\sum_\beta \lambda_\beta w_\beta(s)\right) U\xi_\alpha\,d\Gamma = 0. \tag{5.7}$$

With the help of (5.3), (5.2) and (4.9), we reduce these to

$$\int_\Omega \{-\nabla\cdot(A\cdot\nabla\psi)+b\psi-c\}\eta_\tau\,d\Omega+\int_\Gamma \frac{\partial\psi}{\partial n}\,\eta_\tau\,d\Gamma=0 \tag{5.8}$$

and

$$\int_\Omega \{-\nabla\cdot(A\cdot\nabla\psi)+b\psi-c\}\xi_\alpha\,d\Omega+\int_\Gamma \frac{\partial\psi}{\partial n}\,\xi_\alpha\,d\Gamma=\lambda_\alpha. \tag{5.9}$$

We emphasize again the fact that the set of variations (or test functions) is only a finite set (i.e., linear combinations of $\{\phi_\mu(x)\}$), so that the arguments necessary to derive (1.16) and both equations (2.5) cannot go through. We must therefore *identify* (5.8) and (5.9) with the equations (2.4), (2.7) and (2.8), in some manner. The most reasonable choice is to identify (5.8) with both (2.4) and (2.7), and (5.9) with (2.8). The finiteness of the set of possible discrete variations prevents us from disentangling the volume and surface terms in (5.8). In fact, we shall look to the nullifier which must be added to Φ, to cancel some of the surface components, and thereby to force the Euler equation to be satisfied (in the discrete sense). Thus, we shall adopt the position that the sets $\{\eta_\tau(x)\}$ and $\{\xi_\alpha(x)\}$ correspond to the classical continuous variations from which the Euler equation and the induced boundary condition are respectively derived. With this identification, it also appears that Z corresponds to the null vector L in (2.6), and V corresponds to the "other" vector leading to (2.8). These various correspondences place us in a position to proceed further with the construction of a discrete nullifier.

Equations (5.8) and (5.9) can be derived in a more direct manner by using a more convenient representation of $\psi(x)$. Using (3.4), (4.3) and (5.1), we can obtain

$$\begin{aligned}
\psi(x,\theta) &\equiv \sum_\mu \phi_\mu(x)\theta_\mu \\
&= \sum_\mu \phi_\mu(x)\left(\sum_\alpha V_{\mu\alpha}\sigma_\alpha+\sum_\tau Z_{\mu\tau}\rho_\tau\right) \\
&= \sum_\alpha\left(\sum_\mu \phi_\mu(x)V_{\mu\alpha}\right)\sigma_\alpha+\sum_\tau\left(\sum_\mu \phi_\mu(x)Z_{\mu\tau}\right)\rho_\tau \\
&= \sum_\alpha \xi_\alpha(x)\sigma_\alpha+\sum_\tau \eta_\tau(x)\rho_\tau \\
&\equiv \psi(x;\sigma,\rho),
\end{aligned} \tag{5.10}$$

which is just another parametrization of ψ in terms of σ and ρ. If we now substitute this form of ψ into (3.17) and differentiate with respect to ρ_τ, we obtain (5.8). This observation makes the derivation of the discrete boundary nullifier more direct, as we shall see in the next section. We also note that if we differentiate the same expression for Φ with respect to σ, we obtain (5.9). These results are not hard to obtain, given the apparatus already set up.

6. The discrete nullifier. In the discrete context, there seems to be nothing that corresponds to the independent variation of ϕ and $\partial\phi/\partial n$, such as appears in (2.10). It is therefore not clear how the variation of a nullifier N, considered as a function of ϕ and $\partial\phi/\partial n$, should be handled. We are therefore forced to fall back on the approach of dealing directly with the discrete form of N, considered as a function of σ and ρ.

The functional Φ, as defined in (3.17), depends on ψ and various of its derivatives. By substituting for ψ according to (4.3), Φ becomes a function of the σ's and ρ's. The result of differentiating this function with respect to ρ_τ is given by the left member of (5.8) (as explained in the previous section). The ρ_τ-derivative of the composite functional, including N, is thus given by

$$\int_\Omega \{-\nabla\cdot(A\cdot\nabla\psi)+b\psi-c\}\eta_\tau\,d\Omega+\int_\Gamma \frac{\partial\psi}{\partial n}\eta_\tau\,d\Gamma+\frac{\partial N}{\partial\rho_\tau}=0. \tag{6.1}$$

Our strategy, as indicated in the previous section, is to nullify the surface integral with N, so as to make the second line in (6.1) vanish, insofar as possible. Thus, if we set

$$\frac{\partial N}{\partial\rho_\tau}=-\int_\Gamma \frac{\partial\psi}{\partial n}\eta_\tau\,d\Gamma \tag{6.2}$$

we are left with the "projected" Euler equation corresponding to the η_τ in question.

Before proceeding further, however, we must reinterpret, in the discrete context, the role of the boundary correction. We imposed the vanishing of *certain weighted integrals* (the moments) of the boundary condition, as shown in (3.8). The induced (natural) boundary condition is presumed to clash, i.e., be inconsistent, with the imposed condition. Therefore, we do not wish to nullify the *entire* induced condition ($\partial\psi/\partial n$ in the linear case) as was done in [3], but only *that part corresponding to the imposed moments.*

We have a clue as to how to identify this, in that the $\lambda(s)$ in the continuous formulation (2.9) was replaced by the finite expansion $\sum_\alpha \lambda_\alpha w_\alpha(s)$. We can interpret this as meaning that λ was "projected" into the subspace spanned by $\{w_\alpha(s)\}$. We shall therefore take the position that any function that occupies a corresponding place in an integral should be similarly projected.

It will be more concise, in what follows, to use a bracket notation to represent the integral of a product of functions on Γ. Thus

$$\langle F, G\rangle\equiv\int_\Gamma F(s)G(s)\,d\Gamma. \tag{6.3}$$

Our first step in reducing the right member of (6.2) is to expand ψ and η in terms of the ϕ's. Referring to (3.4) and (5.1), we obtain

$$\left\langle\frac{\partial\psi}{\partial n},\eta_\tau\right\rangle=\sum_\mu\sum_\nu\theta_\mu\left\langle\frac{\partial\phi_\mu}{\partial n},\phi_\nu\right\rangle Z_{\nu\tau}, \tag{6.4}$$

so that we now have to consider the projection of $\partial\phi_\mu/\partial n$. As indicated previously, we expand $\partial\phi_\mu/\partial n$ as a linear combination of the w's, thereby obtaining an approximation to it,

$$\frac{\partial\phi_\mu}{\partial n}\sim\sum_\alpha G_{\mu\alpha}w_\alpha(s); \tag{6.5}$$

and, in order to evaluate the G's, we multiply by $\gamma_\beta(s)$ and integrate over Γ to obtain

$$\begin{aligned}\left\langle \frac{\partial \phi_\mu}{\partial n}, \gamma_\beta \right\rangle &= \left\langle \sum_\alpha G_{\mu\alpha} w_\alpha, \gamma_\beta \right\rangle \\ &= \sum_\alpha G_{\mu\alpha} \langle w_\alpha, \gamma_\beta \rangle \\ &= \sum_\alpha G_{\mu\alpha}\, \delta_{\alpha\beta} \\ &= G_{\mu\beta}\end{aligned} \tag{6.6}$$

with the help of (4.9). Thus, we get for $\partial\phi_\mu/\partial n$,

$$\begin{aligned}\left(\frac{\partial \phi_\mu}{\partial n}\right)(s) &\sim \sum_\alpha \left\langle \frac{\partial \phi_\mu}{\partial n}, \gamma_\alpha \right\rangle w_\alpha(s) \\ &= \int_\Gamma \left(\frac{\partial \phi_\mu}{\partial n}\right)(t) \sum_\alpha \gamma_\alpha(t) w_\alpha(s)\, d\Gamma_t \\ &= \int_\Gamma \left(\frac{\partial \phi_\mu}{\partial n}\right)(t)\, \Delta(t, s)\, d\Gamma_t.\end{aligned} \tag{6.7}$$

This equation has a form which suggests that the function $\Delta(s, t)$ can be thought of as the kernel of a projection operator Δ.

Next, we substitute from (6.7) into (6.4), and obtain

$$\begin{aligned}\left\langle \frac{\partial \psi}{\partial n}, \eta_\tau \right\rangle &\sim \sum_\mu \sum_v \sum_\alpha \theta_\mu G_{\mu\alpha} \left\langle w_\alpha(s), \phi_v \right\rangle Z_{v\tau} \\ &= \sum_\mu \sum_v \sum_\alpha \theta_\mu G_{\mu\alpha} E_{\alpha v} Z_{v\tau} \\ &\equiv \sum_\mu \sum_v \theta_\mu H_{\mu v} Z_{v\tau},\end{aligned} \tag{6.8}$$

where we have made use of the definition (4.20). The set of quantities $\{H_{\mu v}\}$ is given by

$$\begin{aligned}H_{\mu v} &\equiv \sum_\alpha \left\langle \frac{\partial \phi_\mu}{\partial n}, \gamma_\alpha \right\rangle \left\langle w_\alpha, \phi_v \right\rangle \\ &= \int_\Gamma \int_\Gamma \frac{\partial \phi_\mu}{\partial n}(t)\, \Delta(t, s) \phi_v(s)\, d\Gamma_t\, d\Gamma_s.\end{aligned} \tag{6.9}$$

This is in contrast to the corresponding "unprojected" formula derived in [3], which (implicitly) reads

$$H^u_{\mu v} \equiv \int_\Gamma \frac{\partial \phi_\mu}{\partial n}(s) \phi_v(s)\, d\Gamma_s. \tag{6.10}$$

Recalling that $\alpha = 1, \cdots, L$, it is clear from the form of (6.9) that the rank of $H_{\mu v}$ is at most L. On the other hand, $H^u_{\mu v}$ can clearly have a rank of as much as M, since μ and v range from 1 to M. Hence, we have already succeeded in reducing the rank of the nullifier, inasmuch as it depends on $H_{\mu v}$. We shall see later that we can sometimes reduce the rank of $N(\sigma, \rho)$ still further.

We next substitute for θ_μ in (6.8), according to (4.3), to obtain (note that the *variable* σ and the *index* σ are not related)

$$\begin{aligned}\left\langle \frac{\partial \psi}{\partial n}, \eta_\tau \right\rangle &\sim -\sum_\mu \sum_\nu \left(\sum_\alpha V_{\mu\alpha}\sigma_\alpha + \sum_\tau Z_{\mu\tau}\rho_\tau \right) H_{\mu\nu} Z_{\nu\tau} \\ &= \sum_\alpha \sigma_\alpha \left(\sum_\mu \sum_\nu V_{\mu\alpha} H_{\mu\nu} Z_{\nu\tau} \right) + \sum_\sigma \rho_\sigma \left(\sum_\mu \sum_\nu Z_{\mu\sigma} H_{\mu\nu} Z_{\nu\tau} \right) \\ &\equiv \sum_\alpha \sigma_\alpha B_{\alpha\tau} + \sum_\sigma \rho_\sigma C_{\sigma\tau},\end{aligned} \tag{6.11}$$

so that B and C are given by

$$B_{\alpha\tau} \equiv \sum_\mu \sum_\nu V_{\mu\alpha} H_{\mu\nu} Z_{\nu\tau}, \qquad C_{\sigma\tau} \equiv \sum_\mu \sum_\nu Z_{\mu\sigma} H_{\mu\nu} Z_{\nu\tau}. \tag{6.12}$$

Equation (6.2) now reduces to

$$\frac{\partial N}{\partial \rho_\tau} = -\left(\sum_\alpha \sigma_\alpha B_{\alpha\tau} + \sum_\sigma \rho_\sigma C_{\sigma\tau} \right). \tag{6.13}$$

This system of equations can be integrated *only* if $C_{\sigma\tau}$ is *symmetric*, because the manifestly symmetric cross-partial $\partial^2 N/\partial\rho_\sigma \partial\rho_\tau$ is just equal to $C_{\sigma\tau}$. From the definition (6.12) it is evident that the symmetry of $C_{\sigma\tau}$ depends on that of $H_{\mu\nu}$. Unfortunately, the latter is not a priori symmetric, so it appears that we can only nullify that part of the induced boundary condition involving the *symmetric part* of $C_{\sigma\tau}$. We shall therefore replace $C_{\sigma\tau}$ by $C^S_{\sigma\tau}$ in (6.13), where $C^S_{\sigma\tau}$ is defined by

$$\begin{aligned}C^S_{\sigma\tau} &\equiv \tfrac{1}{2}(C_{\sigma\tau} + C_{\tau\sigma}) \\ &= \tfrac{1}{2} \sum_\mu \sum_\nu Z_{\mu\sigma}(H_{\mu\nu} + H_{\nu\mu}) Z_{\nu\tau} \\ &= \sum_\mu \sum_\nu Z_{\mu\sigma} H^S_{\mu\nu} Z_{\nu\tau}.\end{aligned} \tag{6.14}$$

This forced symmetrization of $C_{\sigma\tau}$ represents a setback in our program of nullifying induced boundary constraints in the discrete case. We shall return to an examination of the effect of this symmetrization later, and try to ascertain the extent of the damage.

The altered equation corresponding to (6.13), viz,

$$\frac{\partial N}{\partial \rho_\tau} = -\left(\sum_\alpha \sigma_\alpha B_{\alpha\tau} + \sum_\sigma \rho_\sigma C^S_{\sigma\tau} \right), \tag{6.15}$$

may now be solved to yield

$$N(\sigma, \rho) = -\sum_\alpha \sum_\tau \sigma_\alpha B_{\alpha\tau} \rho_\tau - \tfrac{1}{2} \sum_\sigma \sum_\tau \rho_\sigma C^S_{\sigma\tau} \rho_\tau + \Lambda(\sigma), \tag{6.16}$$

where $\Lambda(\sigma)$ is an arbitrary function of the σ's, corresponding to the arbitrary function in (2.14). As in the continuous case, the most reasonable choice for Λ is a quadratic function of the σ's. If we denote the symmetric matrix of coefficients also by Λ, (6.16) becomes

$$N(\sigma, \rho) = -\sum_\alpha \sum_\tau \sigma_\alpha B_{\alpha\tau} \rho_\tau - \tfrac{1}{2} \sum_\sigma \sum_\tau \rho_\sigma C^S_{\sigma\tau} \rho_\tau + \tfrac{1}{2} \sum_\alpha \sum_\beta \sigma_\alpha \Lambda_{\alpha\beta} \sigma_\beta. \tag{6.17}$$

Our next and final step is to rewrite $N(\sigma, \rho)$ in terms of the original variables $\{\theta_\mu\}$. This can be easily done because

$$\begin{aligned}\sum_\mu U_{\beta\mu}\theta_\mu &= \sum_\mu \sum_\alpha U_{\beta\mu}V_{\mu\alpha}\sigma_\alpha + \sum_\mu \sum_\tau U_{\beta\mu}Z_{\mu\tau}\rho_\tau \\ &= \sum_\alpha \delta_{\beta\alpha}\sigma_\alpha + \sum_\tau 0\cdot\rho_\tau \qquad (6.18)\\ &= \sigma_\beta\end{aligned}$$

and

$$\begin{aligned}\sum_\mu Y_{\sigma\mu}\theta_\mu &= \sum_\mu \sum_\alpha Y_{\sigma\mu}V_{\mu\alpha}\sigma_\alpha + \sum_\mu \sum_\tau Y_{\sigma\mu}Z_{\mu\tau}\rho_\tau \\ &= \sum_\alpha 0\cdot\sigma_\alpha + \sum_\tau \delta_{\sigma\tau}\rho_\tau \qquad (6.19)\\ &= \rho_\sigma,\end{aligned}$$

so that $N(\theta)$ can be written

$$N(\theta) = \sum_\mu \sum_v \theta_\mu \left(-\sum_\alpha \sum_\tau U_{\alpha\mu}B_{\alpha\tau}Y_{\tau v} - \frac{1}{2}\sum_\sigma \sum_\tau Y_{\sigma\mu}C^S_{\sigma\tau}Y_{\tau v} + \frac{1}{2}\sum_\alpha \sum_\beta U_{\alpha\mu}\Lambda_{\alpha\beta}U_{\beta v}\right)\theta_v. \qquad (6.20)$$

With obvious identifications, $N(\theta)$ may be written in matrix form as follows:

$$\begin{aligned}N(\theta) &= \theta^T\{-U^TBY - \tfrac{1}{2}Y^TC^SY + \tfrac{1}{2}U^T\Lambda U\}\theta \\ &= \theta^T\{-\tfrac{1}{2}(U^TBY + Y^TB^TU) - \tfrac{1}{2}Y^TC^SY + \tfrac{1}{2}U^T\Lambda U\}\theta \\ &= -\tfrac{1}{2}\theta^T\{U^TV^THZY + Y^TZ^TH^TVU + Y^TZ^TH^SZY - U^T\Lambda U\}\theta \qquad (6.21)\\ &= -\tfrac{1}{2}\theta^T\{(VU)^TH(ZY) + (ZY)^TH^T(VU) \\ &\qquad + (ZY)^TH^S(ZY) - U^T\Lambda U\}\theta.\end{aligned}$$

The second line of (6.21) follows from the fact that any matrix of coefficients of a quadratic form is automatically symmetrized, the third line is a result of substituting for B and C from (6.12), and the last line is merely a regrouping of the third.

This expression may in turn be transformed to another interesting form by recalling from (4.7) that $VU + ZY = I$ and also by making use of the decomposition of H into the sum of its symmetric and antisymmetric parts: $H = H^S + H^A$. After some routine algebra, $N(\theta)$ can be written

$$N(\theta) = -\tfrac{1}{2}\theta^T(H^AZY - Y^TZ^TH^A + H^S - (VU)^TH^S(VU) - U^T\Lambda U)\theta. \qquad (6.22)$$

Since U spans only a subspace of R^M, it is clear that only that part of H lying in the U-subspace can be altered by any choice of Λ. Thus, any negative eigenvalues of H associated with eigenvectors in the Y-subspace cannot be removed, so that the nullifier $N(\sigma, \rho)$ may, in some cases, degrade the positivity of the functional Φ. However, Λ may still be chosen so as to adjust those eigenvalues of H associated with the U-subspace.

By the "interior projection" of $(\partial\phi_\mu/\partial n)(s)\phi_v(s)$ by $\Delta(t, s)$ to form $H_{\mu v}$, as exhibited in (6.9), we reduced the rank of H from M to L. We can reduce it still further by taking cognizance of the fact that we only wish to nullify that part of the induced boundary condition that conflicts with *certain moments* of a certain imposed boundary condition. In the discrete context, the imposition of more boundary constraints than one is not forbidden, so that if the "primary" constraints were of Dirichlet

type, involving moments of only the values of ψ, it would still be permissible to impose other "secondary" constraints involving, for example, the normal derivative of ψ. It is only the moments of the constraints induced by the latter that we would want to nullify.

For most of the cell boundary segments, the unwanted induced conditions would be associated with imposed *interface* conditions associated with the presence of other contiguous cells. Each of these would be on the other side of some boundary segment of the original cell. It is possible to characterize this situation simply by *restricting the supports of the weight functions* to one or another of the segments of the cell boundary. As the index α ran over its range, one after another of the boundary segments will be the support of the weight functions. Accordingly, the number L is then the sum of the individual collocation counts for all the boundary segments of the cell, whether they are interfaces or parts of the true boundary Γ.

In any case, we may assume that the index α in (6.9) associated with "nullifiable" induced conditions ranges over a smaller number of values than the total L of all collocations. Let $\alpha' = \{\alpha_1, a_2, \cdots, \alpha_{L'}\}$, where $L' < L$. Then (6.9) becomes

$$\begin{aligned} H'_{\mu\nu} &\equiv \sum_{\alpha'} \left\langle \frac{\partial \phi_\mu}{\partial n}, \gamma_{\alpha'} \right\rangle \left\langle w_{\alpha'}, \phi_\nu \right\rangle \\ &= \int_\Gamma \int_\Gamma \frac{\partial \phi_\mu}{\partial n}(t) \Delta'(t, s) \phi_\nu(s)\, d\Gamma_t\, d\Gamma_s \end{aligned} \tag{6.23}$$

and this H' generates the nullifier of minimum rank.

7. More on the symmetrization discrepancy. We return now to examine the effect of the symmetrization of $C_{\sigma\tau}$. The difference between the correct $C_{\sigma\tau}$ in (6.13) and the symmetrized $C^S_{\sigma\tau}$ in (6.15) is just the antisymmetric matrix $C^A_{\sigma\tau}$, which is given by

$$\begin{aligned} C^A_{\sigma\tau} &\equiv \sum_\mu \sum_\nu Z_{\mu\sigma} H^A_{\mu\nu} Z_{\nu\tau} \\ &= \frac{1}{2} \sum_\mu \sum_\nu Z_{\mu\sigma} \left\{ \sum_\alpha \left\langle \frac{\partial \phi_\mu}{\partial n}, \gamma_\alpha \right\rangle \left\langle w_\alpha, \phi_\nu \right\rangle - \sum_\alpha \left\langle \frac{\partial \phi_\nu}{\partial n}, \gamma_\alpha \right\rangle \left\langle w_\alpha, \phi_\mu \right\rangle \right\} Z_{\nu\tau}. \end{aligned} \tag{7.1}$$

Taking note of the definition (5.1) of the η's, we have

$$C^A_{\sigma\tau} = \frac{1}{2} \left\{ \sum_\alpha \left\langle \frac{\partial \eta_\sigma}{\partial n}, \gamma_\alpha \right\rangle \langle w_\alpha, \eta_\tau \rangle - \sum_\alpha \left\langle \frac{\partial \eta_\tau}{\partial n}, \gamma_\alpha \right\rangle \langle w_\alpha, \eta_\sigma \rangle \right\}. \tag{7.2}$$

Equations (6.9) and (6.10) suggest that the expression for C^A is related to the difference of two boundary integrals which appears in the well-known Green's theorem. The generic form of this which we shall use is

$$\begin{aligned} &\int_\Gamma \left(\frac{\partial J}{\partial n} \right)(s) K(s)\, d\Gamma - \int_\Gamma \left(\frac{\partial K}{\partial n} \right)(s) J(s)\, d\Gamma \\ &\quad = \int_\Omega (\nabla \cdot A \cdot \nabla J)(x) K(x)\, d\Omega - \int_\Omega (\nabla \cdot A \cdot \nabla K)(x) J(x)\, d\Omega. \end{aligned} \tag{7.3}$$

We shall try to construct a reasonable cognate form of this in the discrete context. The first step is to transform a sum of bracket products like those in (7.2) into integral

form. We have

$$\sum_{\alpha}\left\langle\frac{\partial\eta_\sigma}{\partial n},\gamma_\alpha\right\rangle\langle w_\alpha,\eta_\tau\rangle=\int_\Gamma\int_\Gamma\left(\frac{\partial\eta_\sigma}{\partial n}\right)(s)\,\Delta(s,t)\,\eta_\tau(t)\,d\Gamma_t\,d\Gamma_s$$
$$=\int_\Gamma\left(\frac{\partial\eta_\sigma}{\partial n}\right)(s)(\Delta\eta_\tau)(s)\,d\Gamma_s. \tag{7.4}$$

We have used the definition of $\Delta(s,t)$ in (4.17), and have interpreted Δ in the last line as an operator. In order to be able to relate this integral over Γ to one over Ω, we must find a function of x in Ω whose "restriction," or E-trace, is $(\Delta\eta_\tau)(s)$. We can construct such a function by using the extension operator F whose kernel was defined in (4.22). Thus, we define

$$\begin{aligned}(F\Delta\eta_\tau)(x)&\equiv\int_\Gamma\int_\Gamma F(x,s)\,\Delta(s,t)\,\eta_\tau(t)\,d\Gamma_t\,d\Gamma_s\\&=\int_\Gamma\int_\Gamma\int_\Omega F(x,s)\,\Delta(s,t)\,E(t,y)\eta_\tau(y)\,d\Omega\,d\Gamma_t\,d\Gamma_s\\&=F\,\Delta E\,\eta_\tau(x)\\&\equiv P\eta_\tau(x).\end{aligned} \tag{7.5}$$

The second line reflects the fact that $\eta_\tau(s)$ is just the E-trace of $\eta_\tau(x)$. The last line defines the "inner" projection $P(\equiv F\,\Delta E)$, which, operating on $\eta_\tau(x)$, returns an "attenuated" $\eta_\tau(x)$, which reflects the reduction in rank of $\Delta(s,t)$ in the discrete case.

If we now identify, in (7.3), $K(s)$ with $\Delta\eta_\tau$, $J(s)$ with ΔH_σ, $K(x)$ with $P\eta_\tau(x)$, $J(x)$ with $P\eta_\sigma(x)$, and otherwise J and K with η_σ and η_τ, respectively, then the following special form of Green's theorem is valid:

$$\int_\Gamma\left(\frac{\partial\eta_\sigma}{\partial n}\right)(\Delta\eta_\tau)\,d\Gamma-\int_\Gamma\left(\frac{\partial\eta_\tau}{\partial n}\right)(\Delta\eta_\sigma)\,d\Gamma$$
$$=\int_\Omega(\nabla\cdot A\cdot\nabla\eta_\sigma)(P\eta_\tau)-\int_\Omega(\nabla\cdot A\cdot\nabla\eta_\tau)(P\eta_\sigma)\,d\Omega. \tag{7.6}$$

This is the "exact discrete Green's Theorem." Taking into account (7.2), (7.4) and (7.6), it results that an alternative form for C^A is

$$C^A_{\sigma\tau}=\frac{1}{2}\left\{\int_\Omega(\nabla\cdot A\cdot\nabla\eta_\sigma)P\eta_\tau(x)\,d\Omega-\int_\Omega(\nabla\cdot A\cdot\nabla\eta_\tau)P\eta_\sigma(x)\,d\Omega\right\} \tag{7.7}$$

in terms of integrals over Ω.

This residual induced boundary condition appears to be irreducible in the discrete case. The flexibility of the variation in the continuous case would allow us to set the trace of η to zero on Γ, thus eliminating C^A. If, in fact, the trace operator U is the same as the restriction operator E, C^A vanishes because $U\eta_\sigma$ vanishes for all U's, so that $E\eta_\sigma$ will vanish identically. Otherwise, it would appear that it would be necessary to select a basis set for each of whose members the generalized Laplacian (in the brackets) would vanish. This is reminiscent of the Trefftz strategy for the solution of partial differential equations (see [10]). The lack of generality in placing such stringent a priori constraints on the η's makes this approach seem rather unpromising.

A more promising, algebraic approach to reducing the symmetrization discrepancy relies on the fact that there is a residue of arbitrariness in the definitions of Z, Y and

the vector functions $\varepsilon(y) \equiv \{\varepsilon_\sigma(y)\}$ and $\zeta(y) \equiv \{\zeta_\sigma(y)\}$ introduced in § 4. Let $Q \equiv \{Q_{\sigma\tau}\}$ be an $N \times N$ nonsingular matrix, and transform Z, Y, ρ, $\varepsilon(y)$ and $\zeta(y)$ as follows:

$$\begin{aligned} &Z \to ZQ, \\ &Y \to Q^{-1}Y, \\ &\rho \to Q^{-1}\rho, \\ &\varepsilon(y) \to Q^{-1}\varepsilon(y), \\ &\zeta(y) \to Q\zeta(y). \end{aligned} \tag{7.8}$$

Then all the relations (4.3), (4.4), (4.6), (4.7), (4.8), (4.9), (4.14) and (4.16) are unchanged. However, C^A is transformed as follows:

$$C^A \equiv Z^T H^A Z \to Q^T Z^T H^A Z Q = Q^T C^A Q. \tag{7.9}$$

The rank of Z is N, so that no column of the $N \times N$ Q-matrix can be a right null-vector of Z. However, as indicated at the end of § 6, the rank of H is L' and that of H^A may easily be less. In case H is of odd order, this will often be the case, because one of the eigenvalues of H^A will have to be both real and imaginary, so that it can only be zero. If the corresponding eigenvalue of H did not vanish, the rank of H^A would be less. In the important case of H being rank-one, which would be the case if the imposed boundary condition were an overall conservation condition applied to an entire segment of Γ, thereby involving only one moment, H^A would vanish identically.

In any case, it may easily happen that $L' < N$, so that although the Q-matrix cannot contain a right null-vector of Z, it could be constructed so as to contain all the right null-vectors of C^A. In this case, the rank of the symmetrization discrepancy would be reduced and, in some cases, might be eliminated.

There thus appears to be room for further improvement of the discrete nullifier in terms of the reduction of its symmetrization discrepancy.

Acknowledgment. Thanks are due to the referee for many helpful comments on the manuscript. Although he is quite right in feeling that a worked numerical example would be welcome, we are not at this time in a position to supply one, since there will unfortunately be no opportunity for the modest reprogramming of our Fortran program which would make this feasible. By way of mitigating this shortfall, we can again point to the fact that examples of the beneficial effect of the full rank boundary correction have been published in [4] and [5].

On the other hand, the connection that has been drawn between the continuous and the discrete forms of the basic variational formulation of the CD method is quite interesting and significant in a general sense, and has several conceptual "spinoffs" that we hope will prove useful in the future in constructing a convergence proof.

REFERENCES

[1] I. M. GELFAND AND S. V. FOMIN, *Calculus of Variations*, Prentice-Hall, Englewood Cliffs, NJ, 1963.

[2] I. N. SNEDDON, *Elements of Partial Differential Equations*, McGraw-Hill, New York, 1957.

[3] J. GREENSTADT, *Cell discretization algorithm with numerical examples*, IBM Palo Alto Scientific Center Report G320-3405, Palo Alto, CA, 1980.

[4] ———, *The cell discretization algorithm for elliptic partial differential equations*, this Journal, 3 (1982), pp. 261–288.

[5] J. GREENSTADT, *The application of cell discretization to nuclear reactor problems*, Nuclear Sci. Engrg., 82 (1982), pp. 78–95.
[6] L. P. EISENHART, *Continuous Groups of Transformations*, Princeton Univ. Press, Princeton, NJ, 1933.
[7] R. COURANT AND K. O. FRIEDRICHS, *Supersonic Flow and Shock Waves*, Interscience, New York, 1948.
[8] J. GREENSTADT, *On the reduction of continuous problems to discrete form*, IBM J. Res. Dev., 3 (1959), pp. 355–363.
[9] P. D. LAX AND R. D. RICHTMYER, *Survey of the stability of linear finite difference equations*, Comm. Pure Appl. Math., 9 (1956), p. 267.
[10] L. COLLATZ, *The Numerical Treatment of Differential Equations*, Springer-Verlag, New York, 1963.

SIAM J. SCI. STAT. COMPUT.
Vol. 7, No. 4, October 1986

015

LINEAR INVERSION OF BAND-LIMITED REFLECTION SEISMOGRAMS*

FADIL SANTOSA† AND WILLIAM W. SYMES‡

Abstract. We present a method for the linear inversion (deconvolution) of band-limited reflection seismograms. A large convolution problem is first broken up into a sequence of smaller problems. Each small problem is then posed as an optimization problem to resolve the ambiguity presented by the band-limited nature of the data. A new cost functional is proposed that allows robust profile reconstruction in the presence of noise. An algorithm for minimizing the cost functional is described. We present numerical experiments which simulate data interpretation using this procedure.

Key words. inverse problems, ill-posed problems, regularization, band-limited deconvolution, reflection seismology, optimization

AMS(MOS) subject classifications. 35, 49, 65, 86

1. Introduction. In this paper, we examine the problem of linear inversion of band-limited reflection data.

Suppose that wave propagation is modeled by the wave equation

$$A(s)u_{tt} = (A(s)u_s)_s, \qquad t > 0, \quad s \geqq 0, \tag{1.1}$$

where $u = u(t, s)$ is the particle displacement at time t and travel-time depth s. The coefficient $A(s)$ (with $A(0) = 1$) is the acoustical impedance as a function of travel time s, and is the desired unknown. The system is probed by a wavelet $f(t)$ which is described here as the boundary condition,

$$u_s(t, s = 0) = f(t), \qquad 0 \leqq t \leqq T. \tag{1.2}$$

The reflection seismogram is given by the surface particle velocity, i.e.,

$$u_t(t, s = 0) = h(t), \qquad 0 \leqq t \leqq T. \tag{1.3}$$

The problem then is to find $A(s)$ for $0 \leqq s \leqq T/2$ given the data pair $\{f(t), h(t)\}$.

In linear inversion, one takes $A(s) = 1 + \alpha(s)$ and $u(t, s) = u_0(t, s) + u_1(t, s)$, where $\alpha(s)$ and $u_1(t, s)$ are small perturbations. With the choice that $u_0(t, s)$ satisfy (1.1) with $A(s) \equiv 1$, the above inverse problem (1.1)–(1.3) becomes

$$u_{1tt} = u_{1ss} + \alpha'(s)f(t - s), \qquad t > 0, \quad s \geqq 0, \tag{1.4}$$

$$u_{1s}(t, s = 0) = 0, \tag{1.5}$$

$$u_{1t}(t, s = 0) = h(t) + f(t) \equiv g(t), \qquad 0 \leqq t \leqq T. \tag{1.6}$$

The inverse problem is to determine $\alpha(s)$ from $f(t)$ and $g(t)$. The problem is linear and can be recast into a convolutional equation as

$$g(t) = \int_0^t f(t - \tau)\alpha'\left(\frac{\tau}{2}\right) d\tau. \tag{1.7}$$

* Received by the editors February 7, 1985, and in revised form December 6, 1985. This research was sponsored by the Office of Naval Research under an SRO-III grant, contract number N-000-14-83-K0051.

† Department of Theoretical and Applied Mechanics, Cornell University, Ithaca, New York 14853. Present address Department of Mathematical Sciences, University of Delaware, Newark, Delaware 19716.

‡ Department of Mathematical Sciences, Rice University, Houston, Texas 77001.

Defining $r(t)$ as the impulse response, we have

$$g(t) = f(\cdot) * r(\cdot)(t). \tag{1.8}$$

If we know $r(t)$ in $0 \leqq t \leqq T$, we have $A(s) = 1 + \alpha(s)$ in $0 \leqq s \leqq T/2$ because

$$\alpha\left(\frac{s}{2}\right) = \frac{1}{2}\int_0^s r(t)\, dt. \tag{1.9}$$

As we can see, the problem of linear inversion amounts to solving the convolution equation (1.8) for the impulse response $r(t)$. We note here that $r(t)$ has a similar meaning to what is commonly called reflectivity function (see, e.g., Oldenburg et al. [1983]). We prefer the above formulation because it is consistent with nonlinear inversion (equations (1.1)-(1.3)) which is treated in our previous work (Santosa and Symes [1983]).

In most seismic applications the wavelet $f(t)$ is usually band-limited, i.e. has most of its energy concentrated within some passband. This either leads to nonunique solutions of (1.8) or ones which are very sensitive to noise. In particular, if low frequency information is missing, the trend information about the impedance $\alpha(s)$ is also missing. In either case the resulting inversion leads to ambiguous results.

Various remedies for the ambiguity based on a variety of assumptions and prejudices have been proposed. In particular, the use of minimum l_1-criterion has been shown to be very useful in seismic applications. Before proceeding with the discussions of some results in literature, we cast (1.8) in matrix form. That is, we assume that the data have been sampled and we have time series vectors $\mathbf{g}$, $\mathbf{f}$ and $\mathbf{r}$, each in $\mathbb{R}^N$. Define the $N \times N$ matrix $\mathbf{F}$ as

$$F_{ij} = \begin{cases} f_{i-j+1}, & i \geqq j, \\ 0, & i < j. \end{cases} \tag{1.10}$$

Then our problem amounts to solving the linear system

$$\mathbf{Fr} = \mathbf{g}. \tag{1.11}$$

Taylor, Banks and McCoy [1979] pose the following optimization problem: Find $\mathbf{r}$ which minimizes

$$\phi_1(\mathbf{r}) = \|\mathbf{r}\|_1 + \lambda \|\mathbf{Fr} - \mathbf{g}\|_1 \tag{1.12}$$

where the l_1-norm of an $\mathbb{R}^N$ vector is defined by

$$\|\mathbf{v}\|_1 = \sum_{i=1}^{N} |v_i|.$$

This approach is discussed in Claerbout and Muir [1973], and has been shown to display robustness; that is if $\mathbf{g}$ contains a few very noisy components, the l_1-criterion produces consistent estimates of $\mathbf{r}$. The first term in (1.12) forces the solution to have a small l_1-norm, and has the effect of constructing a solution which has the least number of nonzero components in $\mathbf{r}$; i.e., a sparse spike train. The parameter λ is a measure of the relevance of the data fit and hence the irrelevance of the sparse spike criterion. We emphasize here that if the unknown impulse response is a sparse spike train, and the data is noiseless, the use of the minimum l_1-criterion will produce, under many circumstances, the correct result even when the data is band-limited. Hence in many situations, trend information about the impedance can be recovered by using this criterion even with a low frequency cut off. We will discuss this property in more detail and precise mathematical terms in the next section.

Levy and Fullagar [1981], and Oldenburg, Scheuer and Levy [1983] cast the problem in the discrete Fourier transform (DFT) domain. The underlying assumption for the use of DFT is that the matrix $\mathbf{F}$ is circulant, i.e. instead of $\mathbf{F}$ in (1.10), a matrix $\mathbf{F}^c$ is constructed,

$$F^c_{ij} = f_{i-j+1} \tag{1.13}$$

and f has been periodically extended: $f_{i-N} = f_i = f_{i+N}$. This assumption is certainly justified if $\mathbf{f}$ and $\mathbf{g}$ decay to zero. Writing the DFT of $\mathbf{f}$ and $\mathbf{g}$ as $\hat{\mathbf{f}}$ and $\hat{\mathbf{g}}$, we can find the pass band solution of $\mathbf{r}$ by Fourier division. This solution is the least squares minimum solution if we set $\hat{\mathbf{r}}$ outside the pass band to zero. To be more specific, we have

$$\hat{f}_k = \sum_{j=1}^{N-1} f_j \exp\left\{-\left(\frac{2\pi i(j-1)(k-1)}{N}\right)\right\}$$

and similar definition for $\hat{g}_k$. The passband solution for $\hat{r}$ is

$$\hat{r}^p_k = \begin{cases} \hat{g}_k/\hat{f}_k, & \text{if } |\hat{f}_k| \geqq \varepsilon, \\ 0, & \text{if } |\hat{f}_k| < \varepsilon. \end{cases} \tag{1.14}$$

The above-mentioned authors proposed other ways of determining the cutoff in (1.14). In any event, the idea is to use $\hat{r}^p_k$ as constraints in an optimization problem. The problem they pose is:

Find $\mathbf{r}$ which minimizes

$$\phi_2(\mathbf{r}) = \|\mathbf{r}\|_1$$

subject to

$$\begin{aligned} &\operatorname{Re}(\hat{r}^p_k) - \alpha\delta_k \leqq \operatorname{Re}(\hat{r}_k) \leqq \operatorname{Re}(\hat{r}^p_k) + \alpha\delta_k \\ &\operatorname{Im}(\hat{r}^p_k) - \alpha\delta_k \leqq \operatorname{Im}(\hat{r}_k) \leqq \operatorname{Im}(\hat{r}^p_k) + \alpha\delta_k \end{aligned} \quad \text{for } |\hat{f}_k| \geqq \varepsilon, \tag{1.15}$$

where $\hat{\mathbf{r}}$ is the DFT of $\mathbf{r}$. In the event that the data is noise-free, one can use $\hat{\mathbf{r}}^p$ as equality constraints by setting $\alpha = 0$ in (1.15). When noise is expected, the data are used as inequality constraints by appropriately choosing α and δ_k above.

Problems (1.12) and (1.15) can be solved using a linear programming algorithm, and both approaches have yielded very good results which can be found in the references cited. Our approach is similar. We propose to find $\mathbf{r}$ which minimizes

$$\phi_3(\mathbf{r}) = \|\mathbf{r}\|_1 + \lambda \|\mathbf{Fr} - \mathbf{g}\|_2^2. \tag{1.16}$$

The l_2-norm is chosen for several reasons. First, the l_2-space is appropriate for data space as errors are normally estimated in percent RMS. Second, our approach will subsequently be used to solve nonlinear problems (1.1)-(1.3) where all the known results about the properties of the map from $r(t)$ to $A(s)$ are given in L^2 spaces (Symes [1983]). Finally, we choose this norm because it makes use of the data globally (as opposed to (1.15) where each frequency is used to describe a feasible region in the optimization space). The parameter λ plays the same role as in (1.12).

It will be difficult to say which of these methods is the best for all purposes; each one of them will be most suited to solving a certain type of problem. In the case where there are "wild" data points in $\mathbf{g}$ the cost functional in (1.12) is preferred. However, when there is a good global fit to the data, (1.16) should work better because the set $\{\mathbf{r}: \|\mathbf{Fr} - \mathbf{g}\|_1 \leqq c\}$ is smaller than $\{\mathbf{r}: \|\mathbf{Fr} - \mathbf{g}\|_2^2 \leqq c\}$ for small $c > 0$. Hence it is likely that a solution with smaller $\|\mathbf{r}\|_1$, could be found using $\phi_3(\mathbf{r})$ as cost functional. The cost

functional $\phi_2(\mathbf{r})$ is suited for situations in which knowledge about Fourier components of $\hat{r}_k^p$ is available, so that α and δ_k may be prescribed appropriately.

We will proceed by discussing the validity of using the l_1-criterion for deconvolution in § 2. In many physical applications, the wavelet is of a small time duration. We will use this fact and the linearity of the problem to our advantage, and show that the deconvolution problem can in fact be approximated by recursion of small optimization problems, as described in § 3. In § 4, we display examples in which the l_1-criterion produces noise-sensitive results when the data are used as constraints, and a pathological example in which the l_1-criterion does not resolve the ambiguity due to the existence of nontrivial nullvectors for $\mathbf{F}$. A brief qualitative description of the properties of the cost functional (1.16) and results of numerical experiments are presented in § 5. In § 6, we describe the algorithm for the optimization. A new algorithm has been devised as the optimization problem (1.16) cannot be solved using any standard methods. An algorithm that uses the recursive structure of the problem is described in § 7, where we also give numerical examples. Some concluding remarks are made in § 8.

2. The l_1-criterion. It has been shown by several authors (Levy and Fullagar [1981], Oldenburg et al. [1983]) that it is possible to construct a sparse spike train from part of its spectrum using the minimum l_1-criterion. This well-known (and believed) property of the l_1-criterion can in fact be demonstrated in a mathematical way. We proved this property in Santosa and Symes [1983]. We repeat the proof and discuss the implications of the results.

We consider the optimization problem in (1.15). Instead of inequality constraints, we take $\alpha = 0$ and arrive at the following constrained optimization:

$$\begin{aligned} &\text{minimize} && \|\mathbf{r}\|_1 \\ &\text{subject to} && \mathbf{F}^c\mathbf{r} = \mathbf{g}, \end{aligned} \tag{2.1}$$

where $\mathbf{F}^c$ is circulant. Because we can use DFT to decompose the convolution, we can cast the problem in terms of Fourier components. Define the index set $I = \{k \in I: |\hat{f}_k| = 0\}$; the complement I_c can be thought of as the passband of the vector $\mathbf{f}$. Then we can equivalently pose the following optimization:

$$\begin{aligned} &\text{minimize} && \|\mathbf{r}\|_1 \\ &\text{subject to} && \hat{r}_k = \hat{g}_k/\hat{f}_k \quad \text{for } k \in I_c. \end{aligned} \tag{2.2}$$

So, in this problem, we restrict the search to the subspace defined by Fourier components whose index is in I.

We want to show that the minimum of (2.2) occurs when $\mathbf{r}$ is a sparse spike train. That is, at $\mathbf{r}$ equal to a spike train, moving in any direction on the subspace defined by the index set I increases the l_1-norm of $\mathbf{r}$. This is true under certain conditions to be made explicit later. We need to show that for a spike train $\mathbf{r}$ of ν spikes located at $j = j_n$,

$$r_j = \sum_{n=1}^{\nu} \gamma_n \delta_{jj_n}, \tag{2.3}$$

any addition to r_j in (2.3) of the form

$$\delta r_j = \frac{1}{N} \sum_{k \in I} \delta\hat{r}_k \exp\left(-\frac{2\pi i(j-1)(k-1)}{N}\right) \tag{2.4}$$

will increase the l_1-norm.

The way we showed this property was to estimate lower bounds of $|r_j+\delta r_j|$ for $j=j_n$ and $j\neq j_n$. From the triangle inequality and the fact that $|\exp i\alpha|=1$ we obtain

$$|r_{j_n}+\delta r_{j_n}|\geqq|\gamma_n|-\frac{K}{N}\|\delta\hat{\mathbf{r}}\|_\infty \tag{2.5}$$

and

$$\sum_{j\neq j_n}|r_j+\delta r_j|=\sum_{j\neq j_n}|\delta r|\geqq\left(1-\frac{\nu K}{N}\right)\|\delta\hat{\mathbf{r}}\|_\infty, \tag{2.6}$$

where K is the number of elements in I (hence can be interpreted as the number of missing components). These two inequalities lead to the estimate

$$\|\mathbf{r}+\delta\mathbf{r}\|_1\geqq\|\mathbf{r}\|_1+\left(1+\frac{2\nu K}{N}\right)\|\delta\hat{\mathbf{r}}\|_\infty. \tag{2.7}$$

In terms of reconstructing a spike train from part of its spectrum, (2.7) can be used as a rough guide to when the l_1-criterion would work. The estimate states that if the number of spikes (ν) times the number of missing components (K) is less than $N/2$, then using the l_1-criterion would lead to the complete reconstruction of the missing spectrum. In arriving at (2.7), we have not used the fact that the spikes may be separated by large distances, nor the nature of the subspace in (2.4) defined via the Fourier transform. Hence, it is most likely (as numerical experience has shown) that the estimate is suboptimal and pessimistic.

The requirement $\nu K>N/2$ paints a grim picture of the situation, especially when $|\hat{f}_k|=0$ for k small as well as large (recall that K is the number of elements in I). However, if we are willing to sacrifice resolution to obtain low frequency information, i.e., trend in the impedance, we can use a resampling trick.

Let $f(t)$ be a continuous time signal with $f(t<0)=0$, and $f_j=f((j-1)\Delta t)$ for $j=1,2\cdots N$. The Fourier transform of $f(t)$ is

$$\hat{f}(w)=\int_{-\infty}^{\infty}f(t)\exp(-2\pi iwt)\,dt.$$

Suppose that the support (pass band) of $\hat{f}(w)$ is $\operatorname{supp}[\hat{f}(w)]=[w_1,w_2]$. If we sample at Δt increments, the Nyquist limit of the DFT is $w_N=\frac{1}{2}\Delta t$. Thus if we set $w_N=w_2$, or $\Delta t=\frac{1}{2}w_2$, then we arrive at a situation in which only the DFT components corresponding to low frequencies ($w<w_1$) are missing. This procedure reduces the number K to approximately $K\simeq w_1N/2w_2$. In terms of realistic numbers, say that the wavelet has energy concentrated in the 10 to 40 Hz band. This tells us we should sample at $\Delta t=\frac{1}{80}$ s, and the number $K/N\equiv\frac{1}{8}$. Using (2.7), we conclude that the number of spikes in $\mathbf{r}$ that generated the data must be 4 or less for the l_1-criterion to completely reconstruct the spectrum of $\mathbf{r}$ from 0 to 10 Hz. Even though this prediction is on the pessimistic side, it reveals that without incorporating other a priori information, the l_1-criterion by itself may lead to nonsensical results. This phenomenon has been observed by Oldenburg et al. [1983] and uses of other information have been proposed (see also Oldenburg [1984]).

To conclude this section, we generalize the result we proved above slightly to the case of noncirculant F. We write the singular value decomposition of F (cf. e.g. Golub and Van Loan [1983], and Stewart [1973]) as

$$\mathbf{F}=\mathbf{U}\Lambda\mathbf{V}^{\mathrm{T}}$$

where $\mathbf{U}$ and $\mathbf{V}$ are $N \times N$ orthonormal matrices, and $\Lambda = \text{diag}\,(\sigma_1, \sigma_2, \cdots, \sigma_N)$ where σ_i are the singular values. The object is to show that perturbations of the form

$$\delta \mathbf{r} = \sum_{k \in I} \delta \hat{r}_k \mathbf{v}^k \tag{2.8}$$

where $\mathbf{v}^k$ is the kth column of $\mathbf{V}$, to the spike train $\mathbf{r}$ defined in (2.3), will increase the l_1-norm. Note that this problem reduces to the previous one with

$$(v^k)_n = \frac{1}{\sqrt{N}} \exp\left\{\frac{2\pi i(k-1)(n-1)}{N}\right\}$$

if $\mathbf{F}$ is circulant.

We estimate as before

$$|\delta r_j| = \left| \sum_{k \in I} \delta \hat{r}_k v_j^k \right| \leqq K \sup_{k \in I} |\delta \hat{r}_k| \sup_{k \in I} \|\mathbf{v}^k\|_\infty \tag{2.9}$$

and

$$|\delta \hat{r}_k| = |(\mathbf{v}^k)^T \delta \mathbf{r}| \leqq \|\delta \mathbf{r}\|_1 \|\mathbf{v}^k\|_\infty. \tag{2.10}$$

Now we make *uniformity assumptions*: there exist $c > 0$ so that for $k \in I$, any j_1 and J, $1 \leqq j_1 \leqq j \leqq j_1 + J \leqq N$

$$\frac{1}{J} \sum_{j=j_1}^{j_1+J} |v_j^k|^2 \leqq c/N. \tag{2.11}$$

This means that the local average of the components of $\mathbf{v}^k$ is not very far from the total average of the components, which is $1/N$. This is a version of the uncertainty principle: the eigenvector $\mathbf{v}^k$ is perfectly concentrated in spectral coordinates, but must be spread quasi-uniformly in configuration coordinates. In the circulant case, it is a consequence of $|e^{iw}| = 1$.

Notice that the requirement (2.11) must hold for $\mathbf{v}^k$, $k \in I$. In particular, if I corresponds to the low frequency components of $\mathbb{R}^N$, we expect the uniformity assumptions to hold. It follows from (2.11) that we will have, for some (other) $c > 0$,

$$\|\mathbf{v}^k\|_\infty \leqq cN^{-1/2}, \tag{2.12}$$

for $k \in I$, hence (2.9) becomes

$$|\delta r_j| \leqq Kc \sup_{k \in i} |\delta \hat{r}_k| N^{-1/2}$$

and (2.10) becomes

$$|\delta \hat{r}_k| \leqq c \|\delta \mathbf{r}\|_1 N^{-1/2}.$$

With these estimates in hand, we can repeat the rest of the argument given above and obtain

$$\|\mathbf{r} + \delta \mathbf{r}\|_1 \geqq \|\mathbf{r}\|_1 + \frac{\sqrt{N}}{c}\left(1 - \frac{2\nu cK}{N}\right) \sup_{k \in i} |\delta \hat{r}_k|. \tag{2.13}$$

For matrices like the Toeplitz matrices considered here, which are nearly circulant, we expect $c \simeq 1$, so we would expect to be able to recover roughly the same number of spikes as before.

3. Recursive structure and small optimization problems. In many applications, the size of the convolution problem is normally very large. However, it is possible to reduce

the large problem into a sequence of small problems if we are willing to take a different point of view. The size of the small problem depends on the duration of the wavelet.

Suppose the trace Γ is of length M and the duration of the wavelet $\mathbf{f}$ is of length n. The matrix equation describing the convolution is displayed as

$$\Gamma = \Phi \mathbf{T} \tag{3.1}$$

where $\mathbf{T}$ is the desired impulse response, and Φ, an $M \times M$ matrix, is lower banded of bandwidth n (see Fig. 1). This is a size-M problem.

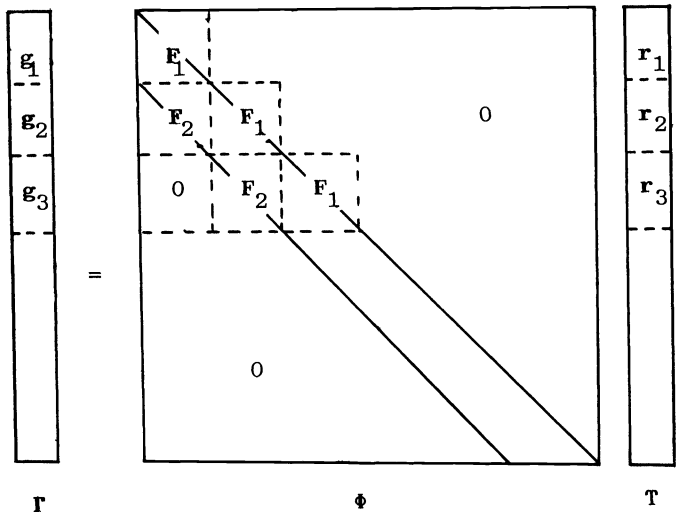


FIG. 1. *The source wavelet* $\mathbf{f}$ *is assumed to be of length m. The convolution equation for the entire trace is written in matrix form* $\Gamma = \Phi \mathbf{T}$, *where* Γ *and* $\mathbf{T}$ *are* $\mathbb{R}^M$, *and* Φ *is* $M \times M$. *The trace and the unknown is partitioned into* $\mathbf{g}_i$ *and* $\mathbf{r}_i$ *of length m. The matrix is made up of blocks of* $m \times m$ *lower triangular* $\mathbf{F}_1$ *and upper triangular* $\mathbf{F}_2$. *The convolution equation is written as* $\mathbf{g}_1 = \mathbf{F}_1\mathbf{r}_1$, $\mathbf{g}_2 = \mathbf{F}_1\mathbf{r}_2 + \mathbf{F}_2\mathbf{r}_{i-1}$ *for* $i > 1$.

We proceed by partitioning Γ and $\mathbf{T}$ into pieces of length m, $n \leqq m \leqq M$ namely vectors $\mathbf{g}_i$ and $\mathbf{r}_i$ respectively. Because of the banded nature of Φ, we can express it as a union of copies of two $m \times m$ matrices $\mathbf{F}_1$ and $\mathbf{F}_2$. Clearly, (3.1) can be rewritten recursively as

$$\mathbf{g}_1 = \mathbf{F}_1\mathbf{r}_1, \qquad \mathbf{g}_i = \mathbf{F}_2\mathbf{r}_{i-1} + \mathbf{F}_1\mathbf{r}_i. \tag{3.2}$$

We take the attitude that Γ is in the range of Φ, i.e., an exact fit is possible. When Γ is not in the range we can filter out outside passband components of Γ to put it in the range. Thus our view is that whatever is outside the passband is taken as noise. If Φ has no null space, i.e. $\mathbf{f}$ full band width in Fourier components, then Φ is nonsingular and $\mathbf{F}_1$ is nonsingular (see e.g. Stewart [1973]). In this case, the system (3.2) can be solved by backsubstitution recursively from knowledge of $\mathbf{F}_1^{-1}$. Moreover, the solutions of (3.1) and (3.2) are *identical*, i.e.,

$$\mathbf{T}^T = [\mathbf{r}_1^T | \mathbf{r}_2^T | \mathbf{r}_3^T | \text{etc.}].$$

However, if Φ has a null space, we have many solutions and we must make a choice to pick out a particular solution using some criterion. For example, we can take as a solution to $\mathbf{T}$ in (3.1), the solution with the least l_2-length, or the solution with the least l_1-length as we have discussed. Let us take the l_1-criterion: we seek $\mathbf{T}^*$ which

minimizes

$$\|\mathbf{T}\|_1 \quad \text{subject to } \mathbf{\Phi T} = \mathbf{\Gamma}. \tag{3.3}$$

It is tempting to pose, equivalently, for (3.2):

For $i = 1, 2, \cdots$, find $\mathbf{r}_i^*$ which minimizes

$$\|\mathbf{r}_i\|_1 \tag{3.4}$$

subject to $\mathbf{F}_1\mathbf{r}_i = \mathbf{g}_1 - \mathbf{F}_2\mathbf{r}_{i-1}^*$ for $i > 1$

(with the convention $\mathbf{r}_0^* = 0$). This preserves the recursive structure of the problem and subsequently decouples the constraints on $\mathbf{r}_i$. However, it is likely that the solutions $\mathbf{T}^*$ and $\mathbf{r}_i^*$ are unequal, i.e.,

$$\mathbf{T}^{*T} \neq [\mathbf{r}_1^{*T} | \mathbf{r}_2^{*T} | \mathbf{r}_3^{*T} \text{ etc.}].$$

We display this fact in an example.

Let $\mathbf{\Phi}$, and subsequently $\mathbf{F}_1$, and $\mathbf{F}_2$ be

$$\mathbf{\Phi} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}, \qquad \mathbf{F}_1 = \begin{bmatrix} 0 & 0 \\ 1 & 0 \end{bmatrix}, \qquad \mathbf{F}_2 = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}. \tag{3.5}$$

We will use singular value decompositions to analyze the matrices $\mathbf{\Phi}$, $\mathbf{F}_1$, and $\mathbf{F}_2$. By solving the eigenvalue problems for $\mathbf{\Phi}\mathbf{\Phi}^T$ and $\mathbf{\Phi}^T\mathbf{\Phi}$, we can decompose $\mathbf{\Phi}$ into

$$\mathbf{\Phi} = \mathbf{U}\mathbf{\Lambda}\mathbf{V}^T = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{3.6}$$

We take a vector $\mathbf{T}^T = (0\ 1\ 1\ 1)$ which is in the range of $\mathbf{\Phi}$. The solutions to $\mathbf{\Phi T} = \mathbf{\Gamma}$ can be parameterized as

$$\mathbf{T}^T = (1\quad 1\quad 1\quad 0) + \alpha(0\quad 0\quad 0\quad 1); \tag{3.7}$$

the first part of the solution is the part obtained by applying the pseudo-inverse $\mathbf{\Phi}^+$ on $\mathbf{\Gamma}$, and the second part corresponds to the null space component (the component that does not effect the data $\mathbf{\Gamma}$). The value α is to be determined by requiring $\|\mathbf{T}\|_1$ minimum. This requirement forces $\alpha = 0$. Hence, we have the solution to (3.3)

$$\mathbf{T}^{*T} = (1\quad 1\quad 1\quad 0). \tag{3.8}$$

Using the same tool, we find that $\mathbf{F}_1$ can be written as

$$\mathbf{F}_1 = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}. \tag{3.9}$$

We break up $\mathbf{\Gamma}$ into $\mathbf{g}_1 = (0\ 1)$ and $\mathbf{g}_2 = (1\ 1)$. The solution to $\mathbf{F}_1\mathbf{r}_1 = \mathbf{g}_1$ is

$$\mathbf{r}_1^T = (1\quad 0) + \alpha(0\quad 1). \tag{3.10}$$

Therefore, we take $\alpha = 0$ and $\mathbf{r}_1^{*T} = (1\ 0)$. Next apply $\mathbf{F}_2$ to $\mathbf{r}_1^*$ and solve

$$\mathbf{F}_1\mathbf{r}_2 = \mathbf{g}_2 - \mathbf{F}_2\mathbf{r}_1^* = \begin{pmatrix} 1 \\ 1 \end{pmatrix}.$$

Apply again $\mathbf{F}_1^+$ to the above and obtain

$$\mathbf{r}_2^T = (1 \quad 0) + \alpha(0 \quad 1). \tag{3.11}$$

The l_1-criterion forces $\alpha = 0$. Hence

$$(\mathbf{r}_1^{*T} | \mathbf{r}_2^{*T}) = (1 \quad 0 \quad 1 \quad 0) \tag{3.12}$$

and *not* equal to $\mathbf{T}^{*T}$. In fact, with (3.12), we no longer have an exact data fit; a residual has been introduced.

Despite this example, we can envision circumstances under which the solution of the problem (3.4) might accomplish the desired task—that is, retrieve a sparse spike train from its discrete convolution with a noninvertible kernel—even when (3.3) fails because the estimate (2.7) is violated. Suppose we assume:

(i) the target spike train is *sparse*;

(ii) the small singular values of F_1 have uniformly distributed singular vectors.

We shall quantify these notions and give a plausible argument (not a proof!) that (i) and (ii) ought to imply the recovery of the target reflectivity $\mathbf{T}$ by means of (3.4). We will give examples in § 7, which support our contention.

We shall take the sparseness of $\mathbf{T}$ to mean precisely that the segments $\mathbf{r}_i$ of the target reflectivity have only a few (significant) nonzeros. Assumption (ii) means that the singular vectors associated with $\mathbf{F}_1$ are uniformly distributed in the sense of the discussion at the end of § 2, especially inequality (2.12). We expect this property to hold for slowly varying (low-frequency) singular vectors because, for such vectors, the action of $\mathbf{F}_1$ differs little from the action of the related circulant matrix. Indeed if $\mathbf{F}_1$ were circulant, then the singular value decomposition would be given by the discrete Fourier transform, and the characterization in terms of low frequencies would have the usual meaning. As we have mentioned earlier, the impossibility of localizing energy is just the discrete version of the uncertainty principle. We assume the same property for the SVD of $\mathbf{F}_1$.

Let $\mathbf{r}_i$, the subvectors of $\mathbf{T}$, be sparse, and $\mathbf{T}$ satisfy (3.1) (hence (3.2) also). We reconstruct the first segment $\mathbf{r}_i$ from the optimization problem in (3.4), i.e., we find $\mathbf{r}_1^*$ which minimizes

$$\|\mathbf{p}\|_1 \qquad \text{subject to } \mathbf{F}_1\mathbf{p} = \mathbf{g}_1.$$

We argue that $\mathbf{r}_1^*$ must be identical to $\mathbf{r}_1$. The constraints allow us to write $\mathbf{p} = \mathbf{p}_0 + \mathbf{p}'$, where $\mathbf{p}_0 = \mathbf{F}_1^+ \leftrightarrow \mathbf{g}_1$ and $\mathbf{F}_1\mathbf{p}' = 0$. Since we have assumed uniformity for the nearly-null vectors, hence for the null vectors, of $\mathbf{F}_1$, we have the result of § 2: if the coset $\mathbf{p}_0 + \ker \mathbf{F}_1$ contains a vector with few enough nonzeros, then the vector is the minimizer of the l_1-functional on the coset, i.e., $\mathbf{r}_1^*$. On the other hand, we have assumed that $\mathbf{r}_1$ consists of few nonzeros: if we define few to mean the number of nonzeros

$$\nu < m/2kc \tag{3.13}$$

(cf. (2.13)). Hence, we must necessarily have $\mathbf{r}_1 = \mathbf{r}_1^*$.

Here $k = \dim \ker \mathbf{F}_1$. Since $\mathbf{F}_1$ and Φ are both assumed to represent discretizations of the same convolution operator at the same sampling rate, we would expect the ratio n/k to remain roughly constant, at least if n is not too much smaller than M, and approximately the same as M/K. If the significant nonzeros are spread out more-or-less uniformly over $\mathbf{T}$, then (3.13) will be satisfied as soon as n is small enough.

Since we have correctly recovered the first segment $\mathbf{r}_1$ of $\mathbf{T}$, we can now go to the second segment. The problem for the reconstruction of $\mathbf{r}_2$ is to find $\mathbf{r}_2^*$ which minimizes

$$\|\mathbf{p}\|_1 \qquad \text{subject to } \mathbf{F}_1\mathbf{p} = \mathbf{g}_2 - \mathbf{F}_2\mathbf{r}_1.$$

The sparseness of $\mathbf{r}_2$ and the uniformity of the nearly null singular vectors of $\mathbf{F}_1$ guarantee that $\mathbf{r}_2^* = \mathbf{r}_2$. The process is repeated until we have completely recovered $\mathbf{T}$.

It is interesting to note that, if $\mathbf{T}$ has correctly placed *gaps* (in its support) of width greater than or equal to the bandwidth of the matrix $\mathbf{\Phi}$, then the product $\mathbf{F}_2\mathbf{r}_1$ vanishes and the problem completely decouples. As the identification of such gaps is a substantial issue, this observation is probably not too important.

The main question, which our heuristic reasoning leaves completely open, is whether a subdivision, that is, the number m, can be chosen so that $\mathbf{r}_1$ has few enough spikes, and yet the nearly null vectors of $\mathbf{F}_1$ closely enough resemble those of $\mathbf{\Phi}$ in their uniformity estimates. This is a complicated condition, involving the SVDs of submatrices of $\mathbf{\Phi}$ and the sparsity structure of $\mathbf{T}$. We take the attitude that this question is best investigated numerically at present. We give the results of numerical experiments in § 7.

Note that the 4×4 example given above violated stricture (ii) rather grossly: the null vector $(0, 0, 0, 1)^T$ was not evenly distributed. In general, it is impossible to write lower-triangular (noncirculant) band matrices with uniformly distributed null-vectors. For instance, the periodic $m \times m$ first-difference operator

$$\mathbf{F}_1^c = \begin{bmatrix} 1 & 0 & \cdots & 0 & -1 \\ -1 & 1 & & 0 & 0 \\ 0 & -1 & & 0 & 0 \\ \vdots & \vdots & & \vdots & \vdots \\ 0 & 0 & \cdots & -1 & 1 \end{bmatrix}$$

has the normalized null-vector $(m^{1/2}, \cdots, m^{1/2})^T = \mathbf{v}$. Its noncirculant relative $\mathbf{F}_1$, obtained by setting the $(1, m)$ entry to zero, is actually nonsingular, but

$$\|\mathbf{F}_1\mathbf{v}\|_2 = m^{-1/2}.$$

Thus $\mathbf{v}$ resides mostly in the small singular subspaces of $\mathbf{F}_1$ as $m \to \infty$. This example indicates that we should expect the considerations outlined above to be predictive when the sampling is fine enough that the lower end of the spectrum of $\mathbf{F}_1$ comes to resemble that of the related periodic convolution.

Finally, we remark that the recursive structure may be taken advantage of in reconstruction techniques where the data is used as penalty or inequality constraints, such as those discussed in the introduction. For the l_2-penalized problem, we pose for $i = 1, 2, 3, \cdots$, find $\mathbf{r}_i^*$ which minimizes

$$\phi(\mathbf{r}_i) = \|\mathbf{r}_i\|_1 + \lambda \|\mathbf{F}_2\mathbf{r}_{i-1}^* + \mathbf{F}_1\mathbf{r}_i - \mathbf{g}_i\| \tag{3.14}$$

(with the convention $\mathbf{r}_i^* = 0$). We expect that a sparse spike $\mathbf{T}$ train can be recovered by solving (3.14) in the presence of noise in the data $\mathbf{g}_i$. Further discussion can be found in the next two sections. Numerical examples are displayed in § 7.

4. Use of data as constraints. In § 2, we showed that a sparse spike train can be reconstructed from part of its spectrum. The idea is to use the passsband data as constraints, and seek a solution with the least l_1-norm. In other words, we are required to solve:

Find $\mathbf{r}^*$ which minimizes

$$\|\mathbf{r}\|_1 \tag{4.1}$$

subject to $\mathbf{Fr} = \mathbf{g}$.

In this and the next section, we will attack the small subproblems that arise in using the recursive structure outlined in the previous section. We will return to solving the big problem in § 7.

As an example of using (4.1) we took a sparse spike train displayed in Fig. 2a (shown also is $\alpha(s)$, the impedance correction) convoluted it with the band-limited wavelet **f** in Fig. 2b, to obtain the data **g** in Fig. 2c. We use an algorithm devised by Bartels, Conn and Sinclair [1978] to solve an overdetermined system of equations. To

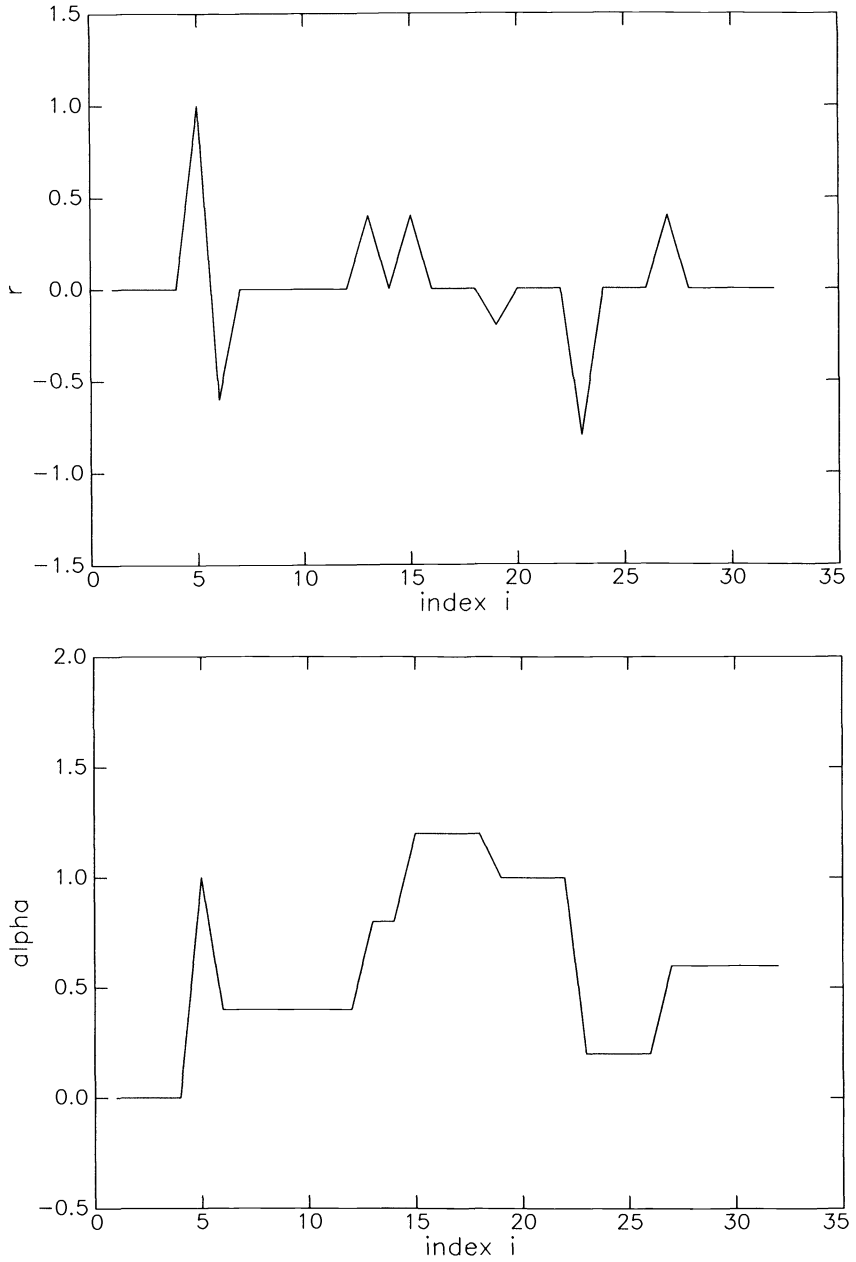


FIG. 2a. *The generating sparse spike train* **r**. *The trace is of length* 32 *points. Shown below is the corresponding impedance correction* $\alpha(t)$.

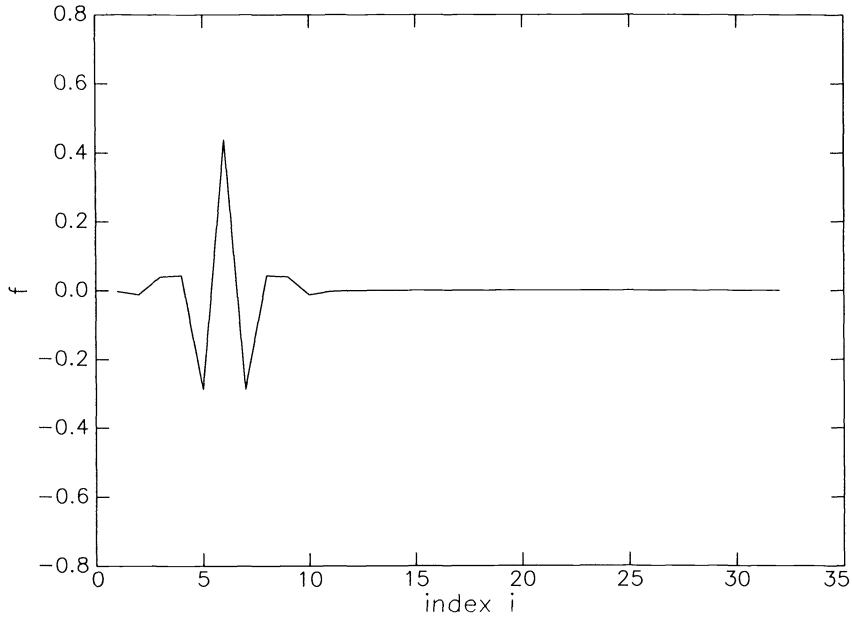


FIG. 2b. *The source wavelet* **f** *used in generating the data in Fig.* 2c. *When* **f** *is put in matrix form for convolution,* 10 *out of* 32 *singular values were found to be below tolerance of* 0.005.

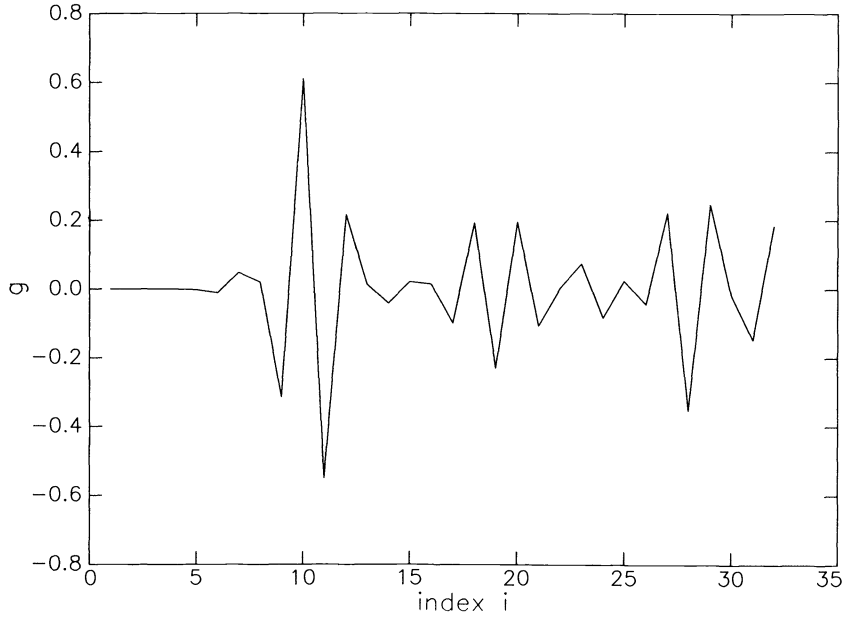


FIG. 2c. *The data* $\mathbf{g} = \mathbf{Fr}$.

be able to use their procedure, we incorporated the equality constraints into the cost functional. This can be accomplished in several ways (see e.g. Bartels et al. [1978]); however, we choose to make use of singular value decomposition on **F**.

For our example, we took **f**, **r** and **g** to be vectors of length 32. The wavelet duration is 12, i.e., **f** has 12 nonzero components. The matrix **F** then is 32×32 and

lower banded of bandwidth 12. We write $\mathbf{F}$ in terms of two orthonormal matrices $\mathbf{U}$ and $\mathbf{V}$, and a diagonal matrix $\Lambda = \text{diag}[\lambda_1, \lambda_2, \cdots]$, $\lambda_1 \geqq \lambda_2 \geqq \lambda_3 \cdots$. The matrices $\mathbf{U}$ and $\mathbf{V}$ are partitioned into parts $\mathbf{U}_p$ and $\mathbf{V}_p$ ($32 \times p$ matrices) $\mathbf{U}_0$ and $\mathbf{V}_0$ ($32 \times (32-p)$ matrices) corresponding to small singular values $\lambda_i < \varepsilon$ for $i > p$.

We construct a least length pseudo-inverse with $\mathbf{U}_p$ and $\mathbf{V}_p$ in the usual manner. The constraints can then be written as

$$\mathbf{r} = \mathbf{F}^+\mathbf{g} + \mathbf{V}_0\mathbf{x} \tag{4.2}$$

where $\mathbf{x}$ is a $(32-p)$-vector. The optimization problem is

$$\text{minimize} \quad \|\mathbf{F}^+\mathbf{g} + \mathbf{V}_0\mathbf{x}\|_1. \tag{4.3}$$

Once the optimal $\mathbf{x}^*$ has been found, we put it in (4.2) and obtain the optimal $\mathbf{r}^*$. We note here that we have found numerical evidence which shows that $\|\mathbf{F}^+(\mathbf{Fr})\|_1 < \|\mathbf{r}\|_1$. This happens when (i) $\mathbf{F}$ has too many zero singular values and (ii) $\mathbf{r}$ not a sparse spike train (cf. (2.13)). In any case, when this occurs, we cannot hope the l_1-criterion will help.

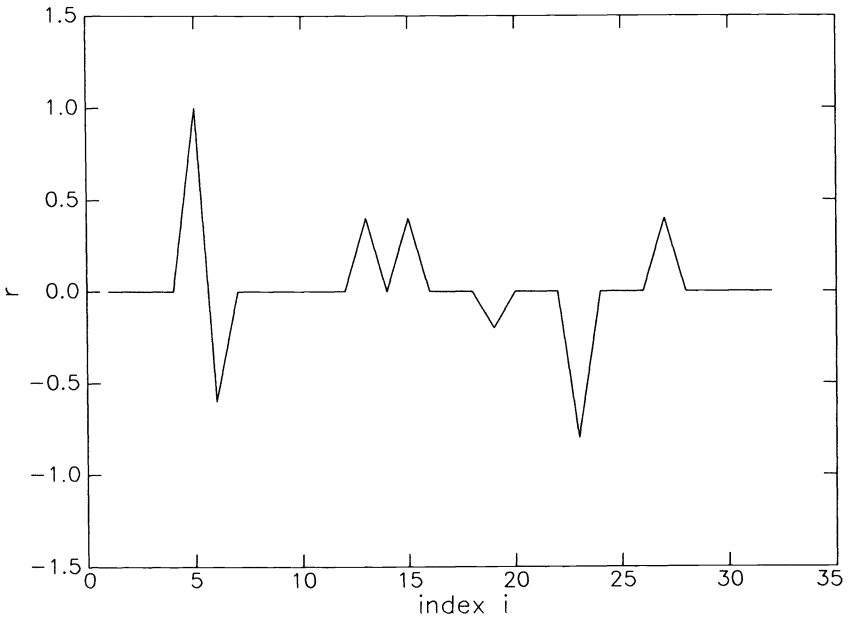


FIG. 2d. *Reconstruction of* $\mathbf{r}$ *using* $\mathbf{g}$ *and* $\mathbf{f}$ *in Figs.* 2b *and* 2c. *The data is used as constraints and the solution with the smallest* l_1*-norm is sought.*

We emphasize here that had we simply taken $\mathbf{r} = \mathbf{F}^+\mathbf{g}$, i.e. taking the least-squares minimum-length solution (equivalently passband Fourier division), we would have a profile $\alpha(s)$ with no *trend* in it. We display this in Fig. 2e.

The result of applying this procedure is displayed in Fig. 2d. The size $(32-p) = 10$ for this problem; for $\varepsilon = 0.005$. Observe that if we have noise in the data $\mathbf{g}$ which is in the orthogonal complement of the range of $\mathbf{F}$, the optimum will clearly be unaffected because $\mathbf{F}^+$ will simply project $\mathbf{g}$ back into the range of $\mathbf{F}$, and the matrix $\mathbf{V}_0$ will remain the same. However, if the noise is in the range of $\mathbf{F}$, $\mathbf{F}^+\mathbf{g}$ will be altered, and hence the constraints will be changed.

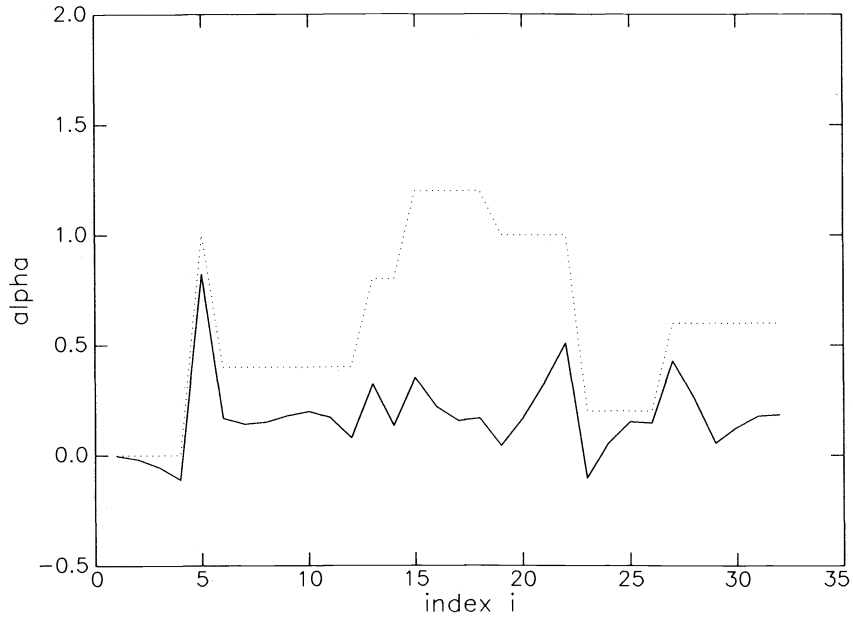


FIG. 2e. *The profile $\alpha(s)$ corresponding to* SVD-*inverse solution of* $\mathbf{Fr}=\mathbf{g}$. *Notice the lack of trend information.*

To display the effect of noise in $\mathbf{g}$ in the range of $\mathbf{F}$, we took the sparse spike in Fig. 2a, and introduced small perturbations to it. The resulting trace is shown in Fig. 3a. Next $\tilde{\mathbf{g}}$ is generated by a convolution. We apply the same procedure as outlined above and obtained as an optimum, the curves in Fig. 3b. To show that we have not altered the constraints too much, we plotted both $\mathbf{F}^{+}\mathbf{g}$ and $\mathbf{F}^{+}\tilde{\mathbf{g}}$, where $\tilde{\mathbf{g}}$ is noisy in

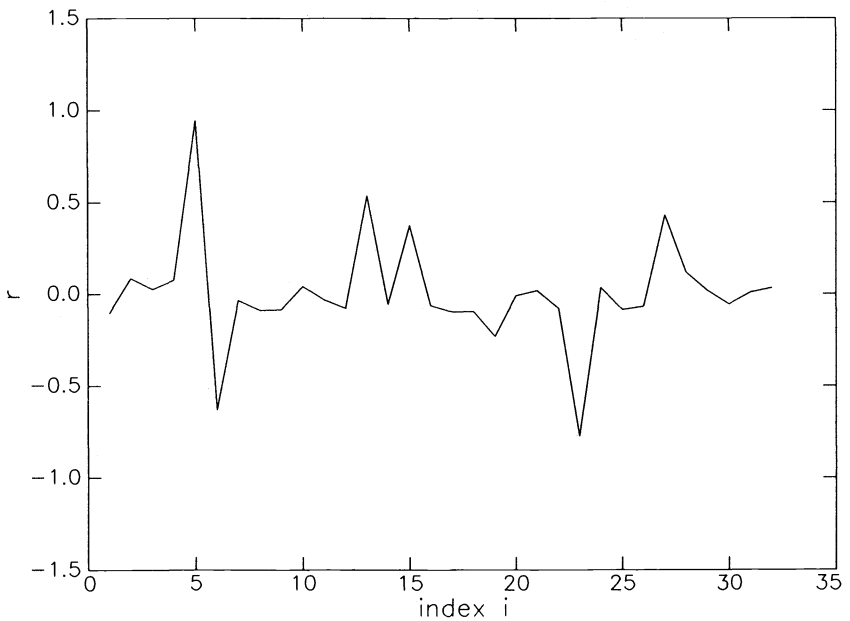


FIG. 3a. *The generating* $\tilde{\mathbf{r}}$ *which is not a sparse spike train. We convolved* $\mathbf{f}$ *with* $\tilde{\mathbf{r}}$ *to simulate a noisy data trace* $\mathbf{g}$. *The* l_2*-relative error of* $\tilde{\mathbf{g}}$ *and* $\tilde{\mathbf{g}}$ *is* 18 *percent.*

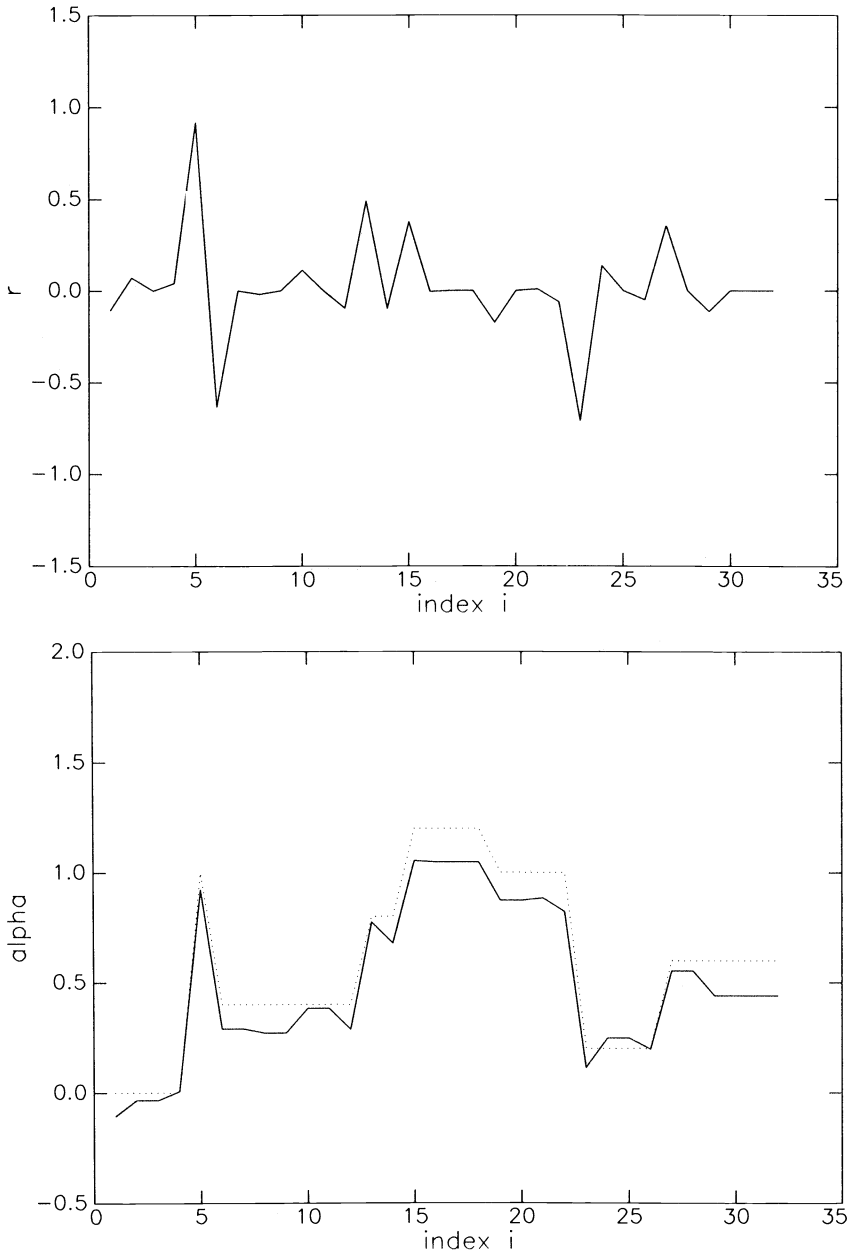


FIG. 3b. *The constrained optimum from noisy data, and corresponding* $\alpha(s)$, *compare with Fig.* 2a.

Fig. 3c. This exercise shows that the constrained minima of (4.1) may be very sensitive to perturbations in the constraints. This is one of the reasons we choose to incorporate the data $\mathbf{Fr}=\mathbf{g}$ as penalty instead of constraints.

Next we demonstrate that under some circumstances, the l_1-criterion, or variants of it, will not resolve the ambiguity in the problem. We consider a 2-D example:

$$\begin{aligned} &\text{minimize} && |x_1|+|x_2| \\ &\text{subject to} && x_2=1-(1+\varepsilon)x_1. \end{aligned} \tag{4.4}$$

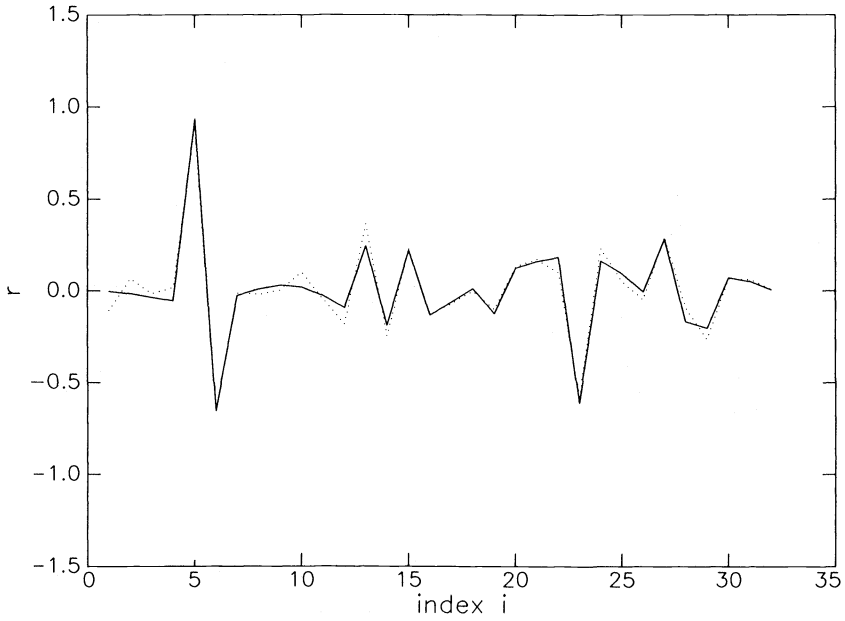


FIG. 3c. *Comparison of the constraints using noise-free and noisy data. Plotted are the vectors* $\mathbf{F}^+\mathbf{g}$ *and* $\mathbf{F}^+\tilde{\mathbf{g}}$.

So, the family of solutions to our "data" is the set of points on $x_2 = 1-(1+\varepsilon)x_1$. Now, we want to "resolve" the ambiguity by choosing a solution with the least l_1-norm. Notice that the level sets of $|x_1|+|x_2|$ in the first quadrant are 45° lines as shown in Fig. 4. The set of points satisfying the data also lie on a line inclined at nearly 45° for ε small.

We make the following observations on the optimal solution of (4.4).

(i) for $\varepsilon = 0$, optimum $x_2 = 1-x_1$, nonuniqueness,

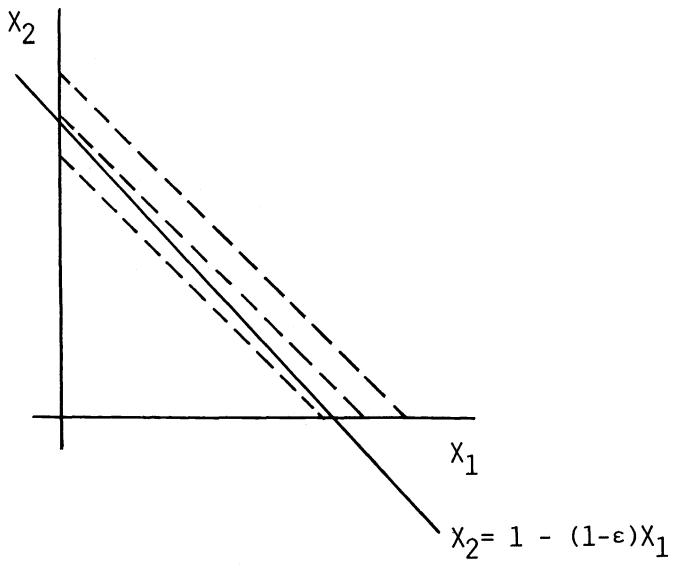


FIG. 4. *Drawn with dashed lines are the level sets of* $|x_1|+|x_2|$. *The "data" requires the solution to lie on the line given by* $x_2 = 1-(1+\varepsilon)x_1$. *When* $\varepsilon = 0$, *the line becomes parallel to the level set, when* $\varepsilon > 0$, *the line is steeper, and when* $\varepsilon < 0$, *the line is flatter than the level sets.*

(ii) for $\varepsilon > 0$, optimum $x_1 = 1/(1+\varepsilon)$, $x_2 = 0$,

(iii) for $\varepsilon < 0$, optimum $x_1 = 0$, $x_2 = 0$.

Thus we get three very different results for three very small changes in the data.

This phenomenon occurs when the bottom of the cost functional of $\|\mathbf{F}^+\mathbf{g} + \mathbf{V}_0\mathbf{x}\|_1$ is flat; a small perturbation in $\mathbf{g}$ changes the shape of the bottom and hence displacing the optimum by a large amount. The ambiguity may be resolved in real physical situation by incorporating more a priori information as constraints. Ideas along these lines are explored in Oldenburg [1984].

5. Use of data as penalty. From the results of the constrained problem, we observe that when the data $\mathbf{g}$ is perturbed, the optimization produced a solution which has a larger l_1-norm than the unperturbed solution. This is so because the former is forced by the constraints to attain its minimum l_1-length at a point with many nonzero components (compare Figs. 2d and 3b). If the solution had been allowed to be displaced slightly away from the subspace defined by the constraints, its l_1-norm could be reduced.

To remedy this situation, Levy and Fullagar [1981] propose inequality constraints in place of equality constraints. These inequality constraints require the Fourier components of the solution in the passband to be within some bound from the values determined by Fourier division (cf. (1.15)). We can pose an equivalent optimization problem in terms of singular values and singular vectors for our problem as we have mentioned, and take advantage of the recursive structure. Our approach here is to use the data, i.e., the convolution equation in a penalty term as we have described earlier.

The problem we choose to solve then is:

Find $\mathbf{r}^*$ which minimizes

$$\|\mathbf{r}\|_1 + \lambda \|\mathbf{F}\mathbf{r} - \mathbf{g}\|_2^2. \tag{5.1}$$

The reasons for choosing this cost functional have been discussed in the Introduction. Thus, in (5.1), we measure the deviation of the vector $\mathbf{r}$ from the subspace defined by $\mathbf{F}\mathbf{r} = \mathbf{g}$ in l_2-norm. Hence, λ is a measure of how much weight (faith) one should put on the data. A rough estimate is that λ should be inversely proportional to the square of the assumed l_2-noise in the data. In exchange for the violation of the equality constraints, we can now find a solution with a smaller l_1-norm than the constrained optimum.

In the following examples, we took again the same wavelet as in Fig. 2b and generated two data vectors $\mathbf{g}$ using the impulse response traces $\mathbf{r}$ in Figs. 2a and 3a. We solved the optimization problem in (5.1) for various values of penalty parameter λ. The results are shown in Figs. 5 and 6. Displayed at the bottom of each figure is the generating impulse response. Plotted above these are the solutions of (5.1) for decreasing λ's. Also shown in Figs. 5b and 6b are the profiles $\alpha(s)$ corresponding to the impulse responses in Figs. 5a and 6a. We see that for noisy data as λ is decreased the profiles come closer and closer to piecewise constant curves with less and less major jumps.

The observation to be made is that when λ is large, we get the constrained solution. But as λ is decreased, we put less and less importance on the smaller nonzero components in $\mathbf{r}$. The smaller the λ, the more zero components we will have in the optimum. Until the limiting case when $\lambda = 0$, we get $\mathbf{r} = \mathbf{0}$. However, observe that we get a very good estimate of the unperturbed constrained solution at about $\lambda = 20$. This suggests that a prudent choice of λ for a certain noise level will produce robust estimates of the model impulse response from noisy data.

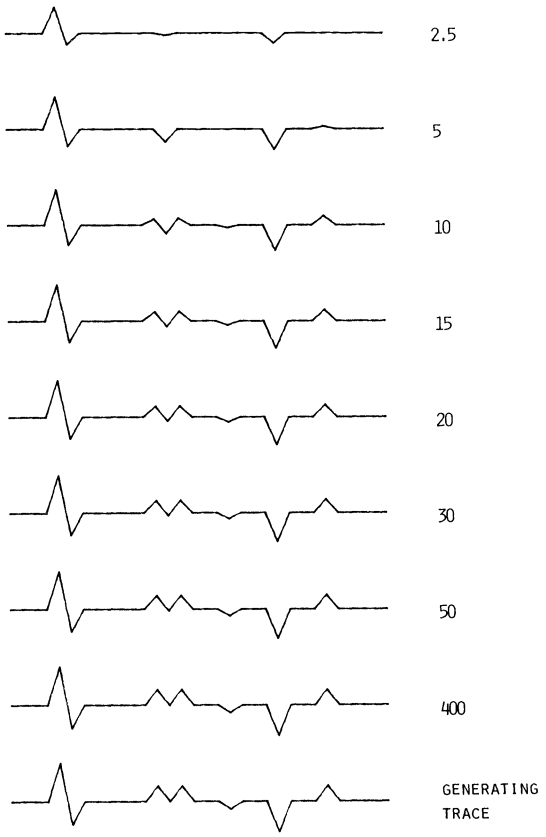


FIG. 5a. *The optima of* $\phi(\mathbf{r}) = \|\mathbf{r}\|_1 + \lambda \|\mathbf{Fr} - \mathbf{g}\|_2^2$ *for decreasing values of the parameter* λ. *The data vectors are* **f** *and* **g** *shown in Figs.* 2b *and* 2c. *The bottom trace is the generating time series* **r**. *Observe that as* λ *is decreased, the smaller spikes disappear.*

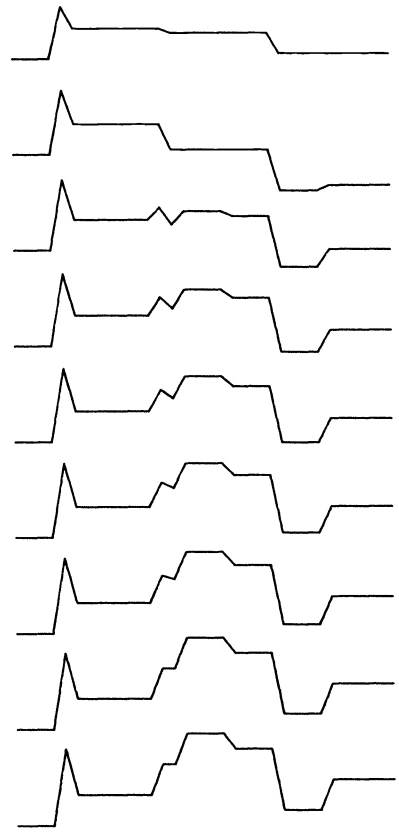


FIG. 5b. *Companion to Fig.* 5a, *plots of* $\alpha(s)$ *as a function of* λ.

6. Optimization algorithm. Recall that our objective is to minimize a cost functional

$$\phi(\mathbf{r}) = \|\mathbf{r}\|_1 + \lambda \|\mathbf{Fr} - \mathbf{g}\|_2^2 \tag{6.1}$$

for **r** and **g** in $\mathbb{R}^m$, and **F** $m \times m$ matrix. The matrix **F** has a null space, so, the l_2-part will be ill-conditioned. In fact, the second term has a geometric interpretation of a long "trough". The l_1-part on the other hand, although nonsmooth, has a simple structure.

If we simply take a modified restricted gradient approach of Bartels, Conn and Sinclair [1978],we will end up zigzagging along the bottom of the trough because of the l_2-part. So, we need to have two strategies:

(i) Move parallel to the null space of **F** until we find a constrained optimum.

(ii) At this point we are transverse to the null space of **F**, and we use a gradient type algorithm.

The above procedure is repeated until convergence is achieved.

The justification for the above strategy comes from the fact that at the constrained optimum, we are already close to the minimum of $\phi(\mathbf{r})$.

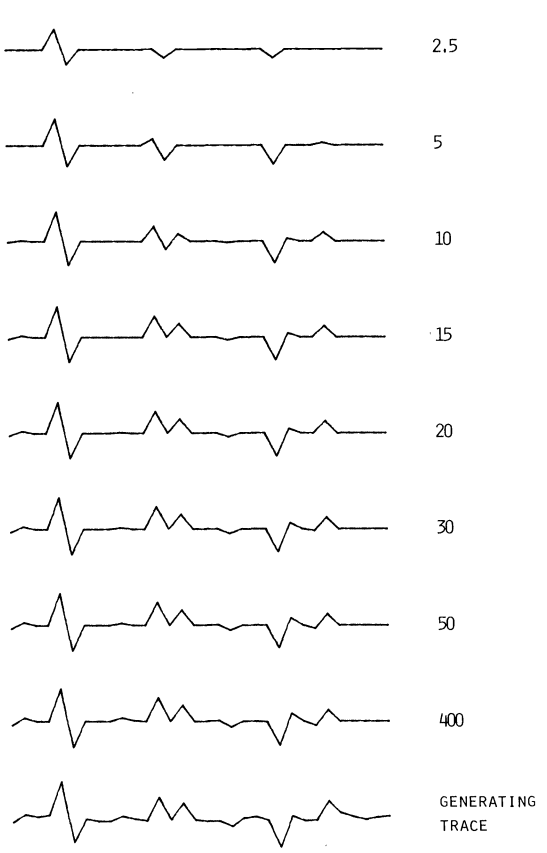


FIG. 6a. *The optima of* $\phi(\mathbf{r}) = \|\mathbf{r}\|_1 + \lambda \|\mathbf{Fr} - \mathbf{g}\|_2^2$ *for decreasing values of* λ. *The data vectors are* $\mathbf{f}$ *and* $\mathbf{g} = \mathbf{Fr}$, *where* $\mathbf{r}$ *is displayed at the bottom (also Fig.* 3a). *Observe that the effects of noise are diminished as* λ *is decreased and sparse spike train solutions are recovered.*

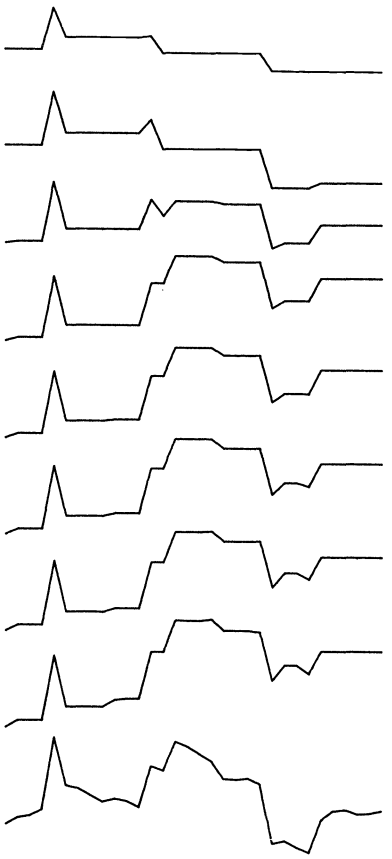


FIG. 6b. *Companion to Fig.* 6a, *plots of* $\alpha(s)$ *as a function of* λ. *Target profile is in Fig.* 5b.

The graph of ϕ is a stratified variety in $\mathbb{R}^{N+L}$:

$$\text{graph } \phi = \bigcup_s V_s$$

where the union ranges over all functions $s: \{1, \cdots, N\} \to \{-1, 0, 1\}$ and

$$V_s = \{(x, \phi(x)): x_i = 0 \text{ if } s(i) = 0, \text{ else } x_i/|x_i| = s(i)\}.$$

The interior $\{(x, t): t > \phi(x)\}$ is convex, so that the infimum of ϕ lies at an extreme point. Because $\mathbf{F}$ has a nontrivial null space, the minimum is guaranteed to lie on a lower-dimensional stratum. Since each V_s is open in its closure, a constrained optimum point lies in its interior. Generically, V_s intersects the constraint set transversely, as do nearby constraint sets. Therefore a step along the gradient $-\nabla\phi$ projected on V_s is possible and results in a decrease in ϕ. One hopes that the angle between the tangent space of V_s and the constraint set $\{\mathbf{Fr} = \text{const.}\}$ is large enough that the constrained (to V_s) optimization problem for ϕ is well conditioned. Then the constrained $\{\mathbf{Fr} = \text{const.}\}$ step presumably accomplishes the same objective as a large number of uncon-

strained gradient steps for ϕ, which tend to oscillate around the "trough", slowly building a constrained step.

In the algorithm explained below, the constrained l_1-search and the projected grad ϕ-search are alternated. To avoid occasional difficulties encountered in projected gradient methods ("dead points"), we intermittently restart the algorithm after a small pseudo-random perturbation.

Let us describe the procedure which we summarize in algorithmic form at the end of the section. Suppose our initial guess is $\mathbf{r}_0$. Now we want to displace $\mathbf{r}_0$ to $\mathbf{r}_0+\mathbf{p}$ such that $\mathbf{Fp}=0$, moving parallel to the null space of $\mathbf{F}$. We write $\mathbf{p}$ in terms of the null vectors as in § 4. We solve:

$$\text{minimize} \qquad \|\mathbf{r}_0+\mathbf{V}_0\mathbf{q}\|_1. \tag{6.2}$$

The optimum is indicated by $\mathbf{r}_1=\mathbf{r}_0+\mathbf{V}_0\mathbf{q}$. In particular, at $\mathbf{r}_1$, we have the index set

$$Z_1=\{i\in Z_1\colon (\mathbf{r}_1)_i=0\}, \tag{6.3}$$

and its complement Z_1^c. To this index set, we associate a subspace of $\mathbb{R}^m$ given by

$$S_1=\{\mathbf{r}\in S\colon (\mathbf{r})_i=0;\ i\in Z\}. \tag{6.4}$$

The projector onto this null space is a matrix which sets the components $i\in Z$, of a vector $\mathbf{r}\in\mathbb{R}^m$ to zero, and is easily found to be

$$\mathbf{P}=\sum_{i\in Z_1}\mathbf{e}_i\mathbf{e}_i^T, \tag{6.5}$$

$\mathbf{e}_i$ is the ith directional unit vector.

Now we go back and try to find the minimum of (6.1) in S, i.e.,

$$\begin{aligned}&\text{minimize} && \phi(\mathbf{r})\\ &\text{subject to} && \mathbf{r}\in S.\end{aligned} \tag{6.6}$$

As we have mentioned, on S, we are transverse to the null space of $\mathbf{F}$, so we expect (6.6) to be better-conditioned than (6.1). We start with $\mathbf{r}=\mathbf{r}_1$, and move $\mathbf{r}_1$ to $\mathbf{r}_1+\mathbf{p}$ where $\mathbf{p}+S$. Suppose $\mathbf{p}$ is small so that

$$\operatorname{sgn}(\mathbf{r}_1+\mathbf{p})_i=\operatorname{sgn}(\mathbf{r}_i)_i \quad \text{for } i\in Z_i. \tag{6.7}$$

Then $\mathbf{p}$ can be found by taking the gradient of $\phi(\mathbf{r})$. The gradient is

$$\mathbf{h}=\sum_{i\in Z_1^c}\operatorname{sgn}(\mathbf{r}_1)_i\mathbf{e}_i+\mathbf{F}^T(\mathbf{F}\mathbf{r}_1-\mathbf{g})+\mathbf{F}^T\mathbf{F}\mathbf{p}. \tag{6.8}$$

To find the search direction $\mathbf{p}$, we set the projection of $\mathbf{h}$ onto S to zero,

$$P\mathbf{h}=\mathbf{0}. \tag{6.9}$$

Along $\mathbf{p}$, we have a one-dimensional optimization problem. We vary α in $\phi(\mathbf{r}+\alpha\mathbf{p})$ until we find its minimum. If the minimum lies at a vertex, i.e., at a point where

$$(\mathbf{r}_1+\alpha\mathbf{p})_j=0, \tag{6.10}$$

we add the index j to Z_1.

We repeat the process by beginning at (6.6) with a new subspace S and index set Z_1. The algorithm arrives at $\mathbf{p}=0$ in finite steps. Let $\mathbf{r}_2$ denote the point where $\mathbf{p}=\mathbf{0}$. At this point, we add a small random vector to $\mathbf{r}_2$, and begin the process again. When we have reached a stable fix point, we terminate the procedure.

We can summarize the optimization scheme in the following algorithm.

(0) initial given $\mathbf{r}_0$($\|\mathbf{Fr}-\mathbf{g}\|$ small)
(1) SVD of $\mathbf{F}$
(2) minimize $\|\mathbf{r}_0+\mathbf{V}_0\mathbf{q}\|_1$ (using Bartels, Conn and Sinclair algorithm) arrive at $\mathbf{r}_1$, index set Z_1 and subspace S
(3) evaluate $\mathbf{h}$ in equation (6.8)
(4) solve for $\mathbf{P}=\sum_{i\in Z_1^c}\beta_i\mathbf{e}_i$ by requiring $\mathbf{Ph}=\mathbf{0}$, if $\mathbf{p}=\mathbf{0}$, go to (7)
(5) search for minimum of $\phi(\mathbf{r}_1+\alpha\mathbf{p})$
  (a) minimum is at some j such that $(\mathbf{r}_1+\alpha\mathbf{p})_j=0$
  (b) else
(6) $\mathbf{r}_1\to\mathbf{r}_1+\alpha\mathbf{p}$
  if (a) $Z_1\to Z_1+\{j\}$
  return to (3)
(7) $\mathbf{r}=\mathbf{r}_2$, add small random vector $\mathbf{n}$: $\mathbf{r}_2+\mathbf{n}$ return to (2).

Our experience with the numerical experiments indicate that this algorithm is efficient, although improvements can be made. No proofs of convergence exist for this algorithm. However, we find repeatedly the same fix points in all the experiments with slightly different initial guesses. Step 7 is normally taken once or twice before a stable fix point is found.

7. Recursive algorithm. In §§ 4 and 5, we treated the small optimization problems that arise when we use the recursive structure and partition the large problem. These small subproblems are described in (3.14), and an algorithm for solving them is presented in § 6. In this section, we describe the complete algorithm, which exploits the nature of the convolution.

We use the notations in § 4. Again, suppose we have a wavelet $\mathbf{f}$ of duration (n). The data vector $\mathbf{\Gamma}$ is of length M. From $\mathbf{f}$, we have an $M\times M$ convolution matrix $\mathbf{\Phi}$, the problem is to find $\mathbf{T}$, also an M-vector, in $\mathbf{\Phi T}=\mathbf{\Gamma}$. The algorithm to solve this large problem is:

(0) from wavelet vector $\mathbf{f}$, construct the submatrices $\mathbf{F}_1$ and $\mathbf{F}_2$, each $m\times m$ $(m>n)$ where

$$(F_1)_{ij}=\begin{cases}0, & 0\geqq i-j+1>m,\\ f_{i-j+1}, & m\geqq i-j+1>0,\end{cases}$$

$$(F_2)_{ij}=\begin{cases}0, & 0\geqq i-j+n+1>m,\\ f_{i-j+n+1}, & m\geqq i-j+n+1>0.\end{cases}$$

See Fig. 1.
(1) partition $\mathbf{\Gamma}$ into m-vectors $\mathbf{g}_i$, $i=1,2\cdots L$, $Lm=M$
(2) for $i=1,2\cdots L$, solve
  Find minimizer $\mathbf{r}_i^*$ of
    $\phi(\mathbf{r}_i)=\|\mathbf{r}_i\|_1+\lambda\|\mathbf{F}_2\mathbf{r}_{i-1}^*-\mathbf{F}_1\mathbf{r}_i-\mathbf{g}_i\|_2^2$
    $(\mathbf{r}_0^*=0)$
Algorithm in § 6
(3) augment $\mathbf{r}_i^*$ to reconstruct $\mathbf{T}$; $\mathbf{r}^T=(\mathbf{r}_1^T|\mathbf{r}_2^T|\cdots)$.

This algorithm accomplishes the task of extracting a sparse spike train from a noise-corrupted data. Moreover, it is efficient because we solve a sequence of small problems (size-m), instead of the entire problem all at once (size-M). If the minimum

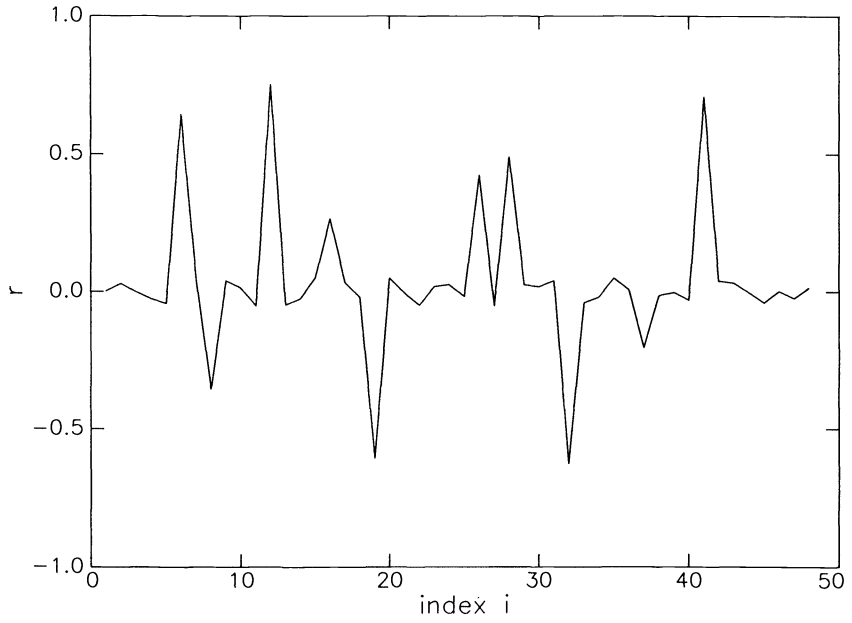


FIG. 7a. *The generating noisy "sparse" spike train.*

of the size-M problem is desired, we claim that our algorithm provides a good initial guess at a very low cost.

We coded the algorithm above and solved a 48-point problem in 2 steps of size 24. The wavelet in our example is shown in Fig. 2b. When noiseless data, generated using the noiseless version of the spike train in Fig. 7a, are used as constraints, we recover the spike train exactly in 2 steps. This exercise verifies our argument in § 3.

More interesting is the case when the generating trace is contaminated by noise. We display the data Γ in Fig. 7b. The target profile, the primitive of $\mathbf{T}$, is shown in dots in Fig. 7c. This is our solution when the data is noise free and used as constraints. In our example, Φ is a 48×48 matrix which has 14 singular values of less than 0.005. We took the noisy data as penalty with $\lambda = 20$, and used our algorithm to solve first the size-48 problem (taking $m = 48$), and second, two size-24 problems (taking $m = 24$). The results are displayed in Fig. 7c for comparison. Shown in the dashed line is the result of solving the large problem, and in the solid line is the result of solving the problem in two recursions. In this experiment, the recursive algorithm produces a solution which is very close to the solution of the large optimization.

In terms of computation time, it took 280 seconds to solve the 48-pt problem (on a Prime 750), and 31 seconds to solve each of the 24-pt problems. This algorithm allows us to envision solving 1024-pt problems, which are typical in seismic experiments. As a comparison, we also display in Fig. 7c the primitive of the least length solution of $\Phi\mathbf{T} = \Gamma$ (shown in dash-dot).

8. Conclusions. We have proposed a cost functional to be used in resolving the ambiguity in the linear inversion (deconvolution) of band-limited data for seismic exploration purposes. The method represents an alternative to those proposed earlier by other authors. It uses as a measure of data-fit the l_2-norm, which can be shown to be advantageous in certain situations. The choice of the parameter to weigh the data is left to the user of the algorithm, and hence the method is suited for interactive data

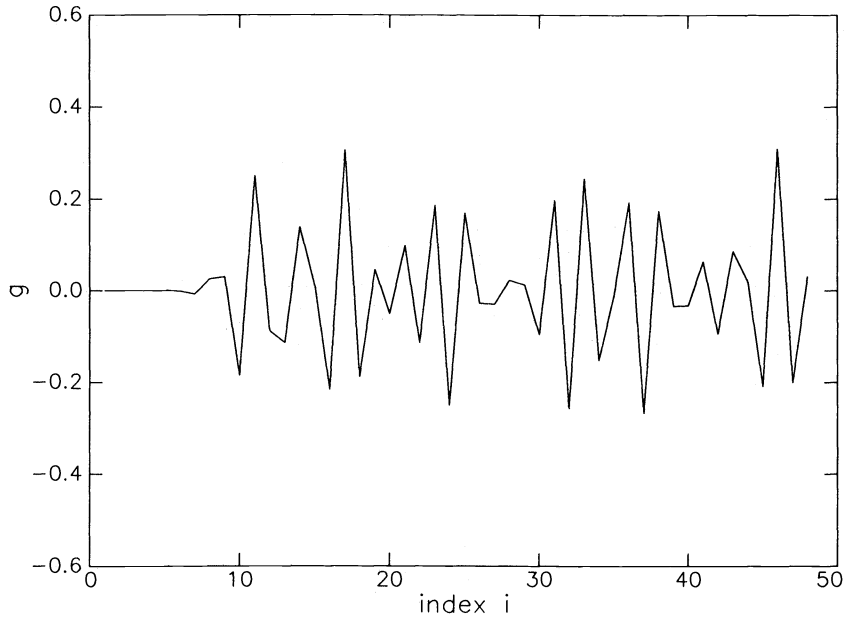


FIG. 7b. *Noisy data from reflectivity in Fig.* 7a. *Noise level is* 15 *percent in* l_2*-norm.*

interpretations. We have also proposed to take advantage of the recursive structure of convolution equations, and show that solving a sequence of small problems, which is advantageous in terms of computational requirement, produces a good estimate of the solution to the large problem. An algorithm for the optimization process has been constructed for this purpose.

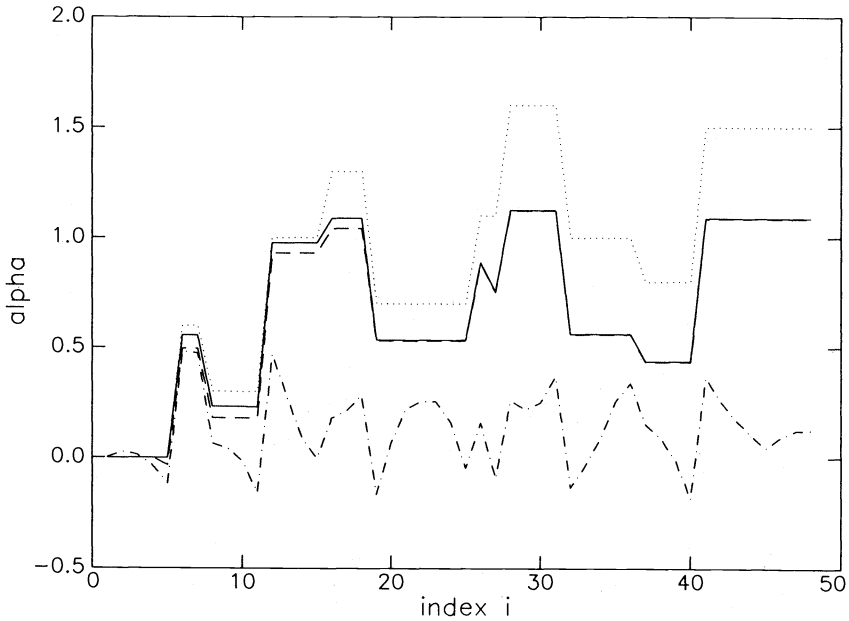


FIG. 7c. *Shown in dots is the target profile* $\alpha(s)$ *corresponding to noiseless version of reflectivity in Fig.* 7a. *The solution to the* 48*-point optimization is given in dashes, and to the* 2*-step recursion in solid. For comparison, the profile corresponding to the* SVD*-inverse is shown in dash-dots. Observe that in-range (pass band) noise effects the determination of out-of-range (low frequency) components of profile.*

Numerical results are presented to display various features of the problem. It must be noted here that although the requirement that the solution of the deconvolution problem has the least feature (small l_1-norm), the trend of the profile is still constructed erroneously when the data is noisy. This indicates that information about the trend of the profile, when the low frequency components of the data are missing, must be supplied by incorporating other a priori information.

Acknowledgments. The authors would like to express their grateful appreciation to Tom Coleman of the Computer Science Department at Cornell University for his helpful suggestions in solving our optimization problems. We also thank Horst Schwetlick of the Technical University of Berlin for many useful discussions on deconvolution problems in the real world.

REFERENCES

R. H. BARTELS, A. R. CONN AND J. W. SINCLAIR [1978], *Minimization techniques for piecewise differentiable functions: the l_1-solution to an overdetermined linear system*, SIAM J. Numer. Anal., 15 (2), pp. 224–241.

J. F. CLAERBOUT AND F. MUIR [1973], *Robust modeling with erratic data*, Geophysics, 38 (5), pp. 826–844.

G. H. GOLUB AND C. F. VAN LOAN [1983], *Matrix computations*, The Johns Hopkins University Press, Baltimore, MD.

S. LEVY AND P. K. FULLAGAR [1981], *Reconstruction of a sparse spike train from a portion of its spectrum and application to high-resolution deconvolution*, Geophysics, 46 (9), pp. 1235–1243.

D. W. OLDENBURG [1984], *Inversion of band limited reflection seismograms*, in Inverse Problems of Acoustic and Elastic Waves, F. Santosa et al., editors, Society for Industrial and Applied Mathematics, Philadelphia, PA.

D. W. OLDENBURG, T. SCHEUER AND S. LEVY [1983], *Recovery of the acoustic impedance from reflection seismograms*, Geophysics, 48 (10), pp. 1318–1337.

F. SANTOSA AND W. W. SYMES [1983], *Inversion of impedance profile from band-limited data*, in Digest, International Geoscience and Remote Sensing Symposium, 1983.

G. W. STEWART [1973], *Introduction to Matrix Computations*, Academic Press, New York.

W. W. SYMES [1983], *Impedance profile inversion via the first transport equation*, J. Math. Anal. Appl., 94 (2), pp. 435–453.

H. L. TAYLOR, S. C. BANKS AND J. F. MCCOY [1979], *Deconvolution with the l_1 norm*, Geophysics, 44 (1), pp. 39–52.

SIAM J. SCI. STAT. COMPUT.
Vol. 7, No. 4, October 1986

016

INVERSE SCATTERING WITH NOISY DATA*

A. M. BRUCKSTEIN†, I. KOLTRACHT† AND T. KAILATH†

Abstract. In this paper we analyze error propagation in layer-peeling inversion methods. A bound for the error in recovering the reflection coefficient at a certain depth is given in terms of the *estimated* reflection coefficients. The error propagation results are then used to discuss some practical inversion algorithms that exploit available prior information on the reflection coefficient sequence.

Key words. inverse scattering, recursive profile identification, error propagation analysis

AMS(MOS) subject classifications. 35L05, 65G05

1. Introduction. Inverse scattering problems arise in a wide range of fields such as geophysics, transmission-line synthesis, speech research, physics, digital filter design, biomedical research. It is therefore not surprising that a great deal of effort has gone into the study of such problems. We shall study the inverse problem that calls for the identification of a layered wave propagation medium, from a set of scattering data, usually the recorded response of the medium to some probing signals; this problem is a prototype of a host of one-dimensional inverse scattering problems.

The one-dimensional problem is by now well understood, and several approaches for its solution have been devised [1]–[5]. The most recently studied approaches are the so-called direct dynamic deconvolution or layer-peeling or differential methods, which exploit in a straightforward manner the underlying model of the scattering medium. The recursive solutions provided by layer-peeling approaches use causality to identify and conceptually remove the effect of one medium layer at a time [6]–[12]. While the resulting algorithms are fast and numerically stable [13]–[15], a basic assumption in deriving them is that the scattering data are noise free. If the data are corrupted by noise, the sequential nature of the identification process seems to predict a catastrophic accumulation of errors; because of this, geophysicists, the first ones to derive layer-peeling methods, refrained from using them and returned to indirect methods, based on solving integral/matrix equations. It was a common belief that such approaches have an implicit noise smoothing effect. However, given infinite precision computation, it is obvious that the integral equations-based methods, when applied to noisy data, will yield exactly the same results as the layer-peeling procedures do. In this work we will show that the noisy-data situation is not hopeless. In fact, a careful analysis of error propagation in the inversion algorithms reveals that, in many practical cases, the error accumulation is slow enough to enable reliable identification up to a reasonable depth into the medium. Furthermore, additional assumptions on the medium parametrization should help in getting better estimates. For example, in some geophysical problems, in which the discretized medium is parametrized by the so-called "reflection coefficients," it may be known in advance that the reflection coefficients are close to zero most of the time, and only at certain depths, where the earth properties change drastically, will the local reflection coefficients assume significant values [16]–[18]; as will be shown in this paper, this a priori information can be used to advantage

* Received by the editors August 14, 1984, and in revised form August 9, 1985. This work was supported in part by the Air Force Office of Scientific Research, Air Force Systems Command under contract AFOSR-83-0228 and by the U.S. Army Research Office, under contract DAAG 29-83-K-0028.

† Information Systems Laboratory, Stanford University, Stanford, California 94305.

(the same point has recently been made by Arsene et al. [9], though on more qualitative grounds).

In § 2 we discuss the general layer-peeling inversion process. The next section provides the basic conditioning and error accumulation results. We show that, if the data are perturbed by errors bounded in magnitude by ε, then the error in estimating the pth reflection coefficient is bounded by

$$|k_p - \hat{k}_p| \leqq M\varepsilon \prod_1^{p-1} \frac{1+|\hat{k}_i|}{1-|\hat{k}_i|} + O(\varepsilon^2) \tag{1.1}$$

where $\hat{k}_i$'s are previous reflection coefficient estimates, and M is a constant depending on the type of scattering data given. This is an important result, since it provides error bounds in terms of the estimated medium parameters, and therefore it can be readily used to make decisions about the usefulness of the results as the inversion algorithm proceeds. Then § 4 presents simulation results confirming the error propagation results, and discusses several practical ways to implement layer-peeling algorithms with noisy data that also exploit prior information on the medium.

2. Inverse scattering via layer-peeling methods. Suppose we are given that the signal propagation through a layered medium is modeled by the following set of waveform *transfer* equations (Fig. 1)

$$\begin{bmatrix} U(n+1,t) \\ V(n+1,t) \end{bmatrix} = \Theta(k_{n+1}) \begin{bmatrix} \Delta & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} U(n,t) \\ V(n,t) \end{bmatrix}, \tag{2.1}$$

where Δ stands for a time delay operator defined by $\Delta f(t) = f(t-1)$ and $\Theta(k)$ is a static "gain" matrix parametrized by a number k. Such propagation equations arise naturally in geophysics, transmission-line theory, optical propagation and linear prediction with lattice filters (see e.g. [3]-[8], [16]-[22]). In the above model, if we are given the parametric form of $\Theta(k)$, the sequence of parameters $\{k_n\}$ and the initial conditions $U(0,t)$ and $V(0,t)$, the propagating signals are completely determined. Thus it is rather straightforward to solve the *direct scattering* problem. Suppose however that we are given the same data, and we make the causality assumption (see Fig. 1) that

$$U(n,t) = 0 \quad \text{for } t < n. \tag{2.2}$$

Then with the parametric model (2.1) for the signal transfer, it turns out that the given data in fact determine the medium, i.e. the sequence $\{k_n\}$, under very general conditions. The process of recovering the medium from the signals at $n=0$ specifies an *inverse scattering* algorithm.

To see this, let us first consider an equivalent *scattering representation* of the propagation model. The scattering representation relates $U(n+1,t)$ and $V(n,t)$ to

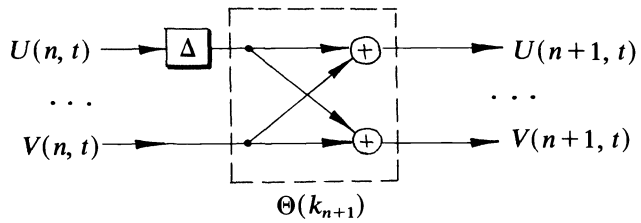


FIG. 1. *The transmission model of the scattering medium.*

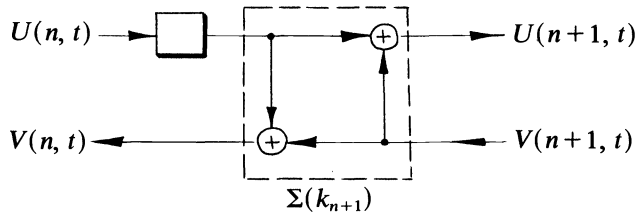


FIG. 2. *Equivalent wave-scattering representation of the medium.*

$U(n, t)$ and $V(n+1, t)$ (see Fig. 2). The equations describing the scattering model are

$$\begin{bmatrix} U(n+1,t) \\ V(n,t) \end{bmatrix} = \Sigma(k_{n+1}) \begin{bmatrix} \Delta & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} U(n,t) \\ V(n+1,t) \end{bmatrix}. \tag{2.3}$$

In this domain we think of $U(0, t)$ as a probing signal sent into a quiescent (zero initial conditions) medium and of $V(0, t)$ as the causally returning response. The relationship between the scattering and transfer representation of the gain matrix that describes the interaction between U and V can easily be recognized to be the following "Mason exchange rule." If the transfer representation is

$$\Theta(k) = \begin{bmatrix} \theta_{11}(k) & \theta_{21}(k) \\ \theta_{21}(k) & \theta_{22}(k) \end{bmatrix}, \tag{2.4}$$

then the corresponding scattering representation

$$\Sigma(k) = \begin{bmatrix} \sigma_{11}(k) & \sigma_{12}(k) \\ \sigma_{21}(k) & \sigma_{22}(k) \end{bmatrix} = \begin{bmatrix} \theta_{11}(k) - \theta_{12}(k)\theta_{22}^{-1}(k)\theta_{21}(k) & \theta_{12}(k)\theta_{22}^{-1}(k) \\ -\theta_{22}^{-1}(k)\theta_{21}(k) & \theta_{22}^{-1}(k) \end{bmatrix}. \tag{2.5}$$

In the development that follows we shall assume that the propagation models are lossless, i.e., energy is conserved when the right and left propagating waves pass through the layers of the medium. In this case the transfer and scattering gain matrices have the form

$$\Theta(k) = \frac{1}{(1-k^2)^{-1/2}} \begin{bmatrix} 1 & -k \\ -k & 1 \end{bmatrix} \tag{2.6}$$

and

$$\Sigma(k) = \begin{bmatrix} (1-k^2)^{-1/2} & -k \\ k & (1-k^2)^{-1/2} \end{bmatrix}. \tag{2.7}$$

Note that the lossless propagation model is entirely parametrized in terms of the local "left reflection coefficient" of the scattering representation, i.e., that $k = \sigma_{21}(k)$. This observation underlies a very simple procedure for the recovery of the medium, given the data $U(0, t)$ and $V(0, t)$.

Let us show that, given $U(0, t)$ and $V(0, t)$ we can recursively determine the parameters of the medium that *causally* generated these scattering data. Indeed, k_1 is simply given by

$$k_1 = \frac{V(0,1)}{U(0,0)}, \tag{2.8}$$

since no returns from the initially quiescent medium could influence the first nonzero lag of the response $V(0, t)$. But, once the first reflection coefficient is known we can use the exchange rule (2.5) to determine $\Theta(k_1)$ and then equation (2.1) to "forward propagate" the data and obtain $U(1, t)$ and $V(1, t)$. Now, we are facing a similar

situation as before, except that the medium to be identified starts at depth 1. Thus we are able to recover k_2 next and propagate the scattering data further, and so on.

Algorithms that successively identify and then effectively remove or "peel off" layer after layer from the given medium by operating on the given data sequences are called, for historical reasons, Schur-type algorithms [9]-[12].

THE SCHUR/LAYER-PEELING ALGORITHM.

(a) initialize vectors **U**, **V** with scattering data

$$U=[u_0, u_1, \cdots] \leftarrow [U(0,0), U(0,1), \cdots]$$
$$V=[v_0, v_1, \cdots] \leftarrow [V(0,0), V(0,1), \cdots]$$

(b) left shift **V** (delay **U** w.r.t. **V** by one time unit)

$$V \leftarrow [v_1, v_2, \cdots]$$

(c) compute the reflection coefficient

$$k=\frac{v_0}{u_0}$$

(d) propagate **U**, **V** through $\Theta(k)$

$$\begin{bmatrix} U \\ V \end{bmatrix} \leftarrow \Theta(k) \begin{bmatrix} U \\ V \end{bmatrix}$$

(e) goto step (b).

The complexity of this algorithm is $O(N^2)$, i.e., the count of elementary operations (multiplications and additions) that are required to obtain the first N medium parameters is proportional to N^2.

We note that the above layer-peeling algorithm will solve the inverse scattering problem for *any* propagation model of the type (2.1) that is entirely parametrized in terms of the local left reflection coefficient ($k=\sigma_{21}(k)$) of the scattering representation. If the medium model is given via the scattering representation, we only have to assume that the functional dependence of the $\sigma_{i,j}$'s on k are given, and that $\sigma_{22}(k)$ is always invertible.

3. Conditioning and noise propagation. Throughout this section we shall assume that the computations are all performed with arbitrary precision and thus analyze only the propagating effect of perturbations in the *scattering data.* To evaluate the cumulation of errors in a layer-peeling algorithm it is helpful to rewrite the algorithm as a sequence of matrix operations acting on the data vector pair $\{U, V\}$. Suppose we are given N data points for U and V. Then the Schur, or layer-peeling, procedure for recovering $N-1$ layers of the medium can be conveniently expressed as a product of operators **Q** and $\mathbf{K}(k_i)$ acting on the data vector $[U, V]$, where the operator

$$\mathbf{Q}=\begin{bmatrix} I & 0 \\ 0 & Z \end{bmatrix} \quad \text{with } Z=\begin{bmatrix} 0 & 1 & 0 & \cdot & \cdot & 0 \\ 0 & 0 & 1 & \cdot & \cdot & 0 \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ 0 & 0 & 0 & \cdot & \cdot & 1 \\ 1 & 0 & 0 & \cdot & \cdot & 0 \end{bmatrix} \tag{3.1}$$

describes a cyclic shifting of the V sequence in the vector $[U, V]$, and the gain matrix

$$\mathbf{K}(k)=\begin{bmatrix}\theta_{11}(k)I & \theta_{12}(k)I\\ \theta_{21}(k)I & \theta_{22}(k)I\end{bmatrix}=\Theta(k)\otimes\begin{bmatrix}I & I\\ I & I\end{bmatrix} \tag{3.2}$$

generates the vectors that would result by passing the sequences U and V through a transition layer $\Theta(k)$ as given by (2.6).

To determine the pth reflection coefficient we have to apply $\mathbf{S}_p$ defined as

$$\mathbf{S}_p=\mathbf{K}(k_{p-1})\mathbf{Q}\mathbf{K}(k_{p-2})\mathbf{Q}\cdots\mathbf{K}(k_1)\mathbf{Q} \tag{3.3}$$

to the scattering data pair $\{U, V\}$, and form k_p as the ratio of v_1 by u_0 that appear as appropriate entries of the resulting vector pair. In forming $\mathbf{S}_p$ we have to use the estimates of the reflection coefficients $k_1, k_2, \cdots, k_{p-1}$. Suppose now that the first $p-1$ reflection coefficients are known precisely, i.e., that $\mathbf{S}_p$ is given. Then recovering k_p involves first computing

$$\begin{bmatrix}U_p^T\\ V_p^T\end{bmatrix}=\mathrm{S}_p\begin{bmatrix}U^T\\ V^T\end{bmatrix}=\mathbf{S}_p\mathbf{D} \tag{3.4}$$

and then performing one more division. It is well-known that if we have a "perturbed" operator $\mathbf{S}_p=\varepsilon F_p$ ($\in>0$) acting on a data set "perturbed" by εf_p, then the relative error in the resulting vector pair $\{U_p, V_p\}$ is bounded by the number

$$\varepsilon c(\mathbf{S}_p)\left[\frac{\|f_p\|}{\|\mathbf{D}\|}+\frac{\|\mathbf{F}_p\|}{\|\mathbf{S}_p\|}\right] \tag{3.5}$$

where $c(\mathbf{S}_p)=\|\mathbf{S}_p\|\,\|\mathbf{S}_p^{-1}\|$ is the *condition number* of the operator $\mathbf{S}_p$ (see e.g. [23]). In our case this number can be bounded (using $\|\cdot\|_\infty$ norms) as follows. We have for $\|S_p\|$ that

$$\|S_p\|\leqq\prod_1^{p-1}\left(\max\{|\theta_{11}(k_i)|+|\theta_{12}(k_i)|, |\theta_{21}(k_i)|+|\theta_{22}(k_i)|\}\right) \tag{3.6}$$

and a similar expression holds for $\|S_p^{-1}\|$, with θ_{ij}'s replaced by the entries of the inverse of Θ. For the symmetric and lossless case, where $\Theta(k)$ is given by (2.6), we have $\Theta^{-1}(k)=\Theta(-k)$ providing the bound

$$c(\mathbf{S}_p)\leqq\prod_1^{p-1}\frac{1+|k_i|}{1-|k_i|}. \tag{3.7}$$

Note that the above bound on the condition number is expressed in terms of the *unknown* reflection coefficient sequence $\{k_1, k_2, \cdots, k_{p-1}\}$. Furthermore, we can show that the expression (3.7) also bounds the error of getting k_p, since by modifying the normalization of the transfer matrices in a trivial way one can force the leading term in U_p, u_0, to always be 1, thus setting k_p equal to the corresponding v_1, one of the entries of V_p.

The number $c(\mathbf{S}_p)$ has the following significance: even with perfect knowledge of all previous reflection coefficients the relative error in computing k_p may be $c(\mathbf{S}_p)$ times the relative perturbation in the data. We may therefore define that a portion of a given medium has an *inherent conditioning* determined by the reflection coefficients via (3.7). If this value is high, there can be no guarantee for a reliable identification of the medium parametrization from noisy data by *any procedure.* The conditioning of the problem will be high if reflection coefficients are close to one in absolute value. Note also the important fact that the conditioning formula (3.7) is physically meaningful,

since a reflection coefficient close to one would imply that most of the energy of the probing wave is reflected at that point, leaving very little energy for probing farther. In this case we can not expect to get reliable results beyond the depth at which such high reflection occurs and this is exactly the behavior predicted by the condition number formula. Fortunately, in many practical cases we have some prior information regarding the domains in which the medium parameters lie. If, from this knowledge, we can conclude that the problem is well-conditioned we may proceed and search for algorithms for its solution and a numerically stable procedure will yield answers in which we can believe.

Expression (3.7) was obtained by Cybenko as the condition number of a symmetric Toeplitz matrix parametrized by reflection coefficients [15]. The layer-peeling process, when applied to a particular type of scattering data, is in fact closely related to the Levinson recursions that provide the factorization of the inverse of a given Toeplitz matrix (see Appendix).

In practice we are given the ε-perturbed data and suppose we apply the layer-peeling algorithm to recover estimates of the reflection coefficients, $\hat{k}_i$. The question arises: can we, from the estimates obtained, infer the conditioning of the medium, assuming the algorithm is propagated with infinite precision? If this is possible then after obtaining the results we can decide how meaningful they are. We shall see that, for the symmetric and lossless case, the numbers

$$\prod_{1}^{p-1} \frac{1+|\hat{k}_i|}{1-|\hat{k}_i|} \tag{3.8}$$

are also "amplification factors" that measure the error accumulation with depth when estimating reflection coefficients from noisy data. This is an important result showing that a straightforward estimate of the medium conditioning also provides a measure of the actual performance of the inverse scattering algorithm.

In order to prove this result we have to rely on a particular set of scattering data that often arises in practical situations. It is the data set defined as follows:

$$U(0,0)=1 \quad \text{and} \quad V(0,t)=U(0,t)=r_t \quad \text{for } \textit{all } t>0. \tag{3.9}$$

This set of data corresponds to a special way of probing the medium: a unit impulse is launched into it and the returns are reflected back into the medium. In geophysics this type of data corresponds to the so-called reflection or "marine" seismogram [2], [5]. In this case it can be shown that the inverse scattering process is equivalent to solving a nested set of matrix equations, with positive definite and symmetric matrices having Toeplitz structure [3], [5], [12]. This structure yields recursive solutions that enable a rather straightforward error propagation analysis. The main result of this analysis is the following (see Appendix for the proof).

Suppose that the scattering data is ε-perturbed, i.e., that we are given the numbers

$$\hat{r}_t = r_t + \varepsilon_i \quad \textit{with } |\varepsilon_i| < \varepsilon \tag{3.10}$$

and the inversion process yields a sequence of reflection coefficient estimates $\hat{k}_i$ that are all bounded by one in absolute value. Then the error in evaluating the pth reflection coefficient is bounded by

$$|k_p - \hat{k}_p| \leqq 2\varepsilon \prod_{1}^{p-1} \frac{1+|k_i|}{1-|k_i|} + O(\varepsilon^2). \tag{3.11}$$

The above result shows that the medium conditioning indeed provides an estimate of the error accumulation; however, a more surprising fact follows from it. By symmetry, we can regard the unknown "true scattering data" as an ε-perturbation of the noisy data, and then we have that

$$|k_p - \hat{k}_p| \leq 2\varepsilon \prod_1^{p-1} \frac{1+|\hat{k}_i|}{1-|\hat{k}_i|} + O(\varepsilon^2). \tag{3.12}$$

This is a remarkable result, providing error bounds in terms of quantities obtained while propagating the layer-peeling algorithm.

It would be important to know whether similar results hold for the general case, in which the returning response of the medium is measured with ε accuracy. It turns out that if $V(0, t)$ is corrupted by noise bounded in absolute value by ε then the above error bounds indeed hold within an amplification factor $M(U, V)$ that can be evaluated (see Appendix). It is given by

$$M(U, V) = \frac{\|\mathbf{U}^{-1}\|}{(1-\|\mathbf{U}^{-1}\mathbf{V}\|)^2} \tag{3.13}$$

where $\mathbf{U}$ and $\mathbf{V}$ are lower triangular Toeplitz matrices with first columns determined by the scattering data sequences U and V respectively. We note, however, that in all simulations performed, the experimentally obtained "amplification" factor was always unity (see § 4).

To summarize these results, we have obtained evaluations on the rate of error growth with depth in a layer-peeling algorithm, in terms of the reflection coefficient estimates that are obtained. Note that the error accumulation is a result of noisy data only, and it is assumed that no numerical errors are introduced by finite-precision arithmetic.

4. Experimental results. The previous sections described the recursive inverse scattering procedure and, for the symmetric lossless case, also provided error propagation bounds when the given data are perturbed within a priori known bounds. This section presents some simulation results supporting the theoretical error-propagation analysis while also indicating that it yields rather pessimistic error bounds in many practical cases. Then the error propagation results are used together with simple ways to incorporate some available prior information on the reflection coefficient sequence to derive practical inversion algorithms. Those exhibit better medium recovery than a straightforward application of layer-peeling to the noisy data and, for moderate noise levels, are similar in performance to the continuous-case maximum likelihood methods that were proposed by Mendel and his coworkers [18], [24].

Error propagation results. We tested error propagation in layer-peeling algorithms as follows. First the scattering data were generated by passing a given probing sequence U through a 200-layer medium, and obtaining the medium response sequence V. Then pseudo-random white noise sequences with either uniform or Gaussian distribution were added to the medium response V and the inversion methods were applied to the noisy data. The error in recovering the reflection coefficients was computed and the logarithm of the mean absolute error over 50 runs was plotted as a function of depth (from 1 to 200). The underlying medium for the error propagation results was set to have one of the following reflection coefficient profiles:

(a) a step function, i.e., $k_i = B$ for $i < 100$ and $k_i = 2B$ for $i > 100$, with $B = 0.01$, 0.02, 0.03, 0.04, 0.05;

(b) a "randomly modulated" step function, i.e., the previous profile multiplied by a Bernoulli (1/2) sequence of ± 1's;

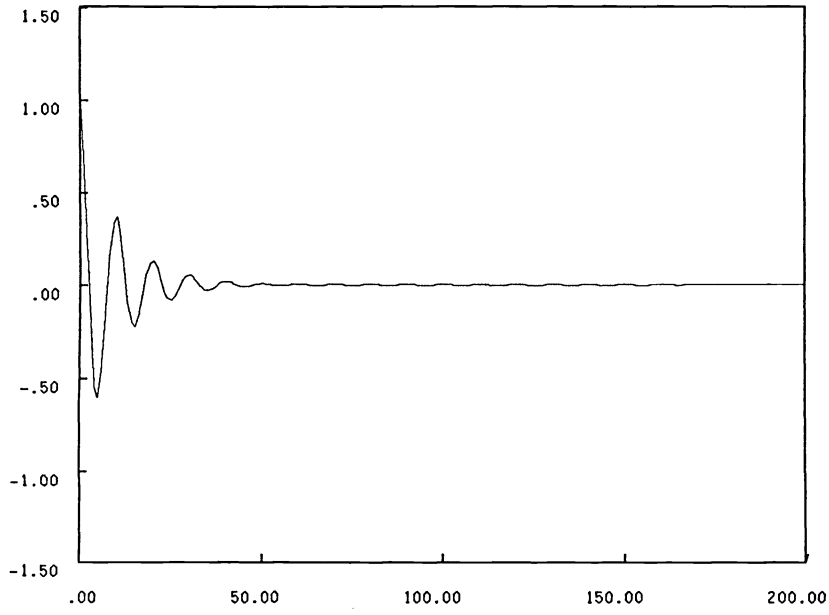


FIG. 3. *A probing sequence U used in numerical experiments.*

(c) a sequence of reflection coefficients uniformly distributed between $(0, B)$ for $i<100$ and in the range $(0, 2B)$ for $i>100$, for $B=0.05, 0.1, 0.15, 0.2, 0.25$;
(d) a sequence of reflection coefficients uniformly distributed between $(-B, B)$ for $i<100$ and in the range $(-2B, 2B)$ for $i>100$, for $B=0.05, 0.1, 0.15, 0.2, 0.25$.

In all these cases a "noise-free" run was also performed to check the propagation of the round-off errors. The given probing sequence U was either a unit impulse at $t=0$ or the function plotted in Fig. 3. In some cases, however we also tested reconstruction with noisy reflection seismogram data as in (3.10). The results for the first type of profile are given in Fig. 4. We can conclude that the error propagation is, in this case, very well described by the formula

$$|k_p - \hat{k}_p| = \varepsilon c(\mathbf{S}_p) = \varepsilon \left(\frac{1+a}{1-a}\right)^p \tag{4.1}$$

for both general scattering data and the reflection seismogram or "marine" (Toeplitz) case. For the cases of the randomly modulated step profiles, however, we see in Fig. 5 that the error bounds are extremely pessimistic. The error does increase exponentially with depth, although at a much slower rate than in the previous case. This is quite important to note since the error propagation bound here is the same as for the nonmodulated profile. From this experimental result one can conclude that the slope of error increase on the logarithmic scale is, in most practical cases not even closely as steep as predicted by the formula (4.1). In case the profile consists of uniformly distributed reflection coefficients in the ranges $(0, B)$ or $(-B, B)$, the predicted average slope is given by

$$E\left[\frac{1+x}{1-x}\right] = \ln \frac{1}{e(1-B)^{2/B}}. \tag{4.2}$$

A comparison of experimental error accumulation obtained for various values of B,

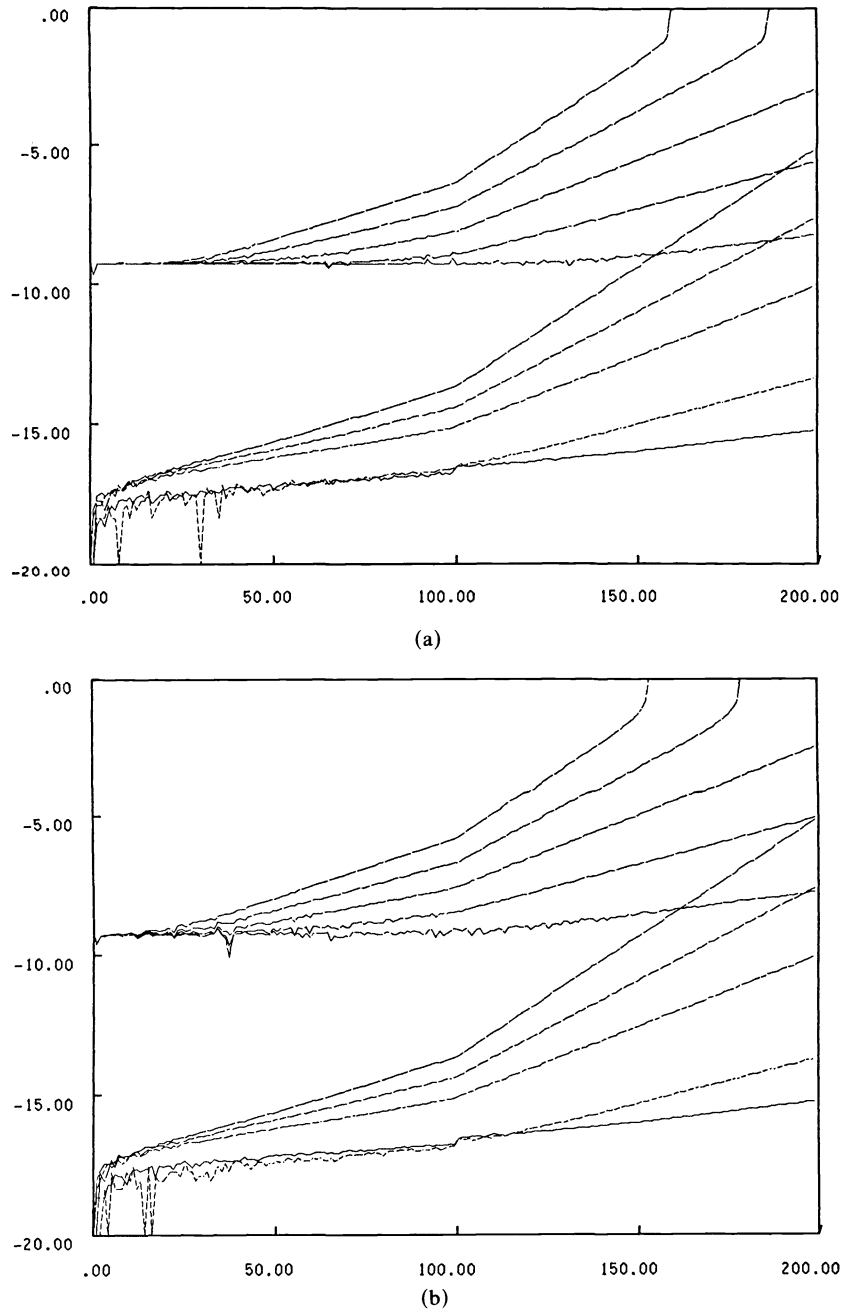


FIG. 4. *Error propagation results for step k-profiles.* (a) *Unit impulse probing sequence.* (b) *The marine, or reflection probing, case. Lower set of curves shows error propagation due to roundoff, the upper set describes identification error accumulation with added uniform noise of span* ($\pm 10^{-9}$).

when propagating the inverse scattering algorithm is given in Fig. 6. From these results we can see that, again, the bounds are good for the all positive reflection coefficients but are extremely pessimistic for the case in which they assume both positive and negative values.

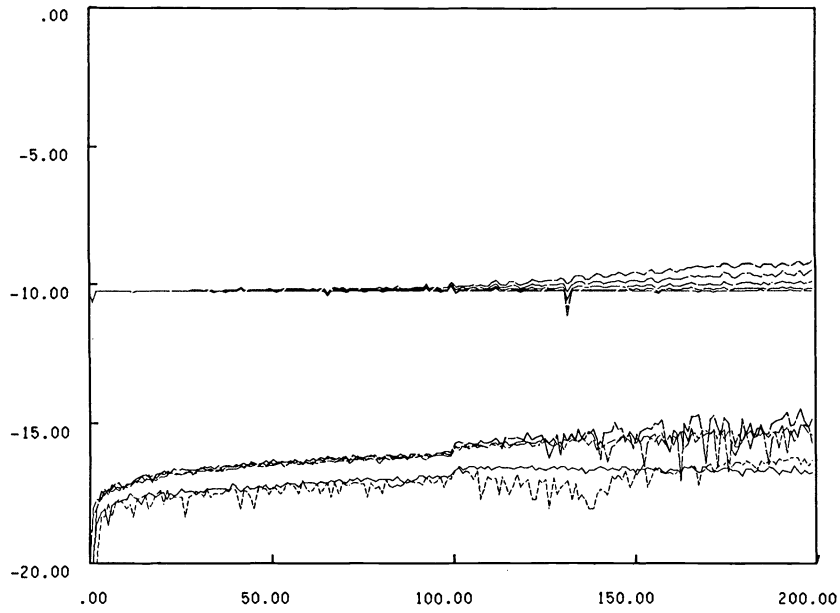


FIG. 5. *Error propagation results for randomly modulated step k-profiles. Note the significant improvement in error accumulation rate.*

Incorporating prior information. In many cases of practical importance, such as reconstructing the impedance profile Z_x of a layered-earth structure from acoustic sounding experiments [16]-[18], we may have some prior information about the medium to be identified. In the case of impedance reconstruction, the continuous propagation equations are discretized and provide a lossless discrete model as described by (2.1). The connection between the reflection coefficients and the (discretized) impedance profile is given by the formula

$$k_{n+1} = \frac{Z_{n+1} - Z_n}{Z_{n+1} + Z_n} \tag{4.3}$$

which immediately provides

$$Z_n = \frac{1+k_n}{1-k_n} Z_{n-1} = \cdots = \prod_n^1 \frac{1+k_i}{1-k_i}. \tag{4.4}$$

Here, Z_n corresponds to the impedance of the nth layer obtained after discretization of the continuous propagation mode (Z_0 is assumed to be 1). For slowly varying impedance profiles and a fine discretization, we have that all of the reflection coefficients will be small by virtue of the formula (4.3). Thus the problem will be quite well conditioned in this case. Suppose, however, that the error bound for a certain discretized impedance profile becomes large for some $n = N$. The effective depth to which the inversion procedure will work is therefore $N\delta x$ where δx is the discretization step. By increasing the spacing in the discretization we will make the reflection coefficients larger and therefore the critical point for the error accumulation will be a smaller value of n. This shows that given a continuous profile there is an "inherent depth" in x to which one can expect good reconstruction with noisy data. This depth will be a function of the profile itself and could be defined as the conditioning of the inversion problem. Comparing the formula for the conditioning of an inversion problem expressed in

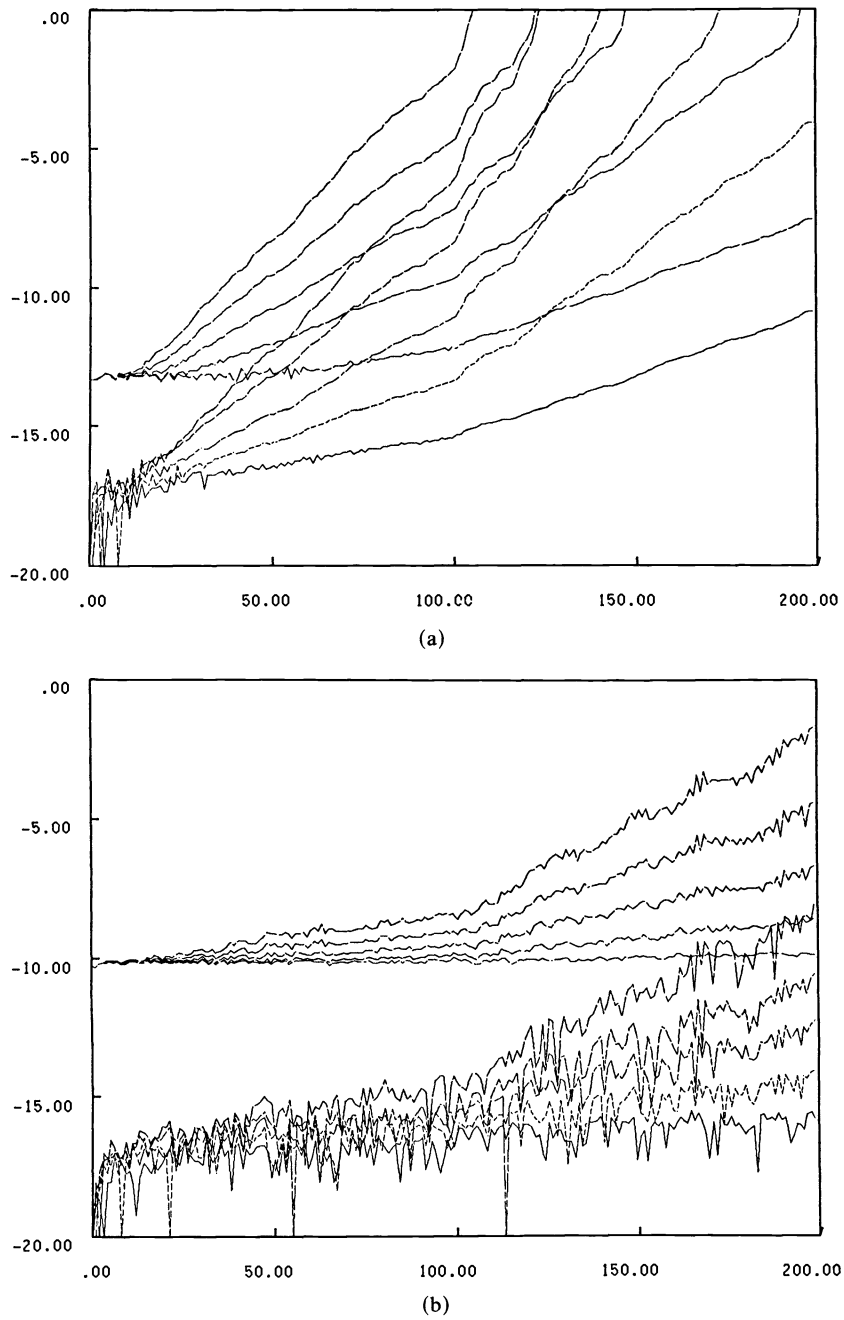


FIG. 6. *Error propagation results for uniformly distributed k-profiles.* (a) *Positive reflection coefficients only: equation* (3.2) *holds.* (b) *Positive and negative reflection coefficients: the error accumulation is much slower than in previous case.*

terms of reflection coefficients, (3.7), with the expression (4.4) for the impedance in terms of the same variables we realize that, for monotone increasing impedance profiles, the impedance itself provides the conditioning. However, when a decrease in impedance occurs the condition number will *increase* as seen from (3.7).

The type of prior information that we can have in a geophysical problem is, for example, that the medium has a relatively small number of layers, each with roughly constant impedance. Also we might have some information regarding the domain to which the impedance can belong, yielding upper bounds (lower than 1) on the absolute values of the reflection coefficient. A fine discretization will result in a reflection coefficient sequence that is practically zero everywhere, except at the jumps between layers. Thus we end up with a quite well-conditioned discrete problem in which we have the prior knowledge that most of the reflection coefficients are zero, and perhaps some more information about the distribution of the points at which the nonzero values occur (i.e., some information about the distribution of layer-width). This information should also be incorporated in the inversion process, in order to improve the identification.

An adaptive-windowing reconstruction method. We implemented an improved inversion procedure as follows. From the error propagation results we have a bound on the error in estimating k_p. If the value that the algorithm provided is close to zero within this bound then we set the reflection coefficient to zero, and proceed. Also if the reflection coefficient provided by the algorithm at some stage is outside the a priori range we set it to be equal to the nearest point within the allowed domain. Figure 7 provides a series of impedance reconstruction examples with various amounts of uniform noise, comparing the performance of the modified algorithm with the straightforward one. We can see from those results that the performance of the modified algorithm is very good, however at higher noise levels the thresholding process leaves some of the small impedance jumps undetected. In the algorithms with thresholding we have an obvious tradeoff between detection of small impedance jumps and the number of "false alarms" due to noise. In Fig. 8 we plotted the *impedance* error bound as predicted by the rather conservative reflection coefficient error range (3.12), comparatively for the two algorithms.

A further step in incorporating prior information could be taken if we knew something about the distribution of layer width. If, for example, we know that the layers have a minimal width, of say L discretized sections, then by a causality argument, we can deduce that prior to any impedance jump point, the response sequence $V = v_0, v_1, v_2, \cdots$ will have, in a noise-free case, $v_i = k_p u_i$ for $i = 0, 1, 2, \cdots, L$, where u_i are the initial lags of the U-sequence. This information can be used to both detect the point at which an impedance jump occurs and estimate the corresponding nonzero reflection coefficient.

For a practical implementation of such a procedure one can propagate the usual layer peeling process, in conjunction with, say, a maximum likelihood search. The search for the next impedance jump point and the corresponding reflection coefficient could be restricted to the neighborhood of a depth at which the usual layer-peeling procedure provided a significant reflection coefficient estimate.

We note that an inversion method based on maximum likelihood principles was proposed and analyzed by Mendel and Habibi-Ashrafi [18], [24] in the continuous case context and a slightly different propagation model. It is a very nice theoretical result of theirs that, by taking the maximum likelihood principle as a starting point one can argue that the procedure for inversion should in fact be the equivalent of a layer-peeling algorithm. The method developed in [24] is different from the one described above in the technique used for detecting the impedance jump points, in reflection coefficient identification and the use of estimates of error propagation. Our approach starts from the layer-peeling algorithm, which is the method that applies in the noise-free case, and combines it with error propagation results and some simple

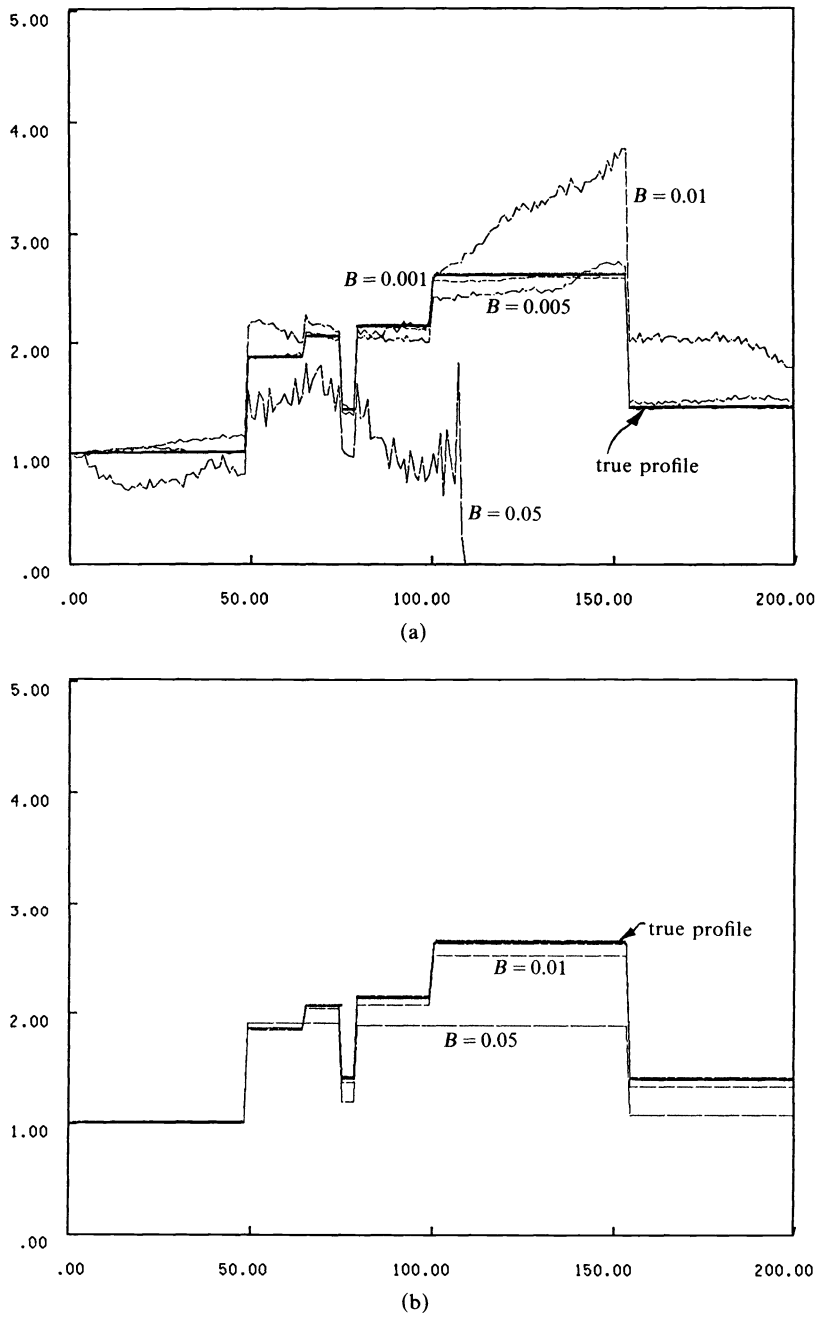


FIG. 7. *Impedance profile reconstruction (with the probing sequence of Fig. 3).* (a) *The standard algorithm.* (b) *The improved version with "adaptive windowing." Uniform noise levels were* $B = 0.001, 0.005, 0.01, 0.05$.

local processing aimed at incorporating and exploiting prior information. We do not treat the continuous case and the discretization process, conceptually done at the medium model level, induces an inherent limitation to the accuracy of the layer-width determination. In Fig. 9 we present several simulation results using our algorithm on a discretized profile corresponding to the inversion example treated in [24]. At low

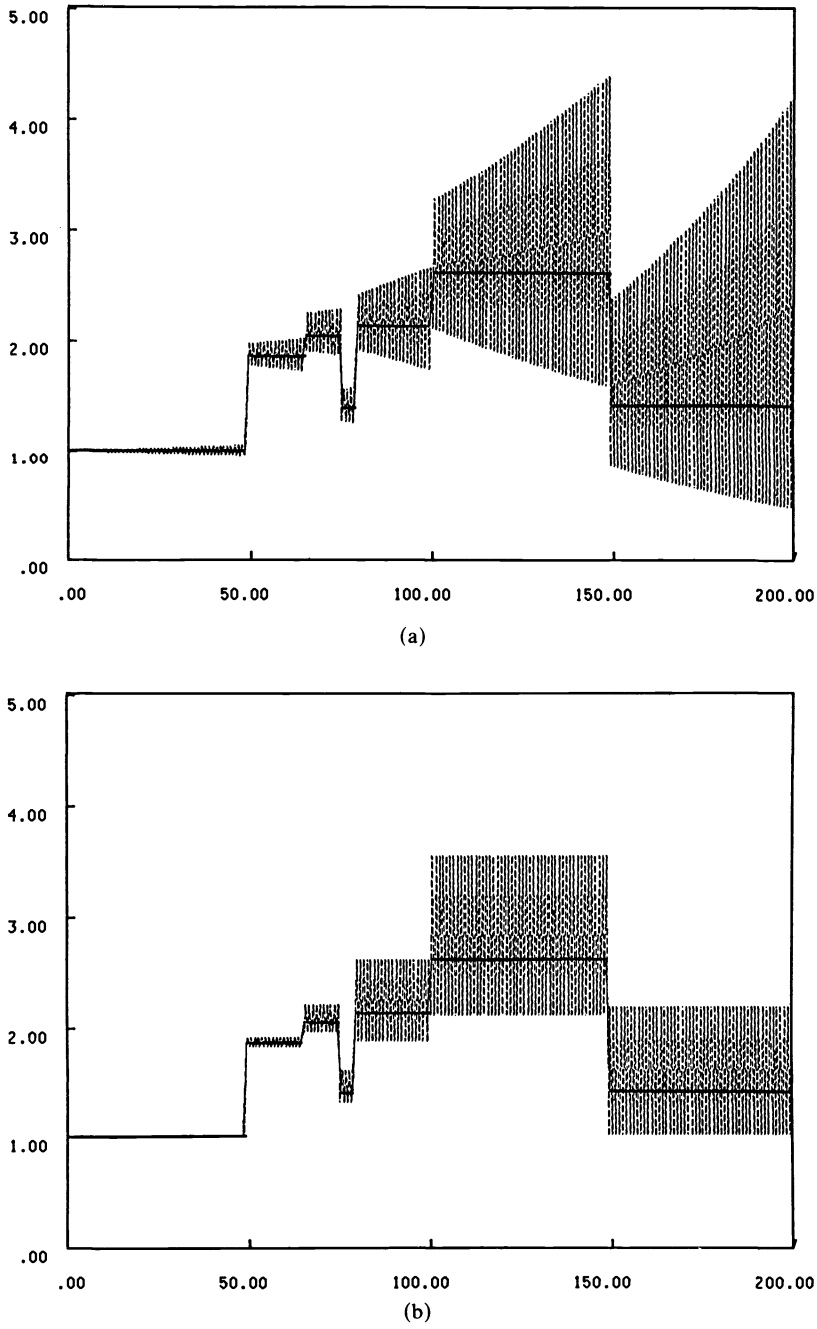


FIG. 8. *Impedance domains predicted by error bound* (3.12), *or the standard algorithm, with noise level* 0.005 (a) *and for the modified version with noise level* 0.01 (b).

and moderate levels of additive Gaussian noise the inversion results obtained by our simple algorithm are similar in both the precision of reflection coefficient estimates and layer-width determination to the algorithm described in [24]. For very low signal-to-noise ratios, none of the algorithms for inversion performs well, and this is due to the

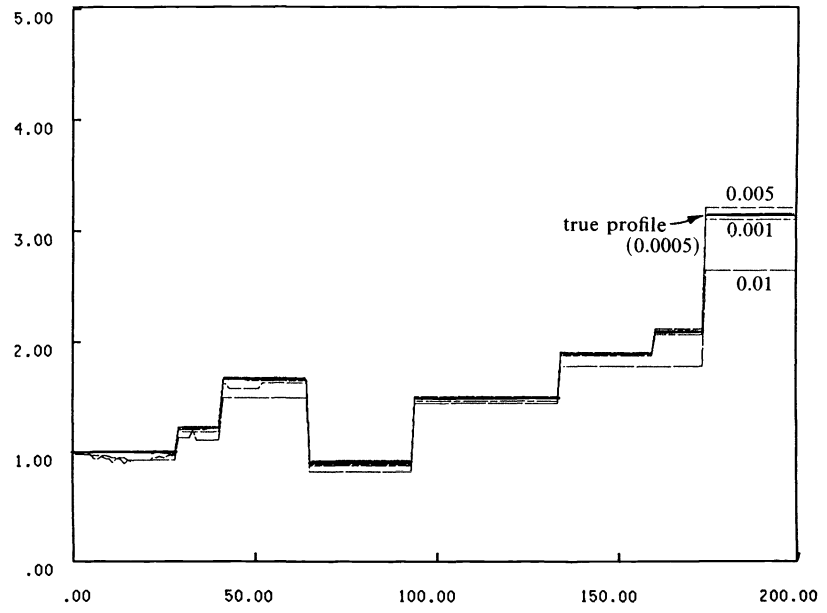


FIG. 9. *Impedance reconstruction example with Gaussian noise, and probing sequence of Fig.* 3. *The adaptive windowing algorithm was used and the noise standard deviations were* 0.0005, 0.001, 0.005, *and* 0.01.

fact that the error accumulation curve hits an unacceptable level at an early stage of the recursive medium identification.

We note that numerical experiments with layer-peeling type algorithms applied to noisy and bandlimited data, for various wave propagation models were also carried out by Symes and Corones and their coworkers, see e.g. [25]-[27]. In [25], an inversion procedure is described calling for the propagation of a Riccati recursion which is an equivalent description of the layer-peeling process (see [10]-[12]). At successive stages of the inversion algorithm this method provides the impulse response data for a portion of the medium starting at a deeper level. This process is mathematically equivalent to the Schur algorithm, however the implicit deconvolution involved at each step leads to an unnecessary complexity increase ($O(N^3)$ vs. $O(N^2)$), which is not expected to provide any improvement in recovery performance with noisy data.

5. Conclusions. Several issues concerning inverse scattering with noisy data are discussed in this paper. An error propagation analysis is provided which shows that we can obtain bounds on the performance of inverse scattering algorithms which are expressed in terms of the *recursively computed* reflection coefficients. The error propagation result, together with the established backwards numerical stability of this type of algorithm [13]-[15], provide a useful method for monitoring identification performance while propagating inversion algorithms.

We then discussed a variety of ways to incorporate prior information on the reflection coefficient profile into the layer-peeling algorithms. This led to significant improvements in the results obtained for an impedance profile reconstruction problem, on which these algorithms were tested.

Appendix. It is well known that one can compute the reflection coefficients corresponding to the "marine data": $1, r_1, r_2, \cdots, r_N$ in the following way. Construct a nested set of symmetric (positive definite) Toeplitz matrices R_p with first row

$[1 \quad r_1 \quad r_2 \quad \cdots \quad r_{p-1}]$. Solve the set of "normal equations"

$$R_p \begin{bmatrix} \phi_0^p \\ \phi_1^p \\ \vdots \\ \phi_{p-1}^p \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \tag{A1}$$

for $p = 1, 2, \cdots, N$. Then the reflection coefficient k_p is given by the inner product

$$k_p = -\sum_{j=0}^{p-1} r_{p-j} \phi_j^p . \tag{A2}$$

However, as we saw in § 1, the reflection coefficients can also be computed in a recursive manner, without inner products, by propagating the Schur algorithm. By a suitable extension of the matrices R_p we can avoid the inner product computations and identify the reflection coefficients recursively from solutions of the equations

$$\mathbf{R}_p \begin{bmatrix} \phi_{p-N-1}^p \\ \vdots \\ \phi_0^p \\ \vdots \\ \phi_N^p \end{bmatrix} = \begin{bmatrix} 0 \\ \vdots \\ 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \quad \text{or} \quad \mathbf{R}_p \phi^p = e_0. \tag{A3}$$

Here the extended matrix $\mathbf{R}_p$ is an $(2N+2-p) \times (2N+2-p)$ matrix defined by

$$\mathbf{R}_p = \begin{bmatrix} I & R_p' & 0 \\ 0 & R_p & 0 \\ 0 & R_p'' & I \end{bmatrix} \tag{A4}$$

where

$$R_p' = \begin{bmatrix} r_{N-p+1} & \cdots & r_N \\ \vdots & \cdots & \vdots \\ r_2 & \cdots & r_{p+1} \\ r_1 & \cdots & r_p \end{bmatrix} \quad \text{and} \quad R_p'' = \begin{bmatrix} r_p & \cdots & r_1 \\ r_{p+1} & \cdots & r_2 \\ \vdots & \cdots & \vdots \\ r_N & \cdots & r_{N-p+1} \end{bmatrix}$$

are nonsquare Toeplitz matrices. Note that the matrix $\mathbf{R}_p$ is invertible if and only if R_p is invertible.

It is easy to see that the reflection coefficients k_p equal to the ϕ_p^p entries of the vectors ϕ^p which can be computed recursively via

$$\phi^{p+1} = \frac{1}{1-k_p^2}(I + k_p J) T_p \phi^p \tag{A5}$$

where J is the "rotation" matrix and T_p is a matrix that sets the ϕ_p^p entry to zero and eliminates the first entry of ϕ_p, ϕ_{p-N-1}^p. The above formulation allows us to obtain bounds for the error propagation.

Suppose the scattering data, $r_1, r_2 \cdots$, is corrupted by noise uniformly bounded in absolute value by $\varepsilon > 0$. Assume that ε is small enough so that the perturbed Toeplitz matrices R_p remain positive definite. Denote by $\mathbf{R}_p + \mathbf{E}_p$ the extended matrix corresponding to the perturbed data $1, r_1 + \varepsilon_1, \cdots$. Applying the inversion process to the

perturbed scattering data is equivalent to recursively solving the sets of equations

$$(\mathbf{R}_p+\mathbf{E}_p)\psi^p = e_0 \tag{A6}$$

and the estimates of the reflection coefficients will be

$$\hat{k}_p = \psi_p^p. \tag{A7}$$

In order to compute error bounds we have to evaluate the magnitude of the difference between ϕ^p and ψ^p. Clearly

$$\begin{aligned}\phi^p-\psi^p &= [\mathbf{R}_p^{-1}-(\mathbf{R}_p+\mathbf{E}_p)^{-1}]e_0 \\ &= [\mathbf{R}_p^{-1}-(\mathbf{I}+\mathbf{R}_p^{-1}\mathbf{E}_p)^{-1}\mathbf{R}_p^{-1}]e_0\end{aligned} \tag{A8}$$

and hence

$$\phi^p-\psi^p = \mathbf{R}_p^{-1}\mathbf{E}_p\mathbf{R}_p^{-1}e_0+(\mathbf{R}_p^{-1}\mathbf{E}_p)^2\mathbf{R}_p^{-1}e_0+\cdots. \tag{A9}$$

Since for a fixed depth N the norms of $\mathbf{R}_p^{-1}$ are bounded, we get for small ε that

$$|\phi_p^p-\psi_p^p| = |\langle \mathbf{R}_p^{-1}\mathbf{E}_p\mathbf{R}_p^{-1}e_0, e_p\rangle| + O(\varepsilon^2). \tag{A10}$$

Now, it follows from (A3) that

$$\begin{bmatrix} & & r_p \\ & R_p & r_{p-1} \\ & & \vdots \\ 0\cdots 0 & & 1 \end{bmatrix}\begin{bmatrix}\phi_p^{p-1}\\ \phi_{p-1}^{p+1}\\ \vdots \\ \phi_0^{p+1}\end{bmatrix} = \begin{bmatrix}0\\0\\ \vdots \\ \phi_0^{p+1}\end{bmatrix} \tag{A11}$$

and using the definition (A4), we can expand (A11) to read

$$\mathbf{R}_p^{-T}e_p = \frac{1}{\phi_0^{p+1}}\begin{bmatrix}0\\ \vdots \\ 0\\ \phi_p^{p+1}\\ \phi_{p-1}^{p+1}\\ \vdots \\ \phi_0^{p+1}\\ 0\\ \vdots \\ 0\end{bmatrix}. \tag{A12}$$

Thus we have

$$\begin{aligned}|k_p-\hat{k}_p| &= |\langle \mathbf{E}_p\mathbf{R}_p^{-1}e_0, \mathbf{R}_p^{-T}e_p\rangle| \\ &\leqq \|\mathbf{E}_p\mathbf{R}_p^{-1}e_0\|_\infty \|(\phi_p^{p+1}\phi_{p-1}^{p+1}\cdots\phi_0^{p+1})\|_1 \frac{1}{\phi_0^{p+1}}.\end{aligned} \tag{A13}$$

From (A3) it follows that

$$\phi_0^{p+1} = \prod_{j=1}^{p}\frac{1}{1-k_j^2} \tag{A14}$$

and we also have

$$\|\mathbf{E}_p\mathbf{R}_p^{-1}e_0\|_\infty \leqq \varepsilon\|(\phi_0^p\phi_1^p\cdots\phi_{p-1}^p)\|_1. \tag{A15}$$

By induction, from (A3) we obtain

(A16)

$$\|(\phi_0^p \phi_1^p \cdots \phi_{p-1}^p)\|_1 \leqq \frac{1}{1-k_{p-1}^2}\|(\phi_0^{p-1}\phi_1^{p-1}\cdots\phi_{p-2}^{p-1})\|_1 + \frac{|k_{p-1}|}{1-k_{p-1}^2}\|(\phi_{p-2}^{p-1}\phi_{p-3}^{p-1}\cdots\phi_0^{p-1})\|_1$$

$$= \frac{1}{1-|k_{p-1}|}\|(\phi_0^{p-1}\phi_1^{p-1}\cdots\phi_{p-2}^{p-1})\|_1 \leqq \cdots \leqq \prod_{j=1}^{p-1}\frac{1}{1-|k_j|}.$$

Finally

$$\text{(A17)} \qquad |k_p - \hat{k}_p| \leqq \varepsilon \prod_{j=1}^{p-1}\frac{1}{1-|k_j|}\prod_{j=1}^{p}\frac{1}{1-|k_j|}\prod_{j=1}^{p}(1-k_j^2) = \varepsilon(1+|k_p|)\prod_{j=1}^{p-1}\frac{1+|k_j|}{1-|k_j|}$$

which yields the desired estimate (2.11).

In order to evaluate error bounds for the general case, i.e., for noise corrupting the scattering data sequence V, we have to realize that we can compute the equivalent "marine data" and in evaluating the error amplification introduced in this process. The equivalent marine data is related to a general input-response pair via a deconvolution, since clearly, both pairs of sequences $I+R=[1 r_1 r_2 \cdots]$, $R=[0 r_1 r_2 \cdots]$ and U, V characterize the same linear system. Writing out in matricial notation that $V/U = R/(I+R)$ yields that $\mathbf{R}=\mathbf{V}(\mathbf{U}-\mathbf{V})^{-1}$, where $\mathbf{R}$, $\mathbf{U}$ and $\mathbf{V}$ are lower-triangular Toeplitz matrices defined by R, U and V respectively. Now assume that the scattering data V are ε-perturbed. The perturbation in R that corresponds to this noise can be bounded, after some algebra, by the norm of $\varepsilon\mathbf{U}^{-1}(I-\mathbf{U}^{-1}\mathbf{V})^{-2}$ up to $O(\varepsilon^2)$ terms. Since $\mathbf{U}^{-1}\mathbf{V}$ is a nilpotent matrix (recall that $\mathbf{V}$ is strictly lower triangular) we obtain that the amplification factor is bounded by

$$\text{(A18)} \qquad M(U, V) = \frac{\|\mathbf{U}^{-1}\|}{(1-\|\mathbf{U}^{-1}\mathbf{V}\|)^2}.$$

The losslessness of the scattering medium ensures that $\|\mathbf{U}^{-1}\mathbf{V}\|$ is always smaller than unity.

Acknowledgment. We thank our referees for their constructive comments.

REFERENCES

[1] R. G. NEWTON, *Inversion of reflection data for layered media: a review of exact methods*, Geophys. J. Royal Astr. Soc., 65/1 (1981), pp. 191-215.

[2] J. A. WARE AND K. AKI, *Continuous and discrete inverse scattering problems in a stratified elastic medium* I: *plane waves at normal incidence*, J. Acoust. Soc. of America, 45/4 (1969), pp. 911-921.

[3] J. G. BERRYMAN AND R. R. GREENE, *Discrete inverse methods for elastic waves in layered media*, Geophysics, 45/2 (1980), pp. 213-233.

[4] J. M. MENDEL AND F. HABIBI-ASHRAFI, *A survey of approaches to solving inverse problems for lossless layered media systems*, IEEE Trans. Geosci. Remote Sensing, GE-18/4 (1980), pp. 320-330.

[5] E. A. ROBINSON, *Spectral approach to geophysical inversion by Lorentz, Fourier and Radon transforms*, Proc. IEEE, 70/9 (1982), pp. 1039-1054.

[6] W. SYMES, *Stable solution of the inverse reflection problem for a smoothly stratified medium*, SIAM J. Math. Anal., 12 (1981), pp. 421-453.

[7] F. SANTOSA AND H. SCHWETLICK, *The inversion of acoustical impedance profile by methods of characteristics*, Wave Motion, 4 (1982), pp. 99-110.

[8] K. P. BUBE AND R. BURRIDGE, *The one-dimensional inverse problem of reflection seismology*, SIAM Rev., 25 (1983), pp. 497-559.

[9] G. ARSENE, Z. CEAUSESCU, F. POTRA AND D. TIMOTIN, *A new algorithm for detecting reflection coefficients in layered media*, Annales Geophysicae, 1, 4-5 (1983), pp. 285-290.

[10] A. M. BRUCKSTEIN, B. C. LEVY AND T. KAILATH, *Differential methods in inverse scattering*, I.S.L. Report, EE Dept., Stanford University, Stanford, CA, 1983; SIAM J. Appl. Math, 45 (1985), pp. 312-335.

[11] A. YAGLE AND B. C. LEVY, *The Schur algorithm and its applications*, L.I.D.S. Report, MIT, 1983, also Acta Appl. Math., 3/3 (1985).

[12] A. M. BRUCKSTEIN AND T. KAILATH, *Inverse scattering for discrete transmission-line models*, SIAM Rev., to appear.

[13] A. BULTHEEL, *Error analysis of incoming and outgoing schemes for the trigonometric moment problem*, Proc. Conference on Rational Approximation, Amsterdam, Springer-Verlag, Berlin, 1981.

[14] I. GOHBERG AND I. KOLTRACHT, *Numerical solutions of integral equations, fast algorithms and the Krein-Sobolev equation*, Numer. Math., 47 (1985), pp. 237-288.

[15] G. CYBENKO, *The numerical stability of the Levinson-Durbin algorithm for Toeplitz systems of equations*, this Journal, 1 (1980), pp. 303-319.

[16] E. A. ROBINSON AND S. TREITEL, *Geophysical Signal Processing*, Prentice-Hall, Englewood Cliffs, NJ, 1980, pp. 492-505.

[17] J. F. CLAERBOUT, *Fundamentals of Geophysical Data Processing*, McGraw-Hill, New York, 1976.

[18] J. M. MENDEL, *Optimal Seismic Deconvolution*, Academic Press, New York, 1983.

[19] B. GOPINATH AND M. M. SONDHI, *Inversion of the telegraph equation and the synthesis of nonuniform lines*, Proc. IEEE, 59 (1971), pp. 383-392.

[20] B. GOPINATH AND M. M. SONDHI, *Determination of the shape of the human vocal tract from acoustical measurements*, Bell Syst. Tech. J., 49 (1970), pp. 1195-1214.

[21] H. J. LANDAU, *The inverse problem for the vocal tract and the moment problem*, SIAM J. Math. Anal., 14 (1983), pp. 1019-1035.

[22] P. DEWILDE, A. C. VIEIRA AND T. KAILATH, *On a generalized Szëgo-Levinson realization algorithm for optimal linear predictors based on a network synthesis approach*, IEEE Trans. Circuits and Systems, CAS-25 (1978), pp. 663-675.

[23] G. H. GOLUB AND C. VAN LOAN, *Matrix Computations*, Johns Hopkins Press, Baltimore, 1983.

[24] F. HABIBI-ASHRAFI AND J. M. MENDEL, *Estimation of parameters in lossless layered media systems*, IEEE Trans. Automat. Control, AC-27 (1982), pp. 31-49.

[25] J. P. CORONES, M. E. DAVISON AND R. J. KRUEGER, *Direct and inverse scattering in the time domain via invariant imbedding equations*, J. Acoust. Soc. Amer., 74 (1983), pp. 1535-1541.

[26] W. W. SYMES AND G. ZIMMERMAN, *Experiments in impedance profile inversion using noisy and bandlimited data*, AMOCO Research Report, 1982.

[27] K. R. DRIESSEL AND W. W. SYMES, *Coefficient identification problems for hyperbolic PDE's: some fast and accurate algorithms for seismic inverse problems in one space dimension*, unpublished research report, 1981.

[28] I. KOLTRACHT AND P. LANCASTER, *Condition numbers for Toeplitz and Block Toeplitz matrices*, in Advances in Operator Theory dedicated to I. Schur, I. Gohberg, ed., Birkhäuser Verlag, Basel, 1986, pp. 271-301.

SIAM J. SCI. STAT. COMPUT.
Vol. 7, No. 4, October 1986

A SIMPLE APPROXIMATE RANDOM CHOICE METHOD FOR SCALAR CONSERVATION LAWS*

JOEL S. COHEN† AND JAMES A. LAVITA†

Abstract. The Random Choice Method for single scalar conservation laws is examined, and an approximate Riemann solver is constructed. The construction is accomplished using a result of Dafermos regarding the solution of a single conservation law with a general convection term. The algorithm is described and numerical results for the Random Choice Method with the approximate Riemann solver are compared to those using an exact solution to the Riemann problem. The approximate Riemann solver gives results similar to those obtained with the exact Riemann solver. It is easier to implement with a general convection term but does require more cpu time.

Key words. gas dynamics, numerical analysis, Random Choice Method

AMS(MOS) subject classifications. Primary, 65M99; secondary 35L65, 76S05

1. Introduction. Chorin (1976) has introduced an effective computational method, called the Random Choice Method (RCM), for nonlinear hyperbolic partial differential equations. He first considered the one-space-dimensional equations arising in the flow of an ideal inert gas, and then extended the method to the situation of a reacting gas. The technique produces sharp fronts (in one dimension), correct reaction zone speeds and approaches first-order accuracy.

The RCM, as a numerical scheme for computing the solutions of nonlinear systems of hyperbolic partial differential equations, is based on a fundamental result of Glimm (1965) concerning convergence in the large of approximations to nonlinear hyperbolic conservation laws. Chorin (1976), (1977) appears to have been the first to construct a significant computational scheme based on the work of Glimm. (Earlier, Moler and Smoller (1970) had implemented Glimm's method, but had concluded, at that time, that it did not possess significant computational value.) The problems treated with the RCM (or Glimm's) method involve Burgers' equation (Colella (1979)), porous flow (Albright et al. (1980), Albright and Concus (1980), Concus and Proskurowski (1979)), and the previously mentioned gas dynamical problems dealt with by Chorin.

The significant feature of the RCM from our point of view is that it is not a difference method, i.e. does not employ difference approximations to derivatives. Instead, the RCM employs a probabilistically sampled value chosen from the exact solution of an elementary wave pattern constructed as a local Riemann problem. The main problem in studying the RCM is the difficulty associated with constructing the Riemann solver which is needed to advance in time. This appears to be the major stumbling block to easy and clear implementation of the RCM in practical situations. Thus, an analysis of approximate Riemann solvers and their utility is essential. However, there is a substantial gap between the theoretical analysis suggested by several researchers (see Harten, Lax and van Leer (1983) for a discussion of these points) and workable algorithms.

The RCM has also been employed in situations which, while still complex and physically significant, lend themselves to less cumbersome numerical experimentation. Initiated by Concus and Proskurowski (1979) and continued by Concus and his co-workers, a series of papers on the application of the RCM to two-phase reservoir

* Received by the editors October 17, 1983, and in revised form April 20, 1985.
† Department of Mathematics and Computer Science, University of Denver, Denver, Colorado 80208.

flow has appeared. Here, the RCM is used to compute the solutions of the Buckley-Leverett (B-L) equation, the principal wavelike component in an otherwise elliptic setting. In the simplest case, the B-L equation has a convection term with an S-shaped function $f(u)$. With the addition of gravity, however, a nonconvex function of greater complexity appears. In other words, the strict nonlinearity of the equation is violated. The B-L equation is one of a general class of nonconvex equations which is of physical interest. In particular, there is application to two-phase immiscible flow of petroleum in underground reservoirs.

In order to apply the RCM to the B-L equation, or to any equation for that matter, the solution of some particular Riemann problem has to be found. In the previously mentioned examples, the Riemann problem was solved exactly, and this solution was employed where needed. This Riemann solution was, in fact, a major part of the analytical work, and also appears to be a significant portion of the algorithmic development and the computational cost. Thus, recent attempts have been made to reduce the computational complexity of the method by introducing approximate solutions to the Riemann problem. Harten and Lax (1981) have discussed this issue and presented some results for scalar equations. The application and use of approximate methods remains, however, inconclusive.

In this paper the authors examine another approach suited to the limited, but important, special case of the scalar conservation law. The approach which seems to have several advantages is based on a method developed by Dafermos (1972). As described below in more detail, Dafermos constructed the solution to the scalar conservation law

$$\frac{\partial u}{\partial t}+\frac{\partial f(u)}{\partial x}=0$$

under mild conditions on $f(u)$ which do not involve convexity requirements. His method involves approximation of $f(u)$ by piecewise linear functions, thereby simplifying the complicated wave interactions.

By combining the solution structure of Dafermos with the elements of the RCM, and by testing it against some known results, we hope to uncover strengths and weaknesses of a more easily implemented Riemann solver. Clearly, a method which employs specific properties of a particular form of the advection term is expected to run better, or at least faster, in that situation. However, the simplicity and clarity of a general method could present advantages in situations where an analytic representation of the Riemann solver is very cumbersome to program and debug.

2. The Random Choice Technique with discrete solver. Consider the single hyperbolic equation

$$\frac{\partial u}{\partial t}+\frac{\partial f(u)}{\partial x}=0. \tag{2.1}$$

The RCM is a numerical technique for finding approximate solutions to systems of hyperbolic conservation laws. The description given here is a two-step version of the scheme for a single equation (2.1). The technique approximates the solution at time t_n $(n=1, 2, \cdots)$ for space points $x=i\,\Delta x(i=0, 1, 2, \cdots, L)$. The value t_{n+1} is chosen so that $\Delta t=t_{n+1}-t_n$ satisfies the Courant-Friedrichs-Lewy (CFL) condition

$$\left(\frac{\Delta t}{\Delta x}\right)\max_u |a(u)|<1 \tag{2.2}$$

where $a(u)=f'(u)$. Let u_i^n be an approximation to $u(i\,\Delta x, t_n)$, the exact solution of (2.1). The value $u_{i+1/2}^{n+1/2}$ at $t=t_n+\Delta t/2$ and $x=(i+\frac{1}{2})\,\Delta x$ is obtained by solving the Riemann problem for (2.1) with initial data

$$u(x,0)=\begin{cases} u_i^n, & x\leqq(i+\frac{1}{2})\,\Delta x,\\ u_{i+1}^n, & x>(i+\frac{1}{2})\,\Delta x. \end{cases} \tag{2.3}$$

Let $v(x,t)$ be the solution to this Riemann problem and let $\alpha_j\in[0,1]$ be an equidistributed sequence of real numbers. An approximation to $u((i+\frac{1}{2})\,\Delta x,\ (n+\frac{1}{2})\,\Delta t)$ is given by

$$u_{i+1/2}^{n+1/2}=v\left((i+\alpha_j)\,\Delta x,\frac{\Delta t}{2}\right).$$

The values u_i^{n+1} are obtained in a similar fashion by solving the Riemann problem with initial data

$$u(x,0)=\begin{cases} u_{i-1/2}^{n+1/2}, & x\leqq i\Delta x,\\ u_{i+1/2}^{n+1/2}, & x>i\Delta x. \end{cases} \tag{2.4}$$

The α_j values are chosen using the van der Corput sequence (see Colella (1982)). A new value α_j is chosen for each sequence of Riemann problems at each half-time step.

The RCM requires the solution to a sequence of Riemann problems at each half-time step. A detailed solution for the Riemann problem

$$\begin{aligned} &\frac{\partial u}{\partial t}+\frac{\partial f(u)}{\partial x}=0,\\ &u(x,0)=\begin{cases} u_L, & x\leqq 0,\\ u_R, & x>0 \end{cases} \end{aligned} \tag{2.5}$$

given by Proskurowski (1981) is summarized below. Using the Rankine-Hugoniot jump condition and the Oleinik E-condition, one obtains the following solution

Case I. $(u_L\leqq u_R)$. Consider the set

$$M=\{(u,v)\colon u_L\leqq u\leqq u_R,\ v\geqq f(u)\} \tag{2.6}$$

and let H be the convex hull of M. The set H is bounded below by $h(u)$, the largest concave up function less than or equal to $f(u)$. In order to describe $h(u)$, one must find the set of intermediate values $u_L=u_1<\cdots<u_H=u_R$ which are endpoints of subintervals over which $f(u)$ coincides with $h(u)$. On each subinterval (u_k, u_{k+1}), $h(u)$ either coincides with $f(u)$ or it is given by the straight line between the points $(u_k, f(u_k))$ and $(u_{k+1}, f(u_{k+1}))$. If over the interval (u_k, u_{k+1}), $f(u)$ equals $h(u)$, u_k is connected to u_{k+1} by an expansion wave centered at the origin which satisfies the equation $a(u)=x/t$. Conversely, if over the given interval $f(u)\neq h(u)$, u_k is connected to u_{k+1} by a shock wave centered at the origin with speed

$$S=\frac{f(u_{k+1})-f(u_k)}{u_{k+1}-u_k}.$$

Case II. $(u_R<u_L)$. Consider the set

$$M=\{(u,v)\colon u_R\leqq u\leqq u_L,\ v\leqq f(u)\} \tag{2.7}$$

and let H be the convex hull of M. The set H is bounded above by the smallest concave (down) function $h(u)$ greater than or equal to $f(u)$. The description of $h(u)$ and the corresponding solution to the Riemann problem proceeds as in Case I.

For functions $f(u)$ with at most two inflection points, the computer algorithms for the above solution to the Riemann problem are straightforward but tedious to implement (see Concus and Proskurowski (1979), LaVita (1981), Proskurowski (1981)). Finding the interior points u_k often means finding line segments which are tangent to $f(u)$ at one or two endpoints. To develop an algorithm, one must consider numerous different cases which depend on the relationships between u_L and u_R and the inflection points of $f(u)$. An algorithm along these lines for general $f(u)$ would be quite complex.

An alternative approach is to use a polygonal approximation to $f(u)$ and compute the convex hulls using an algorithm for discrete point sets. A solution of this type to (2.5) for piecewise linear functions $f(u)$ has been given by Dafermos (1972). In this case an admissible weak solution to (2.5) consists of a finite number of constant states separated by shocks centered at the origin. Suppose $f(u)$ is piecewise linear and is defined on an interval $[a, b]$. The graph of $f(u)$ is a collection of straight lines with vertices at the points $a = u_1 < u_2 < \cdots < u_m = b$. Assume for now that u_L and u_R are vertices of $f(u)$. The solution to the Riemann problem is divided into two cases:

Case I. $(a \leqq u_L \leqq u_R \leqq b)$. Let M be defined by (2.6) and let H be the convex hull of M. The lower boundary of H is a polygonal line $h(u)$ whose vertices $u_L = v_1 < v_2 < \cdots < v_h = u_R$ are a subset of the vertices which define $f(u)$. Since H is convex, the slopes satisfy

$$\frac{f(v_k)-f(v_{k-1})}{v_k - v_{k-1}} \leqq \frac{f(v_{k+1})-f(v_k)}{v_{k+1}-v_k}. \tag{2.8}$$

The Riemann problem solution is given by

$$u(x,t) = \begin{cases} u_L & \text{for } -\infty < \dfrac{x}{t} \leqq \dfrac{f(v_2)-f(v_1)}{v_2-v_1}, \\ v_2 & \text{for } \dfrac{f(v_2)-f(v_1)}{v_2-v_1} < \dfrac{x}{t} \leqq \dfrac{f(v_3)-f(v_2)}{v_3-v_2}, \\ \vdots & \vdots \\ v_{h-1} & \text{for } \dfrac{f(v_{h-2})-f(v_{h-1})}{v_{h-2}-v_{h-1}} < \dfrac{x}{t} \leqq \dfrac{f(v_h)-f(v_{h-1})}{v_h - v_{h-1}}, \\ u_R & \text{for } \dfrac{f(v_{h-1})-f(v_h)}{v_{h-1}-v_h} < \dfrac{x}{t} < \infty. \end{cases} \tag{2.9}$$

Case II. $(a \leqq u_R \leqq u_L \leqq b)$. Let M be defined by (2.7) and let H be the convex hull of M. The upper boundary of H is a polygonal line $h(u)$ whose vertices $u_R = v_h < v_{h-1} < \cdots < v_1 = u_L$ are a subset of the vertices which define $f(u)$. Since H is convex the slopes satisfy (2.8) and the Riemann problem solution is again given by (2.9).

3. Implementing the discrete Riemann solver. The advantages of the discrete Riemann solver (DRS) are its generality and simplicity. In fact, the actual algorithm does not utilize any specific features (such as inflection points) of a given $f(u)$. For this reason the approach has the potential of being more time-consuming than other more involved approaches which utilize specific features of a given function $f(u)$. There are a number of different ways to implement the DRS. One must select an algorithm which is sufficiently accurate and time competitive with other approaches to solving the Riemann problem. Listed below are the prominent features of the DRS algorithm:

1. A discretization $a = u_1 < u_2 < \cdots < u_m = b$ with $\Delta u = u_k - u_{k-1}$ for all k is fixed at the beginning of a run. All function evaluations $f(u_k)$ and derivative evaluations $a(u_k)$ (for the Courant-Friedrichs-Lewy condition) are computed and stored at this time. A solution to (2.5) (and (2.1)) is now restricted to values from the vertex set $\{u_k\}$.

2. An approximation to the initial guess $u(x, 0) = f(x)$ is obtained by rounding $f(x)$ at each grid point to the closest value in the vertex set $\{u_k\}$.

3. The RCM is implemented in the usual fashion. The Riemann problem initial data in (2.5) now consists of initial data from the vertex set $\{u_k\}$.

4. Consider the Riemann problem with initial data

$$u(x, 0) = \begin{cases} u_{\min}, & x \leqq 0, \\ u_{\max}, & x > 0 \end{cases} \tag{3.1}$$

where $u_{\min} < u_{\max}$ belong to the vertex set $\{u_k\}$. The convex hull required by the Dafermos solution is obtained using an algorithm similar to the one given by Akl and Toussaint (1978). Since the original vertex set is already sorted, the algorithm computes the convex hull in $O(\max\text{-}\min)$ operations.

5. A binary search algorithm is used to locate the appropriate vertex value in (2.9).

4. Examples. Certainly, for very small values of Δu, the RCM using the DRS will give nearly the same result as the RCM with an exact Riemann solver (ERS). However, the DRS solution requires approximately $O(1/\Delta u)$ operations and is therefore extremely slow for very small values of Δu. In practice, an exact Riemann solver also involves some degree of numerical approximation. For complicated functions $f(u)$, the exact solution to the Riemann problem often involves finding line segments which are tangent to $f(u)$ at one or two endpoints. All such approximations are done to a given level ε. However, for the examples below, the ERS algorithms involve relatively little ε dependent iteration. For this reason, the cpu times are not very sensitive to changes in ε and one can choose extremely small values of epsilon and obtain accurate solutions to the Riemann problem.

Since it is not practical to use small values of Δu when using the DRS solution, it is important to observe the relationship between error norms and the size of Δu. In the experiments discussed below, accurate solutions are obtained for relatively coarse values of Δu.

Example I. Consider Burgers' equation

$$\frac{\partial u}{\partial t} + \frac{\partial (u^2/2)}{\partial x} = 0, \qquad u(x, 0) = 2 + \sin(x), \qquad x \in [-\pi, \pi]$$

with periodic boundary conditions. This example has been used by Harten and Lax (1981) for testing a Random Choice finite difference scheme. The solution is smooth for $t \leqq 1$. At $t = 1$ a shock forms which first grows in strength and then decays. Experiments were performed with $\Delta x = \pi/10$ for a number of different values for Δu. To facilitate the comparsion of different solutions, a fixed time step $\Delta t = 0.09$ (which satisfies the CFL condition) is used. For $\Delta u = 0.01$, the DRS solution is virtually indistinguishable from ERS solution. For 50 time steps the L^∞ norm between the DRS solution and the ERS solution is always less than 0.0048. The DRS solution tracked the shock as accurately as the ERS solution. For both solutions, the position of the shock is occasionally off by one grid position. Figure 1 is a plot of the DRS solution ($\Delta u = 0.01$) and the exact solution at $t = 2.25$ (25 time steps). For larger values of

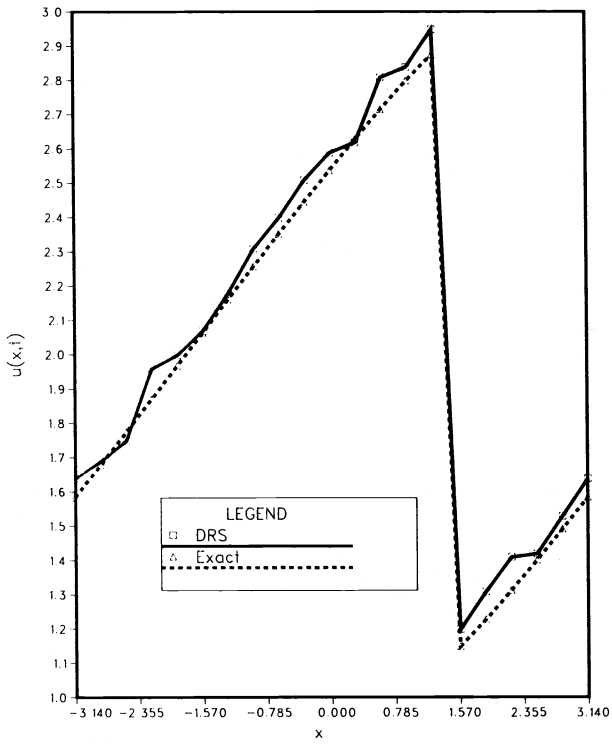


FIG. 1. *DRS solution* ($DELTA = 0.01$) *and exact solution for Burgers' equation at* $t = 2.25$.

$\Delta u \leqq 0.08$, the DRS solution is nearly as accurate as the ERS solution; however, there is a greater tendency for the shock position to be off by one grid position. Since the solution values are restricted to values in the vertex set $\{u_k\}$, there is also a tendency for the solution to have more of a steplike structure. Figure 2 is a plot of the DRS solution ($\Delta u = 0.08$) and the exact solution at $t = 2.25$. Although the shock position is off by one grid position, the solution away from the shock is still as accurate as the ERS solution.

Example II. Consider the Buckley-Leverett equation with gravity forces

$$\frac{\partial u}{\partial t} + \frac{\partial f(u)}{\partial x} = 0,$$

$$f(u) = \frac{u^2}{1+\alpha(1-u^2)}(1-\lambda u^2),$$

$$u(x, 0) = \frac{0.1}{(0.1+x)}, \qquad x \in [0, 1],$$

$$u(0, t) = 1.0, \qquad u(1, t) = u(1, 0)$$

where $\alpha = 0.5$ and $\lambda = 10$. This equation was considered by Proskurowski (1981) as a test case for the RCM technique. Since $f(u)$ has two inflection points, the ERS solution is more involved in this case. The experiments were run with $\Delta x = .05$ and $\Delta t = 0.005$ for various values of Δu. At each time step comparisons were made between the DRS solution, the ERS solution and a simulated exact solution which consists of an ERS

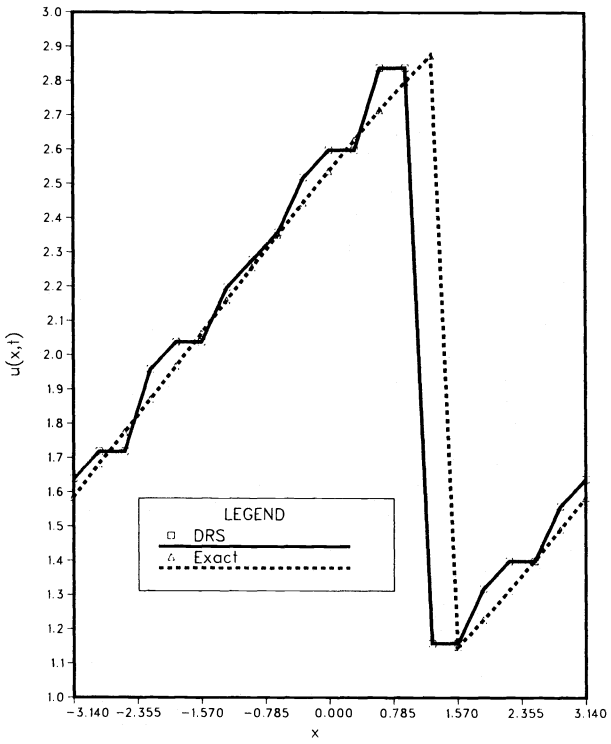


FIG. 2. *DRS solution (DELTA U = 0.08) and exact solution for Burgers' equation at* $t = 2.25$.

solution with $\Delta x = .00625$ and $\Delta t = .000625$. Fixed time steps which satisfy the CFL condition (2.2) are used to facilitate the comparison of solutions. Since the exact solution to the Riemann problem for this equation is rather involved, the example provides a reasonable benchmark for time comparisons. Again, good results were obtained for relatively coarse values of Δu. The solution in this case includes a shock which grows as time progresses. For $\Delta u \leqq .02$, the DRS solution tracked the simulated exact solution as well as the ERS solution. For both the DRS solution and ERS solution, the observed shocks were off by one grid point for the most of the run. At grid points away from the shock, the DRS solution exhibited the same order of accuracy as the ERS solution. Figure 3 is a plot of the DRS solution $\Delta u = 0.02$ and the simulated exact solution at $t = 0.1$ (20 time steps). Figure 4 is the corresponding plot for $\Delta u = 0.01$ and Figure 5 is a plot of the ERS solution and exact solution. For $\Delta u = 0.01$, the ERS solution and the DRS solution are virtually identical. Since the DRS solution does not use specific properties of $f(u)$, execution times are slower than the corresponding ERS solution. For example, with $\Delta u = 0.01$, the DRS solution took about twice as much execution time as the corresponding ERS solution. For $\Delta u = 0.005$, the DRS solution took about three times as much execution time.

These experiments suggest that it is possible to obtain accurate Random Choice solutions to (2.1) using the DRS. The DRS is easy to implement and is applicable to general $f(u)$ with appropriate Δu. The method is potentially useful in situations where time considerations are not of utmost importance and the exact Riemann solution is tedious to implement.

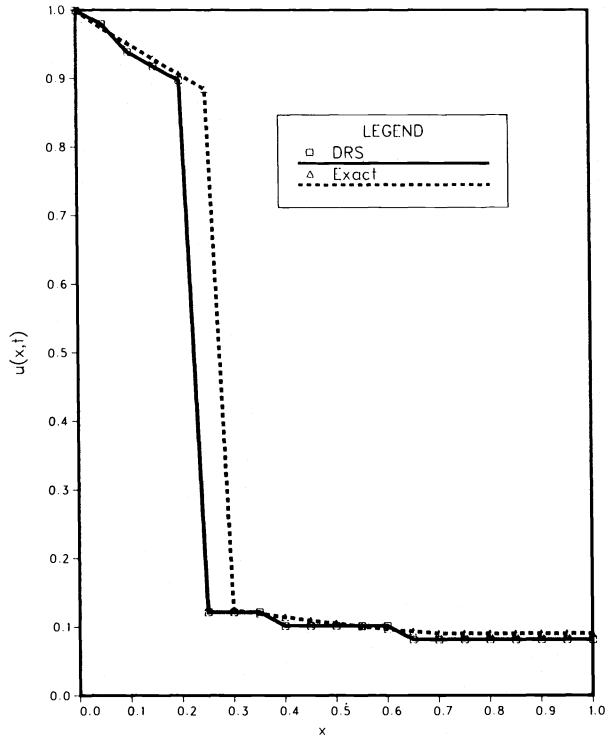


FIG. 3. *DRS solution (DELTA U = 0.02) and exact solution for Buckley-Leverett with gravity equation at t = 0.1.*

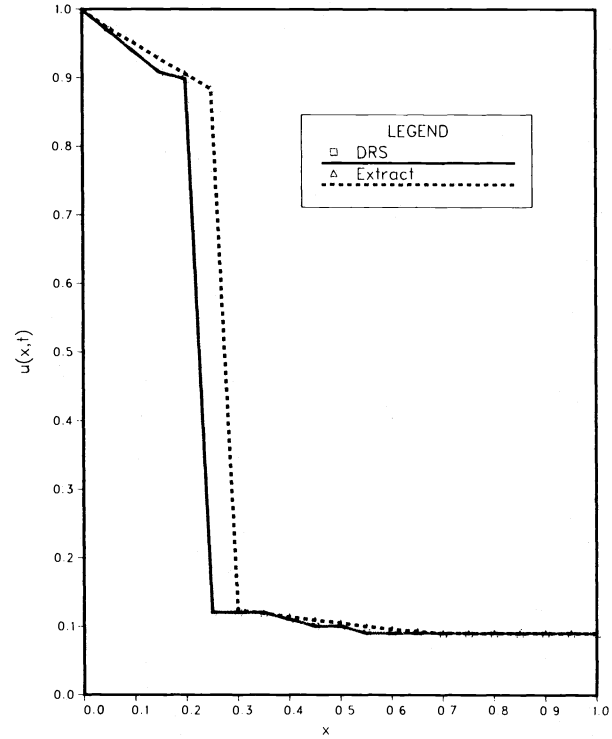


FIG. 4. *DRS solution (DELTA U = 0.01) and exact solution for Buckely-Leverett with gravity equation at t = 0.1.*

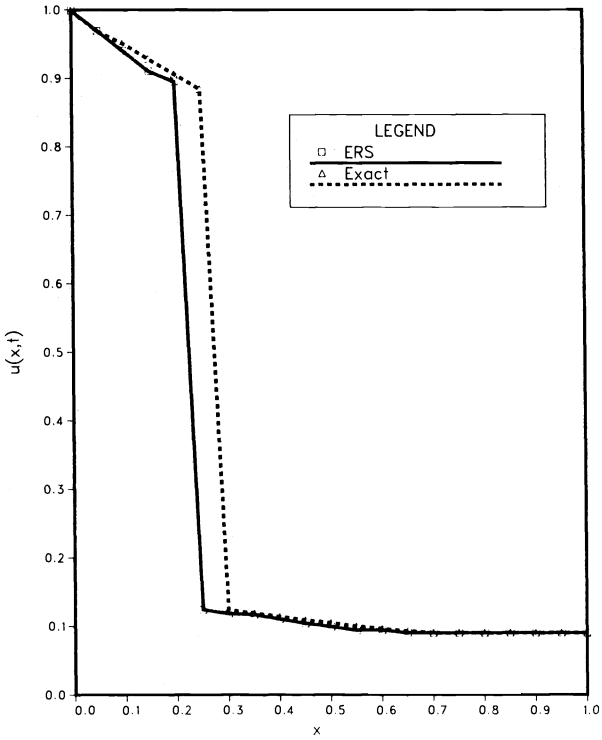


FIG. 5. *ERS solution and exact solution for Buckley–Leverett equation at* $t = 0.1$.

Acknowledgment. The authors would like to thank Howard Pomerantz for producing the graphs in Figs. 1–5.

REFERENCES

S. AKL AND G. TOUSSAINT, *A fast convex hull algorithm*, Inform. Process. Lett., 7 (1978), pp. 219–222.

N. ALBRIGHT, C. ANDERSON AND P. CONCUS, *The random choice method for calculating fluid displacement in a porous medium*, in Boundary and Interior Layers, Computational and Asymptotic Methods, J. J. H. Miller, ed., Boole Press, Dublin, 1980, pp. 3–13.

N. ALBRIGHT AND P. CONCUS, *On calculating flows with sharp fronts in a porous medium*, in Fluid Mechanics and Energy Conservation, J. D. Buckmaster, ed., Society for Industrial and Applied Mathematics, Philadelphia, PA, 1980, pp. 172–184.

A. J. CHORIN, *Random choice solution of hyperbolic systems*, J. Comput. Phys., 22 (1976), pp. 517–533.

———, *Random choice methods with applications to reacting gas flow*, J. Comput. Phys., 25 (1977), pp. 253–272.

P. COLELLA, *An analysis of the effect of operator splitting and of the sampling procedure on the accuracy of Glimm's method*, Ph.D. dissertation, Mathematics Department, University of California, Berkeley, CA, 1979.

———, *Glimm's method for gas dynamics*, this Journal, 3 (1982), pp. 76–110.

P. CONCUS AND W. PROSKUROWSKI, *Numerical solution of a nonlinear hyperbolic equation by the random choice method*, J. Comput. Phys., 30 (1979), pp. 153–166.

C. M. DAFERMOS, *Polygonal approximations of solutions of the initial value problems for a conservation law*, J. Math. Anal. Appl., 38 (1972), pp. 33–41.

J. GLIMM, *Solutions in the large for nonlinear hyperbolic systems of equations*, Comm. Pure Appl. Math., 18 (1965), pp. 697–715.

A. HARTEN AND P. D. LAX, *A random choice finite difference scheme for hyperbolic conservation laws*, SIAM J. Numer. Anal., 18 (1981), pp. 289–315.

A. HARTEN, P. LAX AND B. VAN LEER, *On upstream differencing and Godunov-type schemes for hyperbolic conservation laws*, SIAM Rev., 25 (1983), pp. 35–61.

J. LAVITA, *Some remarks on the comparison of methods for computing discontinuous solutions of conservation laws*, Fourth International Association for Mathematics and Computers in Simulation, Proceedings of International Symposium on Computer Methods in Partial Differential Equations, Lehigh University, Bethlehem, PA, June 30-July 2, 1981.

C. MOLER AND J. SMOLLER, *Elementary interactions in quasi-linear hyperbolic systems*, Arch. Rational Mech. Anal., 37 (1970), pp. 309-322.

W. PROSKUROWSKI, *A note on solving the Buckley-Leverett equation in the presence of gravity*, J. Comput. Phys., 41 (1981), pp. 136-141.

SIAM J. SCI. STAT. COMPUT.
Vol. 7, No. 4, October 1986

018

A MONTE CARLO METHOD FOR SCALAR REACTION DIFFUSION EQUATIONS*

ARTHUR S. SHERMAN† AND CHARLES S. PESKIN†

Abstract. A probabilistic method is presented to solve reaction diffusion equations. A random walk is combined with creation and destruction of elements. The method is applied to Nagumo's equation. Numerical results are given demonstrating convergence of the method. The stochastic process also gives a direct probability interpretation of the equation which may be useful for analysis.

Key words. Monte Carlo, random walk, reaction diffusion, stochastic processes, Nagumo's equation

1. Introduction. In this paper we present an original stochastic method for solving nonlinear initial value problems of the type

$$u_t = u_{xx} + f(u), \qquad -\infty < x < \infty, \tag{1.1a}$$

$$u(x, 0) = g(x). \tag{1.1b}$$

It is well known that the heat equation

$$\begin{aligned} &u_t = u_{xx}, \qquad -\infty < x < \infty, \\ &u(x, 0) = g(x) \end{aligned} \tag{1.2}$$

can be solved up to time T by sampling a Gaussian distribution of mean 0 and variance $2T$ [3, pp. 113–117].

Equation (1.1) has been solved by fractional step methods in which the random walk principle is used to model the diffusion term and an ordinary differential equation is solved, analytically or numerically, to account for the nonlinearity [5]. The advantages of stochastic methods are that they are grid free, stable independent of time step size, and give high resolution near sharp wave fronts without excessive effort.

We also use a fractional step method, but both steps are stochastic. Our method thus gives a probabilistic interpretation directly for (1.1).

We work with $v = u_x$ which satisfies the equation

$$\begin{aligned} &v_t = v_{xx} + f'(u)v, \qquad -\infty < x < \infty, \\ &v(x, 0) = g'(x). \end{aligned} \tag{1.3}$$

We model this by letting discrete computational elements undergo a Gaussian random walk in $T/\Delta t$ time steps of size Δt in which, at each time step, an element at position x has a probability per unit time $|f'(u(x, t))|$ of

(a) being destroyed if $f'(u(x, t)) < 0$,

(b) splitting into two elements if $f'(u(x, t)) > 0$.

The values of u are then obtained by integrating v.

2. The method.

2.1. Discrete representation of functions. We consider the case where u is a monotonically increasing function of x, with $\lim_{x\to-\infty} u(x, t) = 0$ and $\lim_{x\to+\infty} u(x, t) = 1$. Then $v(x, t) > 0$ for $-\infty < x < +\infty$, $t \geqq 0$ and $\lim_{x\to\pm\infty} v(x, t) = 0$. $u(x, t)$ is approximated

* Received by the editors October 29, 1984, and in revised form July 15, 1985. This work was supported by the Department of Energy under contract DE-AC02-76ER03077 at New York University.

† Courant Institute of Mathematical Sciences, New York University, New York, New York 10012.

by a step function

$$u_N(x, t^l), \qquad t^l = l\Delta t, \qquad l = 1, 2, \cdots, T/\Delta t \tag{2.1}$$

which has $N^l = N(t^l)$ equal jumps of size $1/N^0$ located at

$$X_i^l = X_i(t^l), \qquad i = 1, \cdots, N^l. \tag{2.2}$$

N^0 is a convenient number of initial computational elements, and N^l is a random variable which fluctuates around N^0. For $i = 1, 2, \cdots, N^l$

$$X_i(t^l) = X_i(t^{l^*}) + \sum_{m=l^*}^{l} \eta_i^m \tag{2.3}$$

where η_i^m is a Gaussian random variable and t^{l^*} is the time at which the ith element was created. The stochastic process by which these random variables are chosen is described in § 2.2 below.

$v(x, t)$ is approximated by $v_N(x, t^l) = (\partial/\partial x)u_N(x, t^l)$. Thus, v_N is a sum of δ-functions

$$v_N(x, t^l) = \sum_{i=1}^{N^l} \frac{1}{N^0}\, \delta(x - X_i(t^l)). \tag{2.4}$$

u_N can be recovered from v_N by integrating. This is accomplished by counting the number of elements to the left of a given point x:

$$u_N(x, t^l) = \int_{-\infty}^{x} v_N(\xi, t)\, d\xi = \frac{1}{N^0} \sum_{i=1}^{N^l} H(x - X_i(t)) = \frac{1}{N^0} \sum_{i:\, X_i^l < x} 1. \tag{2.5}$$

At the jumps, $x = X_i^l$, u_N is defined to be the average of the left- and right-hand limits:

$$u_N(X_i^l, t^l) = \tfrac{1}{2} \lim_{\varepsilon \downarrow 0} \{u_N(X_i^l + \varepsilon, t^l) + u_N(X_i^l - \varepsilon, t^l)\}. \tag{2.6}$$

We sort the list of elements in order of position whenever we need to compute u, so that the actual evaluation step can be done by stepping through the list once. We used quicksort [1, pp. 92–97]. See § 4.5 for some remarks on sorting technique.

One could use variable sized jumps. However, then one would have to keep track of the sizes of the jumps as well as their positions. Furthermore, using equal jumps automatically concentrates more elements where the function u is increasing more rapidly. In all further discussion we assume that the jumps are of equal size, $1/N^0$.

2.2. The stochastic process. To solve (1.1) up to a specified time T, we proceed as follows.

(a) Initialization

Pick N^0 arbitrarily

Divide the *range* of g (in our case the interval $[0, 1]$) into N^0 equal subdivisions:

For $i = 1, \cdots, N^0$ set

$$X_i^0 = X_i(0) = g^{-1}((i - \tfrac{1}{2})/N^0) \tag{2.7}$$

Divide the time interval $[0, T]$ into L equal subdivisions of size $\Delta t = T/L$.

(For accuracy Δt should be chosen so that $\max_{0\leq u\leq 1}|f'(u)\Delta t|$ is small compared to 1. Further discussion of the choice of Δt is given in § 4.5)

For $l=1, 2, \cdots, L$ we proceed from time $(l-1)\Delta t$ to $l\Delta t$ as follows:

(b) Random walk step.

For $i=1, 2, \cdots, N^{l-1}$

$$X_i^l = X_i^{l-1} + \eta_i^l \Delta t \tag{2.8}$$

where η_i^l are Gaussian random variables, independent in i and l with mean 0 and variance $2\Delta t$.

(c) Sort the N^{l-1} elements by position.

(d) Create and destroy elements.

For $i=1, 2, \cdots, N^{l-1}$

(i) Compute $u_N(X_i^l, t^l)$ by "integrating" v_N as described in § 2.1

(ii) Compute $p_i^l = f'(u_N(X_i^l))$. (We do this by formula since f is a cubic polynomial in our model problem.)

1. If $p_i^l > 0$, create a new element at X_i^l with probability $P_i^l = p_i^l \Delta t$.
2. If $p_i^l < 0$, delete ith element with probability $P_i^l = -p_i^l \Delta t$. (See § 4.5 for details of indexing of elements.)

(e) Set $N^l = N^{l-1} + (\#\text{ created}) - (\#\text{ deleted})$.

(f) Return to (b).

3. The model problem. With an eye toward future applications we considered the Fitzhugh-Nagumo equations [2, p. 370]:

$$u_t = u_{xx} + u(u-a)(1-u) + z, \qquad z_t = \varepsilon u. \tag{3.1}$$

These equations are a simplification of the Hodgkin-Huxley equations for conduction of nerve impulses. We further simplified to the case $z=0$ for which there is a steady traveling wavefront solution [2, p. 373, pp. 392-395]:

$$u(x, t) = \tilde{u}(\xi) = \frac{1}{1+\exp(-\xi/\sqrt{2})} \tag{3.2}$$

where

$$\xi = x + \theta t, \tag{3.3}$$

$$\theta = \sqrt{2}(\tfrac{1}{2} - a). \tag{3.4}$$

We shall refer to this as the "steady solution."

If the initial data, $g(x)$, is monotonic, the solution will remain monotonic in x. Since this holds, the numerical method described in § 2 can be applied. (By construction, the method generates monotonic solutions.)

There are several features of (3.1) in the case $z=0$ which we used to test the numerical method.

(a) General initial data rising monotonically from 0 to 1 as x goes from $-\infty$ to $+\infty$ evolve in time to the steady solution (3.2) [6].

In our tests we found that if we took the steady profile as initial data

$$g_1(x) = \tilde{u}(x) \tag{3.5}$$

then the profile remained approximately unchanged except for statistical fluctuations.

We also tested

$$g_2(x)=\begin{cases}0, & x\leqq -b,\quad b>0,\\ \dfrac{x+b}{2b}, & -b<x\leqq b,\\ 0, & x>b\end{cases} \tag{3.6}$$

and

$$g_3(x)=H(x)=\begin{cases}0, & x<0,\\ 1, & x\geqq 0,\end{cases} \tag{3.7}$$

and found that the profiles evolved into the steady profiles. See § 4.1.

(b) If we view u as a probability distribution with density v, then the total "mass" $m(t)$ is given by

$$m(t)=\int_{-\infty}^{\infty} v(x,t)\,dx=u(\infty,t)-u(-\infty,t). \tag{3.8}$$

Therefore

$$\frac{dm}{dt}=\int_{-\infty}^{\infty}\frac{\partial v}{\partial t}(x,t)\,dx \tag{3.9}$$

$$=\int_{-\infty}^{\infty}\frac{\partial^2 v}{\partial x^2}(x,t)\,dx+\int_{-\infty}^{\infty}f'(u(x,t))\frac{\partial u}{\partial x}(x,t)\,dx \tag{3.10}$$

$$=f(u(\infty,t))-f(u(-\infty,t)). \tag{3.11}$$

Now $f(0)=0$ and, by our construction $u(-\infty,t)=0$ so

$$\frac{dm}{dt}=f(u(\infty,t))=f(m). \tag{3.12}$$

This O.D.E. has a stable equilibrium at $m=1$ since $f(1)=0$ and $f'(1)<0$. In the continuous equation $m(0)=1$ and therefore $m(t)=1$ for all t. In the discrete computational model m fluctuates because of the random creation and destruction of elements. The stability of the solution $m=1$ of (3.11) suggests, however, that such fluctuations will be self-correcting and that m will be continually driven towards the correct value $m=1$.

We tested this numerically by adjusting the jump size, or mass per element, to be $1/N^l$ at each step. When this is done,

$$u_N(\infty,t^l)-u_N(-\infty,t^l)\equiv 1 \tag{3.13}$$

and the method does not "know" whether there is a surplus or deficit of elements. We observed that in this case the fluctuation in N^l/N^0 increases substantially. This demonstrates that the stability property regulates the number of elements above and beyond the effect of the law of large numbers. See § 4.2 for numerical results.

(c) The final criterion for testing the numerical method is the wavespeed of the computed traveling wave solution. For the steady solution, the wavespeed can be interpreted as the rate of translation, μ, of the center of mass, $\bar{x}(t)$ of the probability density represented by v at time t:

$$\bar{x}(t)=\int_{-\infty}^{\infty} xv(x,t)\,dx, \tag{3.14}$$

$$\mu = \frac{d\bar{x}}{dt}. \tag{3.15}$$

The center of mass of the discrete distribution of computational elements is

$$\bar{x}_N(t^l) = \bar{x}_N^l = \frac{1}{N^l} \sum_{i=1}^{N^l} X_i^l \tag{3.16}$$

and the average wavespeed over a time interval $k\Delta t$ is

$$\frac{\bar{x}_N^{l+k} - \bar{x}_N^l}{k\Delta t}. \tag{3.17}$$

We observed that in the steady case ($u(x, 0) = g_1(x)$), the average wavespeed over long periods of time agreed well with (3.4), with large fluctuations over short time intervals.

In the nonsteady cases ($u(x, 0) = g_2(x)$ and $u(x, 0) = g_3(x)$), the wavespeed is not defined since the profiles are deforming as they move. However, the rate of translation of the center of mass is a well-defined quantity, and we found that the average rates of translation converged with increasing time to the theoretical steady wavespeed as the profiles converged to the steady profile. See § 4.3 for numerical results.

The above is an instance of the fact that the speed of the center of mass is determined by the profile:

$$\frac{d\bar{x}}{dt} = \int_{-\infty}^{\infty} x \frac{\partial v}{\partial t}(x, t)\, dx \tag{3.18}$$

$$= \int_{-\infty}^{\infty} x \frac{\partial^2 v}{\partial x^2}(x, t)\, dx + \int_{-\infty}^{\infty} x f'(u(x, t)) \frac{\partial u}{\partial x}(x, t)\, dx \tag{3.19}$$

$$= x \frac{\partial v}{\partial x}(x, t)\Big|_{-\infty}^{\infty} - \int_{-\infty}^{\infty} \frac{\partial v}{\partial x}(x, t)\, dx + x f(u(x, t))\Big|_{-\infty}^{\infty} - \int_{-\infty}^{\infty} f(u(x, t))\, dx. \tag{3.20}$$

If we assume that $u(x, t) \to 0(1)$ as $x \to -\infty(+\infty)$ sufficiently rapidly this reduces to

$$\frac{d\bar{x}}{dt} = -\int_{-\infty}^{\infty} f(u(x, t))\, dx.$$

For the three criteria, profile, mass conservation, and wavespeed, our goal was to demonstrate the improvement in the approximations as the initial number of elements, N^0, is increased and Δt is decreased. The numerical results in § 4 confirm this, and suggest the convergence of the computed solution to the actual solution as $N^0 \to \infty$ and $\Delta t \to 0$.

4. Numerical results.

4.1. Overview. The numerical tests were carried out in double precision (epsmch = 2.78E−17) on a VAX 11/780. Three sets of initial data were used: g_1, g_2, and g_3 as described in § 3.

Figure 1 shows $g(x)$ (see (3.5)) and the computed solutions at times $t = 5.0$ and $t = 10.0$ for $N^0 = 250$ and 1000. $a = 0.1$ giving $\theta = .566$ and $\Delta t = 0.1$. Evidently there is variation in the shape, height (mass conservation), and location (wavespeed) of the profiles. We measure the errors in these three quantities separately below. It is clear, however, that the $N^0 = 1000$ curves are smoother and are better approximations to the steady profile than the $N^0 = 250$ curves.

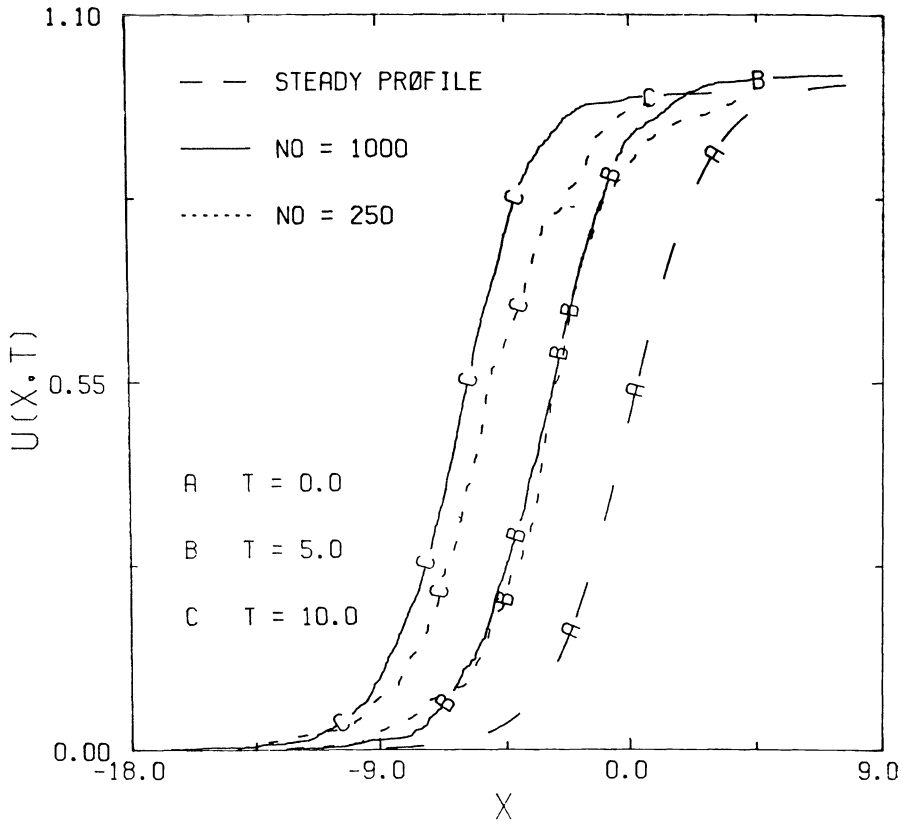


FIG. 1. *Steady solution.*

Figure 2 shows $g_3(x)$ (see (3.6)) and the computed solutions at successive times, demonstrating the evolution to the steady state. $N^0 = 1000$, $\Delta t = 0.1$, and $a = 0.1$. Similar results were obtained for $g_2(x)$.

4.2. Mass conservation. To check whether the numerical method conserves mass the program was run for 10.0 time units with $\Delta t = 0.1$ and with various values of N^0. N^l/N^0 was computed every 1.0 time units. Table 1 displays the average of the 10 values of N^l/N^0, the standard deviations, and for ease of comparison, the error (difference from 1.0).

The left-hand columns give the results for the standard case in which the size of the jumps is set at $1/N^0$ and left unchanged. The right-hand columns show the effect of setting the size of the jumps at $1/N^l$ at the lth time step. This disables the stability mechanism as discussed in § 3. The errors are evidently much larger, but they tend to decrease as N^0 increases, suggesting that the law of large numbers is driving the total mass towards 1 as $N^0 \to \infty$. Note also that the variation as measured by the standard deviation tends to decrease as N^0 increases in this case.

Figure 3 shows the variation over time of the mass for $N^0 = 800$ in the stable and nonstable cases. Note that in the nonstable case the total mass drifts and that the error increases with time.

It is interesting to observe that the data are independent of the choice of $g(x)$. Since we use equal sized jumps, the only property that distinguishes different profiles,

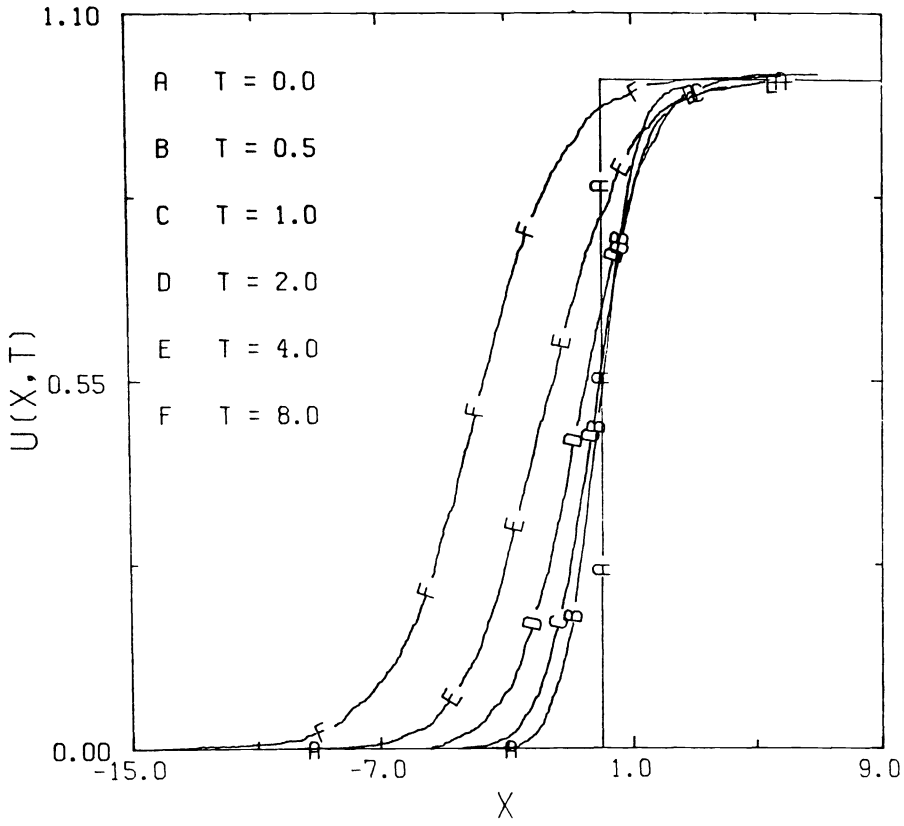


FIG. 2. *Nonsteady case.*

TABLE 1
Mass conservation $\Delta t = 0.1$.

	Stable case		Stability disabled	
N_0	Mean $N^1/N^0 \pm$ S.D.[1]	Mean error	Mean $N^1/N^0 \pm$ S.D.	Mean error
200	1.0060 ± .0258	.0060	1.0910 ± .0344	.0910
400	1.0940 ± .0099	−.0152	1.0155 ± .0164	.0155
600	0.9987 ± .0155	−.0013	0.9877 ± .0149	−.0123
800	0.9987 ± .0098	−.0013	1.0325 ± .0186	.0325
1000	1.0026 ± .0148	.0026	0.9896 ± .0129	−.0104

[1] S.D. = standard deviation.

given N^0, is the locations on the x-axis corresponding to the values of u. But the probabilities of element creation and destruction depend only on u, so if two profiles begin with the same N^0, they will have the same N^1, hence the same N^2, etc.

4.3. Wavespeed. For the steady case we checked the wavespeed by plotting the location of the center of mass every 1.0 time units against time and comparing that with a straight line of slope $-\theta$. For the runs in Fig. 4, $a = 0.25$ and $\theta = .354$. The average velocity over 10.0 time units for $N^0 = 250$ was −.350, and for $N^0 = 1000$, −.345.

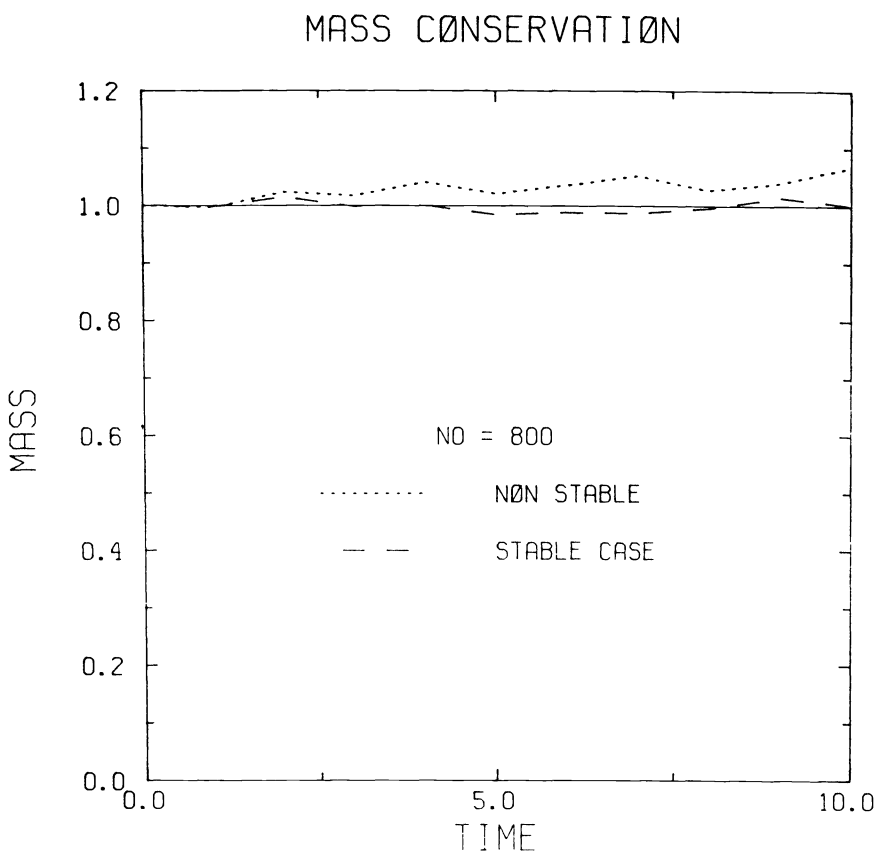


FIG. 3

Over short time intervals (say 0.1 time units) the velocity fluctuated widely and was even observed to have the wrong sign.

Again, increasing N^0 reduces the fluctuation and reduces the root mean square error in center of mass location. Reducing Δt also improves the approximation, but the effect is much smaller. We analyze the relative contributions quantitatively for the profile error in § 4.4.

Figure 5 is the center of mass vs. time plot using $u(x, t) = g_3(x)$. After $t = 5.0$ the profile appeared to have stabilized at approximately the steady profile. We superimpose a straight line of slope $-\theta = -.354$ on the center of mass trajectory from that time on. Note that the center of mass speed approaches the steady wavespeed from below. Similar results were obtained for the case $u(x, 0) = g_2(x)$, with $b = 5.0$ (see (3.6)) except that the center of mass speed decreases to the steady speed.

4.4. Profile. In this section we discuss the error in the shape of the profile, and give a tentative formula for the dependence of the error on N and Δt, where N is the average number of computational elements.

We computed the solution up to $t = 2.0$ using $u(x, 0) = g_1(x)$. To separate the profile error from the mass error we normalized the profile for output purposes so that $u(\infty, t) = 1$ by setting the jump size to $1/N^l$. To eliminate the effect of wavespeed error, we calculated the center of mass and translated the computed profile so that the center

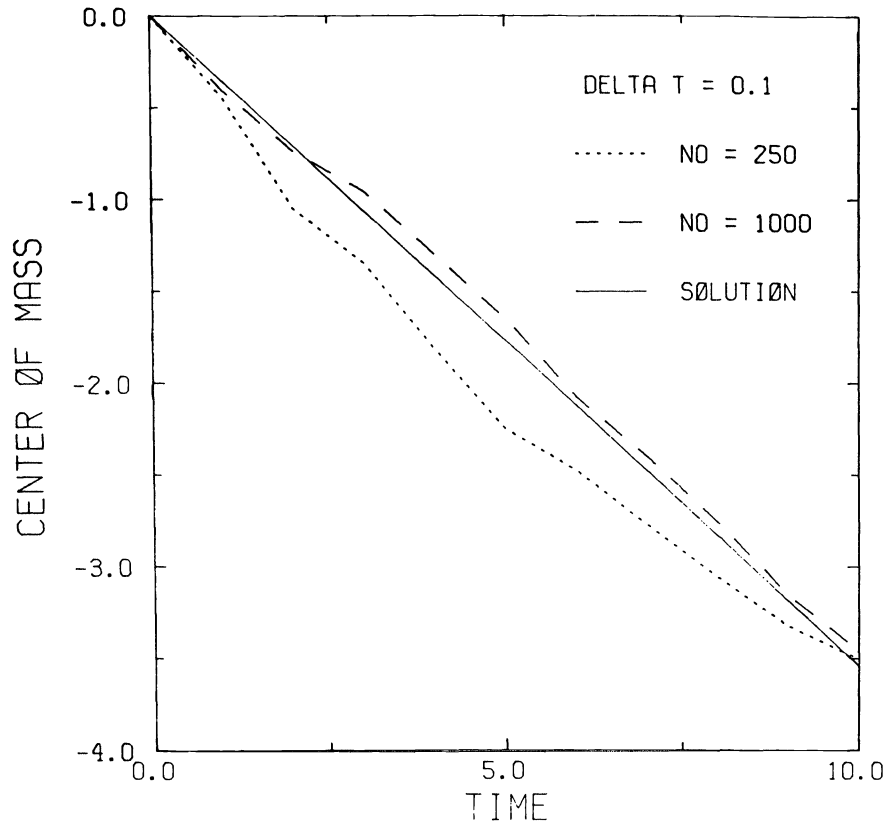


FIG. 4. *Wavespeed—steady case.*

of mass was at $x = 0$. We then calculated by explicit formula the norm of the difference between the solution and the steady profile:

$$\|u_N - \tilde{u}\|_2 = \left(\int_{-\infty}^{\infty} (u_N(x, T) - \tilde{u}(x))^2 \, dx \right)^{1/2}. \tag{4.1}$$

We hypothesized an error of the form

$$\varepsilon = AN^{\alpha} + B(\Delta t)^{\beta}. \tag{4.2}$$

To compute A and α, we fixed Δt and varied N. We then computed the best straight line fit to

$$\log \varepsilon = \log \tilde{A} + \alpha \log N, \qquad \tilde{A} = A\left(1 + \frac{B\Delta t^{\beta}}{AN^{\alpha}}\right) \tag{4.3}$$

for a sequence of Δt values $\to 0$. Table 2 gives the results, and suggests that $\alpha \approx -.5$.

To compute B and β, we fixed N^0 and varied Δt. Table 3 shows the best straight line fits to

$$\log \varepsilon = \log \tilde{B} + \beta \Delta t, \qquad \tilde{B} = B\left(1 + \frac{AN^{\alpha}}{B\Delta t^{\beta}}\right), \tag{4.4}$$

for various values of N^0. These results are less clear-cut because N^0 evidently must

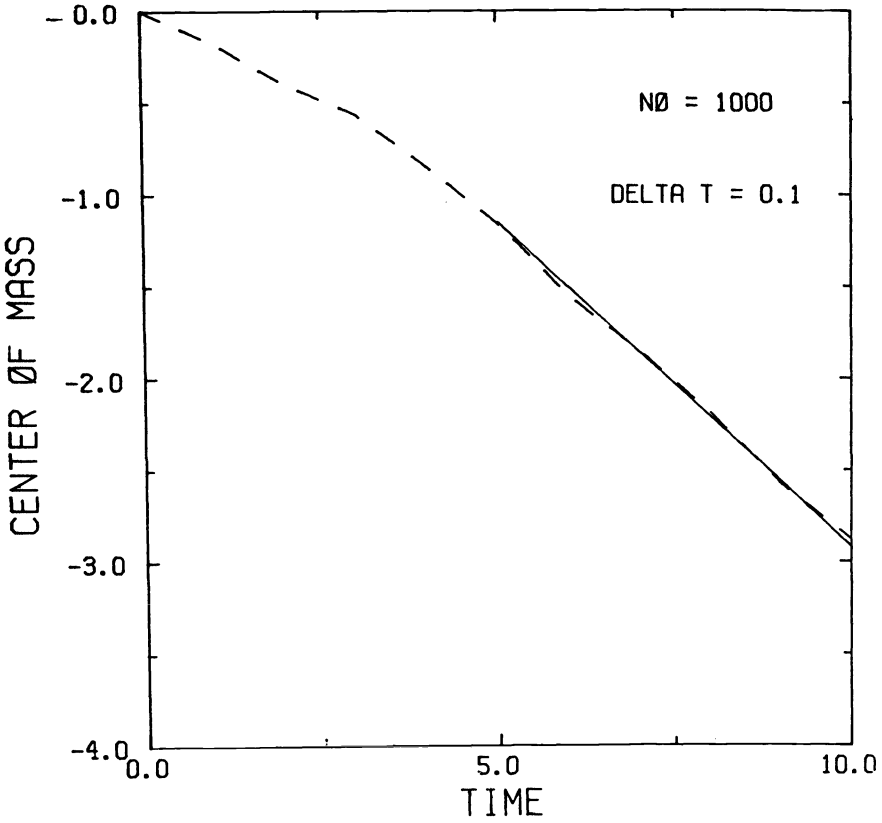


FIG. 5. *Wavespeed—nonsteady case.*

TABLE 2
Best least squares fit.

Δt	$\tilde{A}$	α
.1	.87	−.53
.05	.79	−.52
0.25	.88	−.53

N^0 ranges from 2 to 250, $T = 2.0$.

TABLE 3
Best least squares fit.

N^0	$\tilde{B}$	β
2000	.035	.37
4000	.037	.47
8000	.039	.72
16000	.036	.65
32000	.034	.87
64000	.040	.89

At ranges from 1.0 to 0.1, $T = 2.0$.

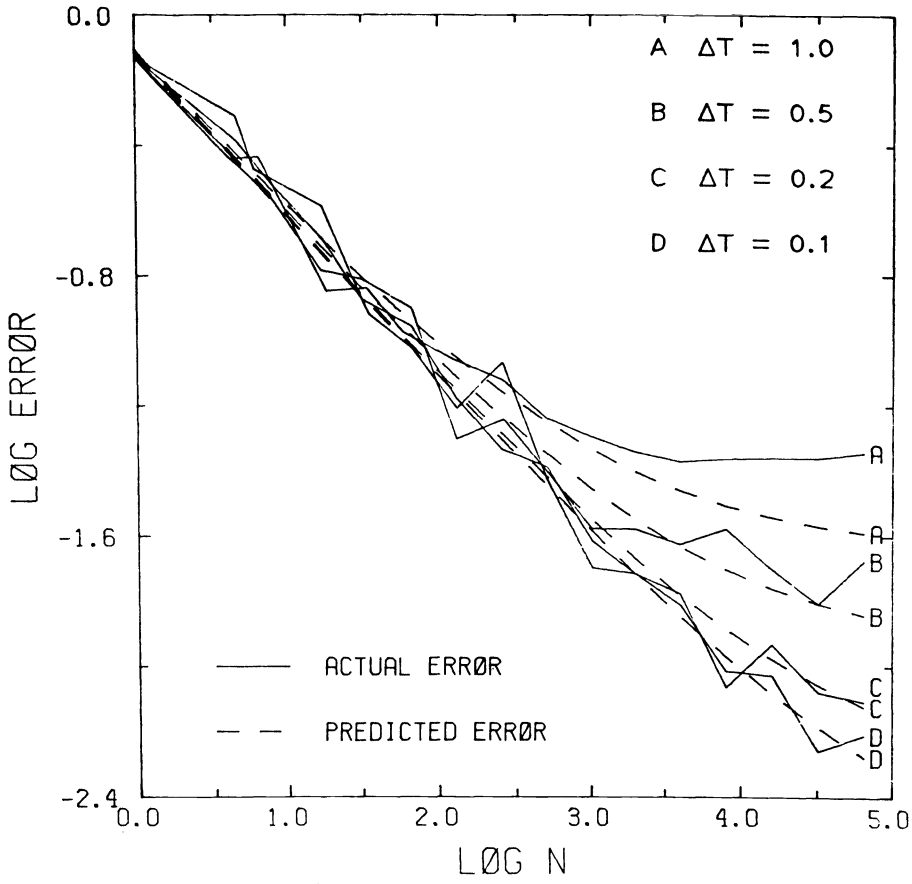


FIG. 6

be taken extremely large for the Δt error to dominate. We concluded that the computed values for β were tending towards 1.0 as $N^0 \to \infty$.

Finally we recalculated A and B by assuming $\alpha = -.5$ and $\beta = 1.0$, and finding the best multiple linear fit to

$$\varepsilon = AN^{-.5} + B\Delta t. \tag{4.5}$$

Various data sets yielded values of A ranging from 0.49 to 0.83 and values of B ranging from .012 to .036, in reasonably good agreement with Tables 1 and 2.

Figure 6 shows the log of the actual error and the predicted error plotted against log N for several values of Δt. $A = 0.76$ and $B = 0.023$ for the set of data chosen.

4.5. Sorting and other algorithmic considerations. In this section we discuss the data structures and space and time complexity of the algorithm.

The bulk of the storage is a one-dimensional array to record the jump locations. Initially the list consists of N^0 jumps and a free list of unused spaces at the end. When an element is created, its location is recorded in the first available free space. It is sorted into its proper position prior to evaluating $u(x, t)$. When an element is destroyed, its location is set to a large flag value. It is stepped over in the random walk and is automatically sorted into the free list before evaluating u. An array of length $(1.2)N^0$

was sufficient in practice to allow for these fluctuations. A small stack of length $O(\log N^0)$ is needed for sorting. Thus the storage requirements are approximately $O(N^0)$.

All the steps of the algorithm as described in § 2 require one pass through the jump array per time step, except for the sorting step. Experiment confirms that the CPU time exclusive of sorting is $O(LN)$, where L is the number of time steps and N is the average number of elements.

For the sorting, we used a nonrecursive version of quicksort since the program is in Fortran. This method has an average time complexity of $O(k \log k)$ over random lists of length k, with the smallest constant in practice of $O(k \log k)$ methods [1]. However, the performance can be as bad as $O(k^2)$ for special lists, so we checked the time for sorting empirically. It was $O(LN \log N)$. There are improved versions of quicksort which guard against the worst case performance, but there is still the problem that the percentage of time spent sorting grows with N. Some results are given in Table 4. A faster method could perhaps be developed to take advantage of the fact that the list is approximately sorted in the large, since, on average, the elements do not move far in one time step.

We can use the above results to derive a guideline for choosing N^0 and Δt. Let us assume for simplicity that the time complexity is $O(LN)$. If we want to solve up to time T with an error ε most efficiently, then we must minimize

$$CN^0/\Delta t, \qquad C = \text{constant}$$

subject to the constraint

$$\varepsilon = A(N^0)^{-1/2} + B(\Delta t).$$

This yields

$$N^0 = \frac{A^2}{4B^2(\Delta t)^2}.$$

Unfortunately, A and B are not known sufficiently precisely to give a meaningful choice for $N^0(\Delta t)^2$, but casual experimentation suggests a value of about 100 to 300.

5. Conclusions and future directions. We believe that the numerical results given demonstrate that the algorithm does in fact solve Nagumo's equation. The performance could perhaps be improved by applying variance reduction techniques.

The algorithm is presently suited for solving scalar, one-dimensional reaction diffusion equations. One desirable extension would be to systems of equations, such as the Hodgkin–Huxley equations. A further possible extension is to problems in more than one space dimension, such as the several variants of Hodgkin–Huxley for heart

TABLE 4
CPU *time* (*sec.*).

N^0	L	Sorting	Other	Percentage sorting
250	10	.54	.96	36
500	10	1.19	1.99	37
1000	2	.52	.81	39
1000	10	2.55	3.81	40
4000	10	12.08	15.02	45
8000	10	26.50	31.57	46

tissue. The linear storage requirements and nearly linear time dependence on the number of computational elements offer hope that this method may be superior to finite difference methods on these more elaborate problems.

On the theoretical side it is hoped that probabilists will be challenged to prove that the stochastic process described in § 2 is a model for Nagumo's equation, and to prove convergence. Especially interesting would be a proof of the error law in § 4.4, or of an alternative law. Such work might also give new insights into Nagumo's equation itself, in particular to the question of global stability of the traveling wave solutions.

REFERENCES

[1] A. V. AHO, J. E. HOPCROFT AND J. D. ULLMAN, *The Design and Analysis of Computer Algorithms*, Addison-Wesley, Reading, MA, 1974.

[2] R. CASTEN, H. COHEN AND P. LAGERSTROM, *Perturbation analysis of an approximation to the Hodgin-Huxley theory*, Quart. Appl. Math., 32 (1975), pp. 365-402.

[3] A. J. CHORIN, *Numerical methods for use in combustion modeling*, in Computing Methods in Applied Sciences and Engineering, R. Glowinski and J. L. Lions, eds., North-Holland, Amsterdam, 1980, pp. 229-235.

[4] A. J. CHORIN AND J. E. MARSDEN, *A Mathematical Introduction to Fluid Mechanics*, Springer-Verlag, New York, 1979.

[5] A. K. OPPENHEIM AND A. GHONIEM, *Application of the random element method to one-dimensional flame propagation problems*, AIAA-83-0600, AIAA 21st Aerospace Sciences Meeting, Reno, NV, 1983.

[6] P. C. FIFE AND J. B. MCLEOD, *The approach of solutions of nonlinear diffusion equations to travelling front solutions*, Arch. Rational Mech. Anal., 65 (1977), pp. 335-361.

SIAM J. SCI. STAT. COMPUT.
Vol. 7, No. 4, October 1986

019

CONVERGENCE OF A RANDOM METHOD WITH CREATION OF VORTICITY*

OLE H. HALD†

Abstract. A system of diffusion equations modeling free convection near a wall is solved by a grid free random walk method that involves creation of the vorticity at the boundary. We prove that the pointwise error and the least squares error of the computed solution tend to zero in probability as the time step tends to zero and the number of particles in the random walk increases.

Key words. convergence, random walk, vorticity

AMS(MOS) subject classifications. 65C20, 65N99, 65U05

Introduction. The purpose of this paper is to prove the convergence of a random method for solving a coupled system of diffusion equations. The equations describe free convection of an incompressible fluid. The flow is parallel to a hot vertical wall and is caused by the uneven heating of the fluid. This problem is of interest because it involves creation of vorticity at the boundary. The mathematical solution can be written down explicitly and this fact allows us to estimate the rate of convergence for the random method, both for the expected value and for the variance.

The algorithm was presented by Ghoniem and Sherman in their survey paper on random methods as an illustration of the basic approach, (see [11]). We regard the free convection problem as a model for more complicated problems in fluid dynamics. In increasing order of complexity we have random methods for the reaction-diffusion equation, the Prandtl boundary layer equations and the Navier–Stokes equations, (see Chorin [6], [5], [4]). The basic idea in all three cases is to use a fractional step method. In the first step we solve the nonlinear part of the equations by a deterministic method and in the second step we simulate the diffusion part of the equations by a random walk technique. If the problem involves boundaries we must also satisfy the boundary conditions. This is done in an intermediate step by creating vorticity at the boundary.

The analysis of the convergence of random methods is less advanced than the understanding of the underlying differential equations. In addition, the methods have been applied to physical problems that are so complicated that they, at present, defy analysis. A good example is turbulent combustion, (see Ghoniem, Chorin and Oppenheim [9]). However, Marchioro and Pulvirenti [16] have proved that in two dimensions the random vortex method converges weakly to a weak solution of the Navier–Stokes equations. A similar statement for the Prandtl boundary layer equations can be found in Benefatto and Pulvirenti [1]. For the reaction-diffusion equation Hald [12] has proved convergence of an inefficient version of Chorin's method, whereas Brenier [3] has generalized Chorin's method to nonlinear advection-diffusion equations and proved the convergence of a deterministic version of the algorithm. The convergence of the random methods is not really in doubt but because of the statistical error it is difficult to check the dependence on the various parameters carefully, (see Milinazzo and Saffman [17] and Roberts [18]).

* Received by the editors October 22, 1984, and in revised form July 3, 1985.

† Department of Mathematics, University of California, Berkeley, California 94720. This work was supported in part by U.S. Office of Naval Research under grant N00014-76-C-0316. The author is a Miller Research Professor.

This paper is organized as follows. In § 1 we discuss the physical problem, give the differential equations and write the solution down explicitly. One of the terms in the solution can be interpreted as coming from the creation of vorticity at the boundary. In § 2 we present the numerical method. Since the ordering of the intermediate steps in a fractional step method is arbitrary we have several choices. Here we follow Ghoniem and Sherman [11] as their choice leads to a simple proof, but the other versions have similar accuracies. To prove that the random method converges we must establish two facts. First that the expected value of the computed solution tends to the exact solution as the time step tends to zero, and secondly that the variance goes to zero as we use more and more particles. By approximating the gradient of the solution by a finite number of particles (or vortex sheets) and following these we in effect reduce the variance of the computed solution, which we obtain via integration. This idea is due to Chorin. It is stated in [6], but used implicitly in the random vortex method [4].

In § 3 we will prove the convergence of the method, but only pointwise. Our estimates are very crude, but uniform in space and time as long as our final time is bounded. In the proof we use only elementary probability theory. Two interesting questions remain, namely: "Is the estimate of the variance of the correct order of magnitude?" and "What happens as the viscosity and the thermal diffusivity tend to zero?" In § 4 we will show that the variance of the solution does not go to zero just because the time step goes to zero, even though the number of computational elements at a fixed time increases. This is not as surprising as it sounds. It corresponds to the fact that a finite difference scheme will not converge unless both the mesh length and the time step tend to zero. Finally in § 5 we use nonuniform bounds for the expected value and for the variance to estimate the global error of the computed solution in a least squares sense. We can then study the effect of the viscosity and diffusivity. Our techniques can also be used to prove convergence for some of the algorithms in Ghoniem and Oppenheim [10] and Ghoniem and Sherman [11].

1. The differential equations. In this section we will present the differential equations for a one-dimensional model of free convection and give the mathematical solution. We consider an incompressible fluid in the half space $x>0$ and assume that the velocity u and the temperature T depend only on the distance x to the wall (at $x=0$). Our differential equations are (see [11])

$$u_t = \nu u_{xx} + g\beta' T, \tag{1.1}$$

$$T_t = \alpha T_{xx}. \tag{1.2}$$

Here ν is the kinematic viscosity, g is the acceleration due to gravity, β' is the coefficient of thermal expansion of the fluid and α is the coefficient of thermal diffusivity. The first equation is derived from the Navier–Stokes equations for a fluid in a gravitational field while the second is a simplified version of the equation for heat transfer, (see [15, p. 189 and p. 213]). To complete the description of (1.1) and (1.2) we need the initial conditions

$$u(x,0)\equiv 0, \qquad T(x,0)\equiv 0$$

for $0<x<\infty$. For $t>0$ we adopt the boundary conditions

$$u(0,t)=0, \qquad u(x,t)\to 0 \quad \text{as } x\to\infty,$$

$$T(0,t)=1, \qquad T(x,t)\to 0 \quad \text{as } x\to\infty.$$

Throughout this paper we assume that $g\beta' = 1$. This simplifies the writing. The solution of the heat equation (1.2) with the appropriate initial and boundary conditions is

$$T(x, t) = \int_x^\infty \frac{2\, e^{-y^2/4\alpha t}}{\sqrt{4\pi\alpha t}}\, dy. \tag{1.3}$$

We can now find the solution of the inhomogeneous equation (1.1). By using Duhamel's principle and the Green's function for the heat equation on a half line and with a Dirichlet boundary condition at the origin (see Hellwig [13, p. 25 and p. 53]), we obtain

$$u(x, t) = \int_0^t \int_0^\infty \left(\frac{e^{-(x-y)^2/4\nu(t-s)}}{\sqrt{4\pi\nu(t-s)}} - \frac{e^{-(x+y)^2/4\nu(t-s)}}{\sqrt{4\pi\nu(t-s)}} \right) T(y, s)\, dy\, ds. \tag{1.4}$$

This formula shows that the velocity u is positive throughout the fluid. To analyze the solution further we introduce the vorticity ξ and the heat flux q by

$$u(x, t) = \int_x^\infty \xi(y, t)\, dy \qquad T(x, t) = \int_x^\infty q(y, t)\, dy.$$

Thus $-\xi$ and $-q$ are the gradients of u and T. Let $\varphi(x, \sigma^2)$ be the normal density with mean 0 and variance σ^2, i.e. $\varphi = (2\pi\sigma^2)^{-1/2} \exp(-x^2/2\sigma^2)$. Since $\partial_x \varphi(x-y) = -\partial_y \varphi(x-y)$ it follows from (1.4) that

$$\xi(x, t) = \int_0^t \int_0^\infty \partial_y [\varphi(x-y, 2\nu(t-s)) + \varphi(x+y, 2\nu(t-s))] T(y, s)\, dy\, ds.$$

After integrating by parts we arrive at

$$\xi(x, t) = -\int_0^t 2\varphi(x, 2\nu(t-s))\, ds + \int_0^t 2\varphi(x, 2\alpha s + 2\nu(t-s))\, ds. \tag{1.5}$$

Here we have used the identity

$$\begin{aligned} &\varphi(x-z, a+b) + \varphi(x+z, a+b) \\ &\quad = \int_0^\infty [\varphi(x-y, a) + \varphi(x+y, a)][\varphi(y-z, b) + \varphi(y+z, b)]\, dy \end{aligned} \tag{1.6}$$

with $z = 0$. This follows from the fact that the convolution of two normal densities with means x, z and variances a, b is the normal density with mean $x+z$ and variance $a+b$, (see [8, p. 45]). Finally, by integrating both sides in equation (1.5) with respect to x we get

$$u(x, t) = -\int_0^t \int_x^\infty \frac{2\, e^{-y^2/4\nu(t-s)}}{\sqrt{4\pi\nu(t-s)}}\, dy\, ds + \int_0^t \int_x^\infty \frac{2\, e^{-y^2/(4\alpha s + 4\nu(t-s))}}{\sqrt{4\pi\alpha s + 4\pi\nu(t-s)}}\, dy\, ds. \tag{1.7}$$

It turns out that the expected value of the computed velocity is a discretized version of the right-hand side of (1.7). Moreover, the numerical method reveals that the first term in the solution is due to the creation of vorticity at the boundary. After several changes of variables in (1.7) we find that $u(x, t) = tf(x/\sqrt{2\alpha t}, \nu/\alpha)$, where f can be given explicitly. This result is close but not identical to Illingworth's form of the solution, (see [14]). It shows that the maximum of u is a linear function of time and depends only on the Prandtl number ν/α.

2. The random method. In this section we present the numerical method following Ghoniem and Sherman [11]. The basic idea is to approximate the gradients of u and

T by a finite number of particles and let these particles undergo random walk. Thus we solve the differential equations

$$\xi_t = \nu\xi_{xx} + q, \tag{2.1}$$

$$q_t = \alpha q_{xx}. \tag{2.2}$$

We begin with the heat equation (2.2). At time $t=0$ we place N particles at the origin. We assume that each particle has mass $1/N$. In each step we let the particles jump by an amount which is drawn from a Gaussian distribution with mean 0 and variance $2\alpha\Delta t$. If a particle lands at a point $x<0$ we reflect it across the wall to $|x|$. In probability language this is called a reflecting barrier and corresponds to a Neumann boundary condition for the differential equations. To describe the process mathematically we let X_j^n be the position of the jth particle at time $t=n\Delta t$. Let $\{X\}$ be a collection of independent normally distributed random variables with mean 0 and variance $2\alpha\Delta t$. Then

$$X_j^n = |X_j^{n-1} + X| \tag{2.3}$$

with $X_j^0 = 0$ for $j = 1, \cdots, N$. We denote the approximations to the heat flux and the temperature by Q and $\Theta(\vartheta\varepsilon\rho\mu\eta = \text{heat})$. To express these approximations we let δ be the Dirac delta function and let H be the Heaviside function. Here $H(x)=1$ if $x \geqq 0$ and 0 otherwise. At $t=n\Delta t$ we set

$$Q(x,t) = \frac{1}{N}\sum_{j=1}^{N} \delta(X_j^n - x),$$

$$\Theta(x,t) = \frac{1}{N}\sum_{j=1}^{N} H(X_j^n - x). \tag{2.4}$$

Thus for each (x, t), Q is a random measure and Θ is a random variable. The temperature at a point x is the number of particles to the right of x times the mass of a single particle. Note that our definition of the Heaviside function ensures that the approximate temperature distribution satisfies the boundary condition at $x=0$ exactly.

The complete fractional step method for (2.1) and (2.2) consists of four intermediate steps

$$q_t = \alpha q_{xx}, \quad \xi_t = \nu\xi_{xx}, \quad \xi_t = q, \quad \int_0^\infty \xi = 0.$$

We have already discussed the first step. The last three constitute a fractional step method for solving (2.1). Here $\int_0^\infty \xi = 0$ is shorthand for the creation of vorticity at the boundary. At time $t=n\Delta t$ we approximate the vorticity ξ by nN vortex sheets of strength $\Delta t/N$ and an equal number of strength $-\Delta t/N$. Here a vortex sheet of strength s is a plane parallel to the wall such that the velocity decreases by the amount s when we cross the sheet in the direction of increasing values of x. The velocity $u(x, t)$ is approximated by the total strength of the vortex sheets in the interval $[x, \infty)$. Thus $u(0, t) = 0$. Note the change in language: We use particles to calculate the temperature and sheets to calculate the velocity. We simulate the diffusion $\xi_t = \nu\xi_{xx}$ by a random walk. The displacements are drawn from a Gaussian distribution with mean 0 and variance $2\nu\Delta t$. If a vortex sheet lands at $x<0$ we reflect it back into the fluid. Next we solve $\xi_t = q$ by Euler's method, i.e.

$$\xi(x, t+\Delta t) = \xi(x, t) + \Delta t q(x, t).$$

At the position of each heat particle we introduce a vortex sheet of strength $\Delta t/N$.

The velocity u at $x=0$ will therefore be Δt. To satisfy the boundary condition $u=\int_0^\infty \xi = 0$ we create N vortex sheets of strength $-\Delta t/N$ at the wall. This completes the description of the algorithm. Note that we have no sheets in the beginning of the first step and in the end of the last step we have N sheets at the origin.

To describe the solution of (2.1) more precisely we let $Y_j^{m,n}$ be the position at time $t=n\,\Delta t$ of one of the N vortex sheets that were created at the origin at time $m\,\Delta t$. Similarly we let $Z_j^{m,n}$ be the positions at time $n\,\Delta t$ of one of the N vortex sheets that were introduced into the flow at time $m\,\Delta t$. Thus $Y_j^{m,m}=0$ and we set $Z_j^{m,m}=X_j^m$ for $j=1,\cdots,N$ and $m=1,\cdots,n$. Let $\{Y\}$ be a collection of independent normally distributed random variables with mean 0 and variances $2\nu\,\Delta t$. Then

$$Y_j^{m,n}=|Y_j^{m,n-1}+Y|, \tag{2.5}$$

$$Z_j^{m,n}=|Z_j^{m,n-1}+Y|, \tag{2.6}$$

for $n>m$. Let n be fixed. Since the Y's are independent we conclude that all the $Y_j^{m,n}$ are independent. Moreover, the class $\{Y_j^{m,n}\}$ is independent of the class $\{Z_j^{m,n}\}$. However, two random variables $Z_j^{m,n}$ with the same j but different m's are not independent, because the two vortex sheets were spawned by the same heat particle, albeit at different times. We illustrate the dependence in Fig. 1. On the other hand, the classes $\{Z_1^{m,n}\},\cdots,\{Z_N^{m,n}\}$ are mutually independent. If Ξ and U denote our approximations to the vorticity and the velocity then

$$\Xi(x,t)=\frac{1}{N}\sum_{j=1}^{N}\left(-\Delta t\sum_{m=1}^{n}\delta(Y_j^{m,n}-x)+\Delta t\sum_{m=1}^{n}\delta(Z_j^{m,n}-x)\right),$$

$$U(x,t)=\frac{1}{N}\sum_{j=1}^{N}\left(-\Delta t\sum_{m=1}^{n}H(Y_j^{m,n}-x)+\Delta t\sum_{m=1}^{n}H(Z_j^{m,n}-x)\right). \tag{2.7}$$

This shows that the computed velocity U at (x,t) is really the number of vortex sheets to the right of x times the strength of a single vortex sheet. Here the vortex sheets are counted as $+1$ if they were introduced into the flow and as -1 if they were created at the origin. Since the random variables with a fixed n but different j's are independent we see that U is an average of N independent, identically distributed random variables. Thus instead of using N heat particles with mass $1/N$ we may as well use one particle with unit mass and average over N trials. We can therefore assume that $N=1$ and this simplifies our analysis considerably.

3. Pointwise convergence. In this section we will prove the convergence of the random method for the temperature and for the velocity. Because of the dependence between some of the particles we shall be satisfied with very crude estimates for the expected value and for the variance (here denoted by E and var).

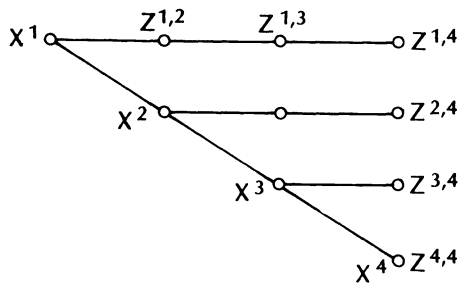


FIG. 1. *Dependence between the vortex sheets that are introduced into the flow.*

THEOREM 1. *Let $u(x, t)$ and $T(x, t)$ be the solution of* (1.1) *and* (1.2) *with $g\beta' = 1$. Let $U(x, t)$ and $\Theta(x, t)$ be the random solutions given by* (2.7) *and* (2.4). *If (x, t) is fixed and $t = n\,\Delta t > 0$ then*

$$|E\Theta - T| = 0, \qquad |EU - u| \leqq \Delta t,$$

$$\operatorname{var} \Theta \leqq \frac{1}{4N}, \qquad \operatorname{var} U \leqq \frac{t^2}{3N}.$$

Remark. The factor t^2 in the variance corresponds to a factor t in the standard deviation. Since the maximum velocity is a linear function of t we expect that the error bound is at least proportional to t. By using Chebyshev's inequality (see [8, p. 149]) we can estimate the error in probability . For each $k > 1$ and each (x, t) each of the inequalities

$$|\Theta(x, t) - T(x, t)| \leqq \frac{k}{\sqrt{4N}},$$

$$|U(x, t) - u(x, t)| \leqq \Delta t + \frac{kt}{\sqrt{3N}},$$

is satisfied with a probability greater than $1 - k^{-2}$. If $k = 3$ the bounds should hold in roughly 8 out of 9 trials. Note that our estimates do not depend upon the viscosity or the diffusivity. A similar phenomenon has been observed by Sethian [19] in his simulation of two-dimensional viscous incompressible flow over a half step. The amount of work for our algorithm is roughly $Nt^2/\Delta t^2$. Thus we get the smallest error bound for the computed velocity if N is proportional to $t^2/\Delta t^2$. To obtain high accuracy in a particular run we need that N is large. If we only have a few particles then there is no point in taking small time steps.

Proof. We begin with the convergence for the temperature. This is the easiest part of the proof. We assume that $N = 1$ and drop the subscript j. We will show that the probability that a heat particle lies in the interval $[0, x]$ at time $m\,\Delta t$ is

$$P(X^m \leqq x) = \int_0^x 2\varphi(y, 2\alpha m\,\Delta t)\,dy. \tag{3.1}$$

This is proved by induction. For $m = 1$ $X^m = |X|$. Thus (3.1) follows from the symmetry of the density function for X. We assume next that (3.1) holds for some $m \geqq 1$. It follows from (2.3) that the density for X^{m+1} is $h(x) + h(-x)$, where h is the density for $X^m + X$. Since X and X^m are independent we have

$$h(x) = \int_0^\infty \varphi(x - y, a) 2\varphi(y, b)\,dy$$

where $a = 2\alpha\,\Delta t$ and $b = 2\alpha m\,\Delta t$, (see [8, p. 7]). By using (1.6) with $z = 0$ we see that the density function for X^{m+1} is $2\varphi(x, a + b)$. This completes the induction.

We can now prove that the temperature converges. Let $t = n\,\Delta t$ be fixed and consider the random variable $H(X^n - x)$. It is one if $X^n \geqq x$ and zero otherwise. Thus

$$EH(X^n - x) = P(X^n \geqq x), \qquad \operatorname{var} H(X^n - x) = P(X^n \geqq x) P(X^n < x). \tag{3.2}$$

Since Θ consists of N independent random variables and the variance of $H(X^n - x)$

is always less than $\frac{1}{4}$ we conclude from (2.4), (3.1) and (1.3) that

$$E\Theta(x,t)=\frac{1}{N}\sum_{j=1}^{N}\int_x^\infty \frac{2\,e^{-y^2/4\alpha t}}{\sqrt{4\pi\alpha t}}\,dy = T(x,t),$$

$$\operatorname{var}\Theta(x,t)=\frac{1}{N^2}\sum_{j=1}^{N}\int_x^\infty \frac{2\,e^{-y^2/4\alpha t}}{\sqrt{4\pi\alpha t}}\,dy\int_0^x \frac{2\,e^{-y^2/4\alpha t}}{\sqrt{4\pi\alpha t}}\,dy \leqq \frac{1}{4N}. \tag{3.3}$$

Our estimate for the variance is the best possible uniform bound. In fact we have equality at the point x where $P(X^n \geqq x) = P(X^n < x)$. On the other hand it is a severe overestimate if $x/\sqrt{2\alpha t}$ is either very small or very large.

We will now discuss the approximation of the velocity. We observe first that it is not essential in the derivation of (3.1) that all X's have the same density. What matters is that the X's are independent and normally distributed. Under this assumption the variances are additive. By using this observation we conclude from (2.5) and (2.6) that

$$P(Y^{m,n} \leqq x) = \int_0^x 2\varphi(y, 2\nu(n-m)\,\Delta t)\,dy, \tag{3.4}$$

$$P(Z^{m,n} \leqq x) = \int_0^x 2\varphi(y, 2\alpha m\,\Delta t + 2\nu(n-m)\,\Delta t)\,dy. \tag{3.5}$$

Here we have suppressed the subscript j as the distributions do not depend on j. The first result holds for $m = 1, \cdots, n-1$, the second for $m = 1, \cdots, n$. In addition $Y^{n,n} = 0$. We can now calculate the expected value of the velocity. Let $x > 0$ and $t = n\,\Delta t$ where $n \geqq 2$. By using (2.7) and (3.2) with $Y^{m,n}$ and $Z^{m,n}$ instead of X^n we conclude from (3.4) and (3.5) that

$$\begin{aligned} EU(x,t) = &-\Delta t\sum_{m=1}^{n-1}\int_x^\infty \frac{2\,e^{-y^2/(4\nu(n-m)\Delta t)}}{\sqrt{4\pi\nu(n-m)\,\Delta t}}\,dy \\ &+\Delta t\sum_{m=1}^{n}\int_x^\infty \frac{2\,e^{-y^2/(4\alpha m\Delta t+4\nu(n-m)\Delta t)}}{\sqrt{4\pi\alpha m\,\Delta t+4\pi\nu(n-m)\,\Delta t}}\,dy. \end{aligned} \tag{3.6}$$

The sums in (3.6) approximate the integrals in the exact solution (1.7). Since the $Y_j^{m,n}$ are created at the origin we can now understand why the first term in (1.7) must be regarded as due to the creation of vorticity at the boundary. To estimate the error in the expected value of U we let $f(s)$ and $g(s)$ be the integrals in (3.6) as functions of $s = m\,\Delta t$. Note that we suppress the dependence on x. By using a change of variables we see that

$$\begin{aligned} f(s) &= \int_{x/\sqrt{2\nu(t-s)}}^\infty \frac{2\,e^{-y^2/2}}{\sqrt{2\pi}}\,dy \\ g(s) &= \int_{x/\sqrt{2(\alpha-\nu)s+2\nu t}}^\infty \frac{2\,e^{-y^2/2}}{\sqrt{2\pi}}\,dy. \end{aligned} \tag{3.7}$$

We consider the two cases $\nu \leqq \alpha$ and $\nu > \alpha$ separately. In the first case f is a decreasing function of s while g is increasing. This implies that the first sum in (3.6) is a lower Riemann sum for the integral $\int_0^t f(s)\,ds$ while the second sum is an upper Riemann sum for the integral $\int_0^t g(s)\,ds$. We remember now that the error in approximating an integral by a Riemann sum is less than the mesh length times the total variation of the function. But our functions are monotone. Hence

$$0 \leqq EU - u \leqq \Delta t f(0) + \Delta t(g(t) - g(0)) = \Delta t g(t)$$

where we have used that $f(0) = g(0)$. If $\nu > \alpha$ then g is a decreasing function of s. The second sum in (3.6) is therefore a lower Riemann sum for $\int_0^t g(s)\,ds$. Consequently

$$-\Delta t g(0) < -\Delta t(g(0) - g(t)) \leqq EU - u \leqq \Delta t f(0).$$

By combining our estimates and using that $0 < g < 1$ we conclude that

$$|EU(x, t) - u(x, t)| \leqq \Delta t \max(g(0), g(t)) < \Delta t. \tag{3.8}$$

To complete the proof we must estimate the variance of the computed velocity. For simplicity we set $N = 1$ and drop the index j. Since the $Y^{m,n}$ are mutually independent and independent of the $Z^{m,n}$ it follows from (2.7) and Cauchy-Schwarz inequality that if $x > 0$ then

$$\begin{aligned} \operatorname{var} U(x, t) &= \operatorname{var} -\Delta t \sum_{m=1}^{n-1} H(Y^{m,n} - x) + \operatorname{var} \Delta t \sum_{m=1}^{n} H(Z^{m,n} - x) \\ &\leqq \Delta t^2 \sum_{m=1}^{n-1} \operatorname{var} H(Y^{m,n} - x) + \Delta t^2 n \sum_{m=1}^{n} \operatorname{var} H(Z^{m,n} - x). \end{aligned} \tag{3.9}$$

By using that the variance of a zero-one random variable is less than $\frac{1}{4}$ we conclude that var $U \leqq 5t^2/16$ if $N = 1$. If $N > 1$ then U consists of N independent, identically distributed random variables and in this case var $U(x, t)$ is less than $5t^2/(16N)$. The proof simplifies if $n = 1$. Note that our estimates also hold at $x = 0$ because the computed temperature and the computed velocity satisfy the boundary conditions exactly. This completes the proof.

4. Improvements and limitations. Our bound for the variance of the random velocity does not tend to zero as the time step tends to zero. This is not due to an imperfect technique. In this section we will show that all the covariances of the $Z^{m,n}$ are positive and this excludes cancellation in the expression for the variance. The problem can be circumvented by modifying the algorithm, but the modification is not in the spirit of random solution of differential equations.

We begin with a reexamination of (3.9). The variance of the sum of the dependent random variables in (3.9) is equal to

$$\Delta t^2 \sum_{m=1}^{n} \operatorname{var} H(Z^{m,n} - x) + 2\Delta t^2 \sum_{l=1}^{n-1} \sum_{m=l+1}^{n} \operatorname{cov}(H(Z^{l,n} - x) H(Z^{m,n} - x)), \tag{4.1}$$

(see [7, p. 216]). We denote the covariances by $C^{l,m,n}$. Let $r = l\,\Delta t$, $s = m\,\Delta t$ and $t = n\,\Delta t$. We will show that

$$C^{l,m,n} = C(2\alpha r, 2\nu(t - r), 2\alpha(s - r) + 2\nu(t - s))$$

where C is a continuous positive function. This implies that the double sum in (4.1) tends to the double integral

$$2 \int_0^t \int_r^t C(2\alpha r, 2\nu(t - r), 2\alpha(s - r) + 2\nu(t - s))\,ds\,dr. \tag{4.2}$$

This expression is positive. Since the first sum in (3.9) and the first sum in (4.1) are $O(\Delta t)$ we conclude that the variance of the computed velocity tends to the expression in (4.2) times $1/N$ as the mesh length tends to zero. We are therefore led to the following observation.

OBSERVATION 1. *The bounds for the variances in Theorem* 1 *have the correct dependence on the number of particles.*

The calculation of the covariances $C^{l,m,n}$ can best be understood by an example. Let

$$Z^l = ||X + Y_1| + Y_2|, \qquad Z^m = ||X + V_1| + V_2|$$

where the Y_i and the V_i are independent normally distributed random variables with mean 0 and variances a_i and b_i. We assume that X has density $2\varphi(x, c)$ for $x \geqq 0$. If $a_1 = a_2 = b_2 = 2\nu\,\Delta t$ and $2b_1 = c = 4\alpha\,\Delta t$ then we have the situation indicated in Fig. 1 with $Z^l = Z^{2,4}$ and $Z^m = Z^{3,4}$. Finally we introduce the functions

$$F_l(z) = EH(||z + Y_1| + Y_2| - x), \tag{4.3}$$

$$F_m(z) = EH(||z + V_1| + V_2| - x),$$

which we shall evaluate explicitly. Note that F_l and F_m are positive. Since X, Y_i and V_i are independent it follows from the definition of the covariance that

$$C^{l,m,n} = \int_0^\infty 2\varphi(z)F_l(z)F_m(z)\,dz - \int_0^\infty 2\varphi(y)F_l(y)\,dy \int_0^\infty 2\varphi(z)F_m(z)\,dz$$

$$= 2\int_0^\infty \int_0^\infty \varphi(y)\varphi(z)[F_l(y) - F_l(z)][F_m(y) - F_m(z)]\,dy\,dz. \tag{4.4}$$

Here we have suppressed c in $\varphi(z, c)$. We will show that F_l and F_m are increasing functions of z. The product $[\cdots][\cdots]$ in (4.4) is therefore positive and this implies that $C^{l,m,n} > 0$ as claimed. Since the Y's are independent we can compute the expected value in (4.3) as

$$F_l(z) = \int_{-\infty}^\infty \int_{-\infty}^\infty H(||z + y_1| + y_2| - x)\varphi(y_1, a_1)\varphi(y_2, a_2)\,dy_1\,dy_2.$$

We attack the innermost variable first. Let $z + y_1 = \eta_1$. By splitting the integral into $\eta_1 > 0$ and $\eta_1 < 0$ we find after a change of variables that

$$F_l(z) = \int_{-\infty}^\infty \int_0^\infty H(|\eta_1 + y_2| - x)[\varphi(\eta_1 - z, a_1) + \varphi(\eta_1 + z, a_1)]\varphi(y_2, a_2)\,d\eta_1\,dy_2.$$

Next we interchange the order of integration, set $\eta_1 + y_2 = \eta_2$, split the integral into $\eta_2 > 0$ and $\eta_2 < 0$ and get

$$F_l(z) = \int_0^\infty \int_0^\infty H(\eta_2 - x)[\varphi(\eta_2 - \eta_1, a_2) + \varphi(\eta_2 + \eta_1, a_2)]$$

$$\times [\varphi(\eta_1 - z, a_1) + \varphi(\eta_1 + z, a_1)]\,d\eta_2\,d\eta_1.$$

By interchanging the order of integration, using the definition of the Heaviside function and (1.6), we obtain

$$F_l(z) = \int_x^\infty [\varphi(\eta - z, a) + \varphi(\eta + z, a)]\,d\eta \tag{4.5}$$

where $a = a_1 + a_2$. Since x and z are positive (4.5) implies that $F_l(z)$ is an increasing function of z. The same result holds for $F_m(z)$. Equation (4.5) gives the general result. If we return to the original variables then it follows from (2.3) and (2.6) that $Z^{l,n}$ and $Z^{m,n}$ depend on X^l if $l < m$. Consequently $F_l(z)$ is given by (4.5) with $a = 2\nu(n - l)\,\Delta t$. For $F_m(z)$ the variance a is replaced by $b = 2\alpha(m - l)\,\Delta t + 2\nu(n - m)\,\Delta t$. Finally (3.1) shows that $c = 2\alpha l\,\Delta t$. By combining these observations we can interpret the double

integral in (4.4) as the evaluation of a positive function at the grid points of a two dimensional grid. This completes the proof of Observation 1.

Our analysis shows that the variance of the computed velocity would be of the order $\Delta t/N$ if the vortex sheets were independent. This can be arranged. At the mth time step we let the particles start at the origin and jump according to a Gaussian distribution with mean 0 and variance $2\alpha m\,\Delta t$. Those that land at $x<0$ are reflected across the boundary. The costs for the two algorithms are identical. The modified version of the algorithm uses the fact that we can simulate the solution of the heat equation by taking one large step instead of many small. However, to take one large step is contrary to the spirit of numerical solution of differential equations. We will therefore not discuss this variant any further.

5. Convergence in L^2. Our estimates in Theorem 1 say nothing about what happens at several points at the same time. From a practical point of view we would like to know that the maximum error in the computed solution tends to zero in probability. It is possible, by using Kolmogorov's theory for empirical distribution functions, to estimate the error in the maximum norm asymptotically, as the number of particles tend to infinity (cf. Billingsley [2, p. 104]). However, I have been unable to find explicit upper bounds for a finite number of particles. In this section we will show that it is unlikely that the computed solution differs from the exact solution by a moderate amount on a large set or by a large amount on a moderate set. The error may still be large on a small set. This is the content of the next theorem.

THEOREM 2. *Let $u(x,t)$ and $T(x,t)$ be the solutions of* (1.1) *and* (1.2) *with $g\beta'=1$. Let $U(x,t)$ and $\Theta(x,t)$ be the random solutions given by* (2.7) *and* (2.4). *Set $\beta=\nu/\alpha$. If $t=n\,\Delta t>0$ and $k>1$ then*

$$1-\frac{1}{k^2}\leqq P\left(\frac{\|\Theta-T\|}{\|T\|}\leqq\frac{k}{\sqrt{N}}\right),$$

$$1-\frac{1}{k^2}\leqq P\left(\frac{\|U-u\|}{\|u\|}\leqq(1+\beta)\left(1+\frac{1+\sqrt{\beta}}{\sqrt{1+\beta}}\right)\left(\frac{\Delta t}{t}+\frac{k}{\sqrt{N}}\right)\right).$$

Remark. The factor that depends on the Prandtl number β is an increasing function. It is always greater than 2 and less than 5 if $\beta\leqq 1$. It is less than 20 if $\beta\leqq 7.75$. If $\Delta t=t/n$, $N=n^4$, $k=n$ and $\beta\leqq 1$ then we have

$$P(\|U-u\|>10\|u\|/n)\leqq 1/n^2.$$

It is surprising that our bound improves as t increases. If Δt is small and t is comparable to Δt then the absolute error is small with high probability, but the relative error could be large. To prove Theorem 2 we need two technical lemmas.

LEMMA 1. *If $a>0$ and $b>0$ then*

$$\int_0^\infty\int_{x/\sqrt{a}}^\infty\frac{2\,e^{-y^2/2}}{\sqrt{2\pi}}\,dy\int_0^{x/\sqrt{b}}\frac{2\,e^{-y^2/2}}{\sqrt{2\pi}}\,dy\,dx=\sqrt{2/\pi}(\sqrt{a+b}-\sqrt{b})$$

$$\int_0^\infty\int_{x/\sqrt{a}}^\infty\frac{2\,e^{-y^2/2}}{\sqrt{2\pi}}\,dy\int_{x/\sqrt{b}}^\infty\frac{2\,e^{-y^2/2}}{\sqrt{2\pi}}\,dy\,dx=\sqrt{2/\pi}(\sqrt{a}+\sqrt{b}-\sqrt{a+b}).$$

Proof. Let I_1 denote the first integral. Change the variable x to $x=z\sqrt{a}$ and set $f(z)=\int_0^{\beta z}\exp(-y^2/2)\,dy$ where $\beta=\sqrt{a/b}$. By interchanging the order of integration in I_1 we get

$$I_1=\frac{4\sqrt{a}}{2\pi}\int_0^\infty e^{-y^2/2}\int_0^y f(z)\,dz\,dy=\frac{2\sqrt{a}}{\pi}\int_0^\infty e^{-y^2/2}\left[y\int_0^{\beta y}e^{-\eta^2/2}\,d\eta+\frac{1}{\beta}\,e^{-\beta^2y^2/2}-\frac{1}{\beta}\right]dy.$$

To compute the double integral we integrate by parts. This yields

$$I_1=\frac{2\sqrt{a}}{\pi}\left[\int_0^\infty e^{-y^2/2}\beta\, e^{-\beta^2y^2/2}\,dy+\frac{1}{\beta}\int_0^\infty e^{-(1+\beta^2)y^2/2}\,dy-\frac{1}{\beta}\int_0^\infty e^{-y^2/2}\,dy\right].$$

The value of I_1 can now be computed. To evaluate the second integral in Lemma 1 we note that $\int_{x/\sqrt{b}}^\infty 2\varphi = 1-\int_0^{x/\sqrt{b}} 2\varphi$ and use the value of I_1. This completes the proof.

In Theorem 2 we estimate the relative error in the computed solution. We must therefore find lower bounds for the L^2 norm of the temperature and of the velocity. Since T and $t^{-1}u$ are functions of $x/\sqrt{2\alpha t}$ it follows that the L^2 norms of T and u are proportional to $(2\alpha t)^{1/4}$ and $t(2\alpha t)^{1/4}$. This simple argument does not give the value of the constants nor does it indicate the dependence on the Prandtl number ν/α. By using Lemma 1 we can calculate the L^2 norms explicitly.

LEMMA 2. *Let $u(x,t)$ and $T(x,t)$ be the solution of* (1.1) *and* (1.2) *with $g\beta'=1$. Then*

$$\|T\|^2=\frac{2(\sqrt{2}-1)}{\sqrt{\pi}}\sqrt{2\alpha t},$$

$$\|u\|^2=\frac{32t^{5/2}}{15\sqrt{\pi}}\alpha^2\frac{(\sqrt{2}-1)(\sqrt{\alpha}+\sqrt{\nu})^2+\sqrt{2}(\sqrt{\alpha}+\sqrt{\nu})\sqrt{\alpha+\nu}+\sqrt{2}(\alpha+\nu)}{(\sqrt{\alpha}+\sqrt{\nu}+\sqrt{\alpha+\nu})^2(\sqrt{\alpha}+\sqrt{\nu})^2(\sqrt{\alpha}+\sqrt{\nu}+\sqrt{2}\sqrt{\alpha+\nu})}.$$

Proof. We begin with the temperature. By using (1.3) and Lemma 1 with $a=b=2\alpha t$ we find that

$$\|T\|^2=\int_0^\infty\left(\int_{x/\sqrt{2\alpha t}}^\infty\frac{2\,e^{-y^2/2}}{\sqrt{2\pi}}\,dy\right)^2dx=\sqrt{2/\pi}(2\sqrt{2\alpha t}-\sqrt{2\cdot 2\alpha t})$$

from which the first statement follows. Let next $u=-A+B$ where A and B are the integrals in (1.7). We replace the variable $t-s$ in A by s and shall compute $\|u\|^2$ as $\int A^2+B^2-2AB$. Fubini's theorem and Lemma 1 yield

$$\int_0^\infty A^2\,dx=\int_0^t ds\int_0^t d\sigma\int_0^\infty dx\int_{x/\sqrt{2\nu s}}^\infty\frac{2\,e^{-y^2/2}}{\sqrt{2\pi}}\,dy\int_{x/\sqrt{2\nu\sigma}}^\infty\frac{2\,e^{-y^2/2}}{\sqrt{2\pi}}\,dy$$

$$=\frac{2}{\sqrt{\pi}}\int_0^t ds\int_0^t d\sigma\,[\sqrt{\nu s}+\sqrt{\nu\sigma}-\sqrt{\nu s+\nu\sigma}].$$

A similar calculation can be carried out for $\int B^2$ and $\int 2AB$ and by combining the results we get

$$\|u\|^2=\frac{2}{\sqrt{\pi}}\int_0^t ds\int_0^t d\sigma\,[2(\nu s+(\alpha-\nu)\sigma+\nu t)^{1/2}-(\nu s+\nu\sigma)^{1/2}$$

$$-((\alpha-\nu)s+\nu t+(\alpha-\nu)\sigma+\nu t)^{1/2}]$$

$$=\frac{8t^{5/2}}{15\sqrt{\pi}}\left[2\frac{(\alpha+\nu)^{5/2}-\alpha^{5/2}-(2\nu)^{5/2}+\nu^{5/2}}{\nu(\alpha-\nu)}-\frac{(2\nu)^{5/2}-2\nu^{5/2}}{\nu^2}\right.$$

$$\left.-\frac{(2\alpha)^{5/2}-2(\alpha+\nu)^{5/2}+(2\nu)^{5/2}}{(\alpha-\nu)^2}\right].$$

It is not obvious from this formula what happens as $\alpha-\nu$ tends to zero or even that $\|u\|^2$ is positive. The rest of the proof therefore consists in transforming the expression into a form which we can work with. Since $\nu^{-1}+(\alpha-\nu)^{-1}=\alpha\nu^{-1}(\alpha-\nu)^{-1}$ it follows that

$$\|u\|^2=\frac{16t^{5/2}}{15\sqrt{\pi}}\frac{\alpha}{\nu}\left[\frac{(\alpha+\nu)^{5/2}-2\sqrt{2}\alpha\nu(\sqrt{\alpha}+\sqrt{\nu})}{(\alpha-\nu)^2}-\frac{\alpha\sqrt{\alpha}-\nu\sqrt{\nu}}{\alpha-\nu}\right].$$

By writing $(\alpha+\nu)^2$ as $(\alpha-\nu)^2+4\alpha\nu$ and using that $\alpha-\nu$ is equal to $(\sqrt{\alpha}-\sqrt{\nu})\times(\sqrt{\alpha}+\sqrt{\nu})$ we obtain

$$\|u\|^2=\frac{16t^{5/2}}{15\sqrt{\pi}}\frac{\alpha}{\nu}\left[\sqrt{\alpha+\nu}+\frac{2\sqrt{2}\alpha\nu}{\sqrt{\alpha}+\sqrt{\nu}}\frac{1}{(\sqrt{\alpha+\nu}+\sqrt{2\alpha})(\sqrt{\alpha+\nu}+\sqrt{2\nu})}-\frac{\alpha+\sqrt{\alpha}\sqrt{\nu}+\nu}{\sqrt{\alpha}+\sqrt{\nu}}\right].$$

We can now put all terms in $[\cdots]$ over a common denominator. Since $2\sqrt{\alpha\nu}$ equals $(\sqrt{\alpha}+\sqrt{\nu})^2-(\sqrt{\alpha+\nu})^2$ we can express the numerator as $(\alpha+\nu)^2$ times a fourth order polynomial in $\gamma=(\sqrt{\alpha}+\sqrt{\nu})/\sqrt{\alpha+\nu}$. After factoring this polynomial we have

$$\|u\|^2=\frac{16t^{5/2}}{15\sqrt{\pi}}\frac{\alpha}{\nu}\sqrt{\alpha+\nu}(\gamma-1)^2\frac{(\sqrt{2}-1)\gamma^2+\sqrt{2}\gamma+\sqrt{2}}{2\gamma^2(\gamma+\sqrt{2})}. \tag{5.1}$$

The formula in Lemma 2 can now be derived from (5.1) by rationalizing the numerator in $(\gamma-1)^2$. This completes the proof.

The expression for $\|u\|^2$ in Lemma 2 is too complicated to be useful. It is better to have a simple approximation. Since $1<\gamma\leqq\sqrt{2}$ it follows that the last quotient in (5.1) is greater than $\frac{3}{8}$ and less than $\frac{1}{2}(7-4\sqrt{2})$. We can therefore estimate the L^2 norm of u by

$$\frac{8t^2}{5\sqrt{\pi}}\sqrt{\alpha t+\nu t}(1+\beta)^{-2}\left(1+\frac{1+\sqrt{\beta}}{\sqrt{1+\beta}}\right)^{-2}\leqq\|u\|^2\leqq 2\cdot\text{the lower bound}. \tag{5.2}$$

Here $\beta=\nu/\alpha$ is the Prandtl number and we have simplified $\beta^{-1}(\gamma-1)^2$ by rationalizing the numerator. The lower bound in (5.2) is best possible because we have equality for $\alpha=\nu$.

Proof of Theorem 2. We begin with the temperature. Since $T=E\Theta$ it follows from Chebyshev's inequality (see [8, p. 149]) that

$$P\left(\|\Theta-T\|>\frac{k\|T\|}{\sqrt{N}}\right)\leqq\frac{E(\|\Theta-E\Theta\|^2)}{k^2\|T\|^2N^{-1}}.$$

To compute the expected value we use Fubini's Theorem, (3.3) and Lemma 1 with $a=b=2\alpha t$. This yields

$$E(\|\Theta-E\Theta\|^2)=\int_0^\infty \operatorname{var}\Theta\,dx=\frac{1}{N}\sqrt{2/\pi}(\sqrt{2}-1)\sqrt{2\alpha t}$$
$$=\frac{\|T\|^2}{\sqrt{2}N}.$$

Here we have used Lemma 2. Since the probability of an event is one minus the probability of the complementary event we have proved the first claim in the theorem. We will now study the velocity. Let δ be a constant such that $\|EU-u\|\leqq\delta\|u\|$. Then

$$\{\|U-u\|>(\varepsilon+\delta)\|u\|\}\subset\{\|U-EU\|>\varepsilon\|u\|\}.$$

Thus it follows from Chebyshev's inequality that

$$1-P\left(\frac{\|U-u\|}{\|u\|}\leqq\varepsilon+\delta\right)\leqq P\left(\frac{\|U-EU\|}{\|u\|}>\varepsilon\right)\leqq\frac{E(\|U-EU\|^2)}{\varepsilon^2\|u\|^2}. \tag{5.3}$$

We will choose the value of ε such that the right-hand side is less than $1/k^2$. To estimate the expected value we use Fubini's theorem and (3.9) with the extra factor $1/N$. By combining (3.2) with the distributions in (3.4) and (3.5) and using Lemma 1

we get

$$E(\|U-EU\|^2) \leqq \frac{1}{N}\Delta t^2 \sum_{m=1}^{n-1} \sqrt{2/\pi}(\sqrt{2}-1)\sqrt{2\nu(n-m)\,\Delta t}$$

$$+\frac{1}{N}\Delta t^2 n \sum_{m=1}^{n} \sqrt{2/\pi}(\sqrt{2}-1)\sqrt{2\alpha m\,\Delta t + 2\nu(n-m)\,\Delta t}$$

$$\leqq \frac{5t^2}{4N}\frac{2}{\sqrt{\pi}}(\sqrt{2}-1)\sqrt{\max(\alpha,\nu)t}.$$

By inserting this result in (5.3) and using our lower bound (5.2) for $\|u\|^2$ it follows that the last term in (5.3) is less than k^{-2} if

$$\varepsilon = (1+\beta)\left(1+\frac{1+\sqrt{\beta}}{\sqrt{1+\beta}}\right)\frac{k}{\sqrt{N}}.$$

To find a convenient choice of δ we use the inequality (3.8). From definition (3.7) and Lemma 1 with $a = b = 2\max(\alpha, \nu)t$ we conclude that

$$\|EU-u\|^2 \leqq \Delta t^2 \int_0^\infty \left(\int_{x/\sqrt{a}}^\infty \frac{2\,e^{-y^2/2}}{\sqrt{2\pi}}\,dy\right)^2 dx$$

$$\leqq \Delta t^2 \sqrt{2/\pi}(2-\sqrt{2})\sqrt{2\max(\alpha,\nu)t}.$$

Combining this result with the lower bound for $\|u\|^2$ we obtain

$$\frac{\|EU-u\|^2}{\|u\|^2} \leqq (1+\beta)^2\left(1+\frac{1+\sqrt{\beta}}{\sqrt{1+\beta}}\right)^2\frac{\Delta t^2}{t^2} = \delta^2.$$

This completes the proof.

A typical numerical analysis statement is as follows: If the time step is sufficiently small and if we use a sufficiently large number of particles then the error in the computed solution is small. In this paper we have proved something similar, namely that if the time step is small and if we use a large number of particles then the error in the computed solution is small—with high probability. This does not mean that the error is small in any particular experiment—only that it is unlikely to be large.

There is another concept of convergence, namely convergence almost everywhere (or almost surely). It says, that if we take an infinite sequence of experiments in which the time steps tend to zero and the number of particles increase sufficiently quickly, then for any $\varepsilon > 0$, the error in the computed solutions will be less than ε in all, but a finite number of experiments—with probability one. The trouble with this formulation is that we must order the experiments in advance and that we get no information about the rate of convergence for a finite number of particles. If we set $\Delta t = t/n$ and $N = n^4$ and make one experiment for each n then the remarks following Theorems 1 and 2 together with the Borel-Cantelli lemma imply that the computed solutions converge almost surely, both pointwise and in the L^2 sense. For the proof use $k = n$. On the other hand, by repeating the experiments an increasing number of times, we can construct a probability space such that the method diverges almost surely. (If anything can go wrong, it will eventually.) The proper concept for numerical work is therefore convergence in probability and not convergence almost everywhere.

Acknowledgments. The problem was suggested by Ahmed Ghoniem. In addition the author thanks Alexandre J. Chorin, Phillip Colella and James Sethian for helpful discussions.

REFERENCES

[1] G. BENEFATTO AND M. PULVIRENTI, *A diffusion process associated to the Prandtl equation*, J. Funct. Anal., 52 (1983), pp. 330-343.

[2] P. BILLINGSLEY, *Convergence of Probability Measures*, John Wiley, New York, 1968.

[3] Y. BRENIER, *A particle method for one-dimensional non-linear reaction advection diffusion equations*, (1983), submitted to Math. Comp.

[4] A. J. CHORIN, *Numerical study of slightly viscous flow*, J. Fluid Mech., 57 (1973), pp. 785-796.

[5] ———, *Vortex sheet approximation of boundary layers*, J. Comp. Phys., 27 (1978), pp. 428-442.

[6] ———, *Numerical methods for use in combustion modeling*, Proc. International Conference on Numerical Methods in Science and Engineering, Versailles, 1979.

[7] W. FELLER, *An Introduction to Probability Theory and its Applications*, 2nd ed., vol. I, John Wiley, New York, 1957.

[8] ———, *An Introduction to Probability Theory and its Applications*, vol. II, John Wiley, New York, 1966.

[9] A. F. GHONIEM, A. J. CHORIN AND A. K. OPPENHEIM, *Numerical modeling of turbulent flow in a combustion tunnel*, Phil. Trans. Roy. Soc. London, A304 (1982), pp. 303-325.

[10] A. F. GHONIEM AND A. K. OPPENHEIM, *Random element method for numerical modeling of diffusional processes*, Proc. Eighth International Conference on Numerical Methods in Fluid Dynamics, E. Krause, ed., Springer, New York, 1982.

[11] A. GHONIEM AND F. S. SHERMAN, *Grid-free simulation of diffusion using random walk methods*, J. Comp. Phys., 61 (1985), pp. 1-37.

[12] O. H. HALD, *Convergence of random methods for a reaction-diffusion equation*, this Journal, 2 (1981), pp. 85-94.

[13] G. HELLWIG, *Partial Differential Equations*, Blaisdell, New York, 1964.

[14] C. R. ILLINGWORTH, *Unsteady laminar flow of a gas near an infinite flat plate*, Proc. Cambridge Phil. Soc., 46 (1950), pp. 603-613.

[15] L. D. LANDAU AND E. M. LIFSHITZ, *Fluid Mechanics*, Pergamon Press, London, 1959.

[16] C. MARCHIORO AND M. PULVIRENTI, *Hydrodynamics in two dimensions and vortex theory*, Comm. Math. Phys., 84 (1982), pp. 483-503.

[17] F. MILINAZZO AND P. G. SAFFMAN, *The calculation of large Reynolds number two-dimensional flow using discrete vortices with random walk*, J. Comp. Phys., 23 (1977), pp. 380-392.

[18] S. ROBERTS, *Accuracy of the random vortex method for a problem with non-smooth initial conditions*, J. Comp. Phys., 58 (1985), pp. 29-43.

[19] J. SETHIAN, *Validation of vortex methods*, to appear.

SIAM J. SCI. STAT. COMPUT.
Vol. 7, No. 4, October 1986

020

CANCELLATION ERRORS IN QUASI-NEWTON METHODS*

R. FLETCHER†

Abstract. It is shown that the effect of cancellation errors in a quasi-Newton method can be predicted with reasonable accuracy on the basis of simple formulae derived by using probabilistic arguments. Errors induced by cancellation are shown to have the potential to increase without bound as the method converges. They are shown to be one of the dominant factors affecting attainable accuracy in the variables of a problem.

Key words. quasi-Newton method, cancellation error, expectation, BFGS method

1. Introduction. The effect of cancellation errors on the performance of quasi-Newton methods has not been studied in any great detail. This paper shows that these effects can be predicted quite accurately (to within a factor of 10 or so) from simple formulae derived by using probabilistic arguments and a Euclidean norm. For variable metric methods in minimization, and in particular the BFGS method, use of the variational weighted Euclidean norm is most suitable. The resulting formulae enable the size of cancellation errors to be assessed in relation to both the size of the low rank corrections and the size of the Hessian or Jacobian matrix itself. In all cases it is seen that errors induced by cancellation have the potential to increase without bounds as the iteration converges. Practical examples are described in which cancellation errors cause all significant figures in the Hessian to be lost. These observations are clearly important when the Hessian is to be used for other purposes, for example, significance analysis. Cancellation errors in the Hessian or Jacobian are also shown to be one of the dominant factors affecting the attainable accuracy in the variables of a problem.

In § 2 the BFGS method is described and it is shown that the size of the low rank correction can be measured in an invariant norm. It is argued that cancellation errors form the dominant source of errors in the Hessian approximation. An appropriate probabilistic model is used to analyse cancellation errors, leading to a realistic estimate that is easily calculated. Some numerical experiments are described in which cancellation errors are measured, and a summary of their effect is given. The termination criterion in use, and also whether or not the minimum function value is zero, are both observed to be critical factors. In § 3 the propagation of cancellation errors in the BFGS method is considered when the gradient is estimated by finite differences. An equivalent probabilistic formula is described which indicates that in this case the effect of cancellation errors is more serious.

In § 4 it is shown that a similar analysis can be carried out in the case of a quasi-Newton method for solving nonlinear equations, although without the use of a weighted norm. The influence of cancellation errors on the attainable accuracy in the variables is considered. A general model for error propagation is described which can be used to analyse both iterative refinement for linear systems, Newton's method for nonlinear systems and quasi-Newton methods for nonlinear systems. It is shown that rounding errors in the residual vector are dominant in limiting the attainable accuracy in the variables of the problem, either directly or through the effect of cancellation in the Jacobian matrix. A probabilistic estimate of attainable accuracy is given on the basis of the rounding errors that arise on a single iteration.

* Received by the editors September 25, 1984 and in revised form May 15, 1985.
† Department of Mathematical Sciences, The University of Dundee, Dundee, Scotland.

2. The BFGS method. The BFGS method (for example, Fletcher (1980)) aims to find a local solution x^* of the problem

$$\text{minimize } f(x), \qquad x \in \mathbb{R}^n \tag{2.1}$$

when the first derivative vector $g(x)(=\nabla f(x))$ is explicitly available. The method is a variable metric line search method: an iterative sequence $x^{(k)}$, $k=1, 2, \cdots$ is determined, and on iteration k a search direction $s^{(k)}$ is calculated which satisfies the equation

$$B^{(k)}s^{(k)} = -g^{(k)} \tag{2.2}$$

($g^{(k)}$ denotes $g(x^{(k)})$). $B^{(k)}$ is a symmetric positive definite matrix which approximates the Hessian matrix $G(x^{(k)})(=\nabla^2 f(x^{(k)}))$. $B^{(1)}$ is user supplied and for $k>1$, $B^{(k)}$ is obtained from $B^{(k-1)}$ by means of an updating formula. The vector $s^{(k)}$ is used in a line search; that is, a step $\alpha^{(k)}$ is calculated which approximately solves the subproblem

$$\text{minimize } f(x^{(k)}+\alpha s^{(k)}), \qquad \alpha \in \mathbb{R} \tag{2.3}$$

in a certain sense. Then the next iterate is defined by

$$x^{(k+1)} = x^{(k)} + \alpha^{(k)}s^{(k)}. \tag{2.4}$$

Finally the difference vectors

$$\delta^{(k)} = x^{(k+1)} - x^{(k)} \tag{2.5}$$

and

$$\gamma^{(k)} = g^{(k+1)} - g^{(k)} \tag{2.6}$$

are determined, and used in an updating formula to calculate $B^{(k+1)}$.

The BFGS method uses the updating formula

$$B^+ = B - \frac{B\delta\delta^T B}{\delta^T B\delta} + \frac{\gamma\gamma^T}{\delta^T\gamma} \tag{2.7}$$

where superscript $^+$ is henceforth used to denote the successor iterate. This formula has the property (for example, Dennis and Schnabel (1983)) that it solves the variational problem

$$\underset{H^+}{\text{minimize}} \; \|H^+ - H\|_W \tag{2.8}$$

$$\text{subject to } H^{+T} = H^+, \tag{2.9}$$

$$H^+\gamma = \delta \tag{2.10}$$

where H refers throughout to B^{-1}, and where the norm is defined by

$$\|E\|_W = (\text{tr } EWE^TW)^{1/2}. \tag{2.11}$$

W is any positive definite symmetric matrix for which $W\delta = \gamma$. In practice the choice $W = B^+$ can be made and Al-Baali and Fletcher (1983) show that the minimum value of the norm is given by

$$\|H^+ - H\|_{B^+} = (a^2 - 2b + 1)^{1/2} \triangleq \Delta, \tag{2.12}$$

say, where

$$a = \frac{\gamma^T H\gamma}{\delta^T\gamma} \tag{2.13}$$

and

$$b = \frac{\delta^T \gamma}{\delta^T B \delta}. \tag{2.14}$$

Equation (2.12) can be interpreted as giving a scalar measure of the size of the correction to B in the BFGS update. This measure is independent of linear transformations in the variables x which is useful invariance property. A feature of (2.8) is that it is the norm of the change to H (that is B^{-1}) which is minimized. This suggests that in measuring other errors in B, we should first find the corresponding error in H, and then measure this error using $\|\cdot\|_{B^+}$. This measure is then directly comparable with the quantity Δ. This approach is used below to assess the effect of cancellation errors incurred in forming B^+. Furthermore the size of the matrix $H^{(k)}$ can also be measured in this norm because

$$\|H^{(k)}\|_{B^{(k)}} = \|H^*\|_{B^*} = \sqrt{n}. \tag{2.15}$$

Thus the relative effect of errors can also be readily assessed.

There are various ways in which rounding errors can affect the matrix B. For example if B is represented by factors $B = LDL^T$ then there will be rounding errors associated with the substitution processes in solving (2.2), and also in updating the factors to account for the two successive rank one changes defined by (2.7). These errors can be bounded in a satisfactory manner (see Wilkinson (1965) and Fletcher and Powell (1974), respectively). The dominant rounding errors are likely to be cancellation errors arising in forming the vector γ in (2.6), and it is solely these errors that are considered in this paper. To analyse them, it is assumed that the error in γ is e and $\bar{\gamma} = \gamma + e$. The consequent error in B^+ is

$$E_{B^+} = \frac{\bar{\gamma}\bar{\gamma}^T}{\delta^T \bar{\gamma}} - \frac{\gamma\gamma^T}{\delta^T \gamma}. \tag{2.16}$$

To first order

$$(\delta^T \bar{\gamma})^{-1} = (\delta^T \gamma + \delta^T e)^{-1} = \delta^T \gamma^{-1} - \delta^T e / \delta^T \gamma^2, \tag{2.17}$$

and hence, also to first order,

$$E_{B^+} = \frac{e\gamma^T + \gamma e^T}{\delta^T \gamma} - \frac{\delta^T e}{\delta^T \gamma} \frac{\gamma\gamma^T}{\delta^T \gamma}. \tag{2.18}$$

Now denote the corresponding error in H^+ *by* E_{H^+}. It follows from $B^+ H^+ = I$ that to first order

$$E_{H^+} = -H^+ E_{B^+} H^+. \tag{2.19}$$

Thus the required measure is given from (2.11) by

$$\|E_{H^+}\|^2_{B^+} = \text{tr}\, H^+ E_{B^+} H^+ E_{B^+}. \tag{2.20}$$

Substituting (2.18) and using $H^+ \gamma = \delta$ it follows that

$$\|E_{H^+}\|^2_{B^+} = \frac{2e^T H^+ e}{\delta^T \gamma} - \left(\frac{\delta^T e}{\delta^T \gamma}\right)^2 = \frac{2}{\delta^T \gamma} e^T \hat{H} e \tag{2.21}$$

where

$$\hat{H} = H^+ - \tfrac{1}{2}\delta\delta^T / \delta^T \gamma. \tag{2.22}$$

It is easy to show that $\hat{H}$ is positive definite from the form of the DFP formula and the fact that $H^+_{\text{BFGS}} \geqq H^+_{\text{DFP}}$.

The cancellation errors in γ arise due to errors in g and it is necessary to make some assumption about how the latter errors arise. Let h_i denote the error in g_i, arising from the use of floating point arithmetic of relative precision ε. If $g = Gx - b$ were linear, then $|h_i| \sim p_i\varepsilon$ would be a good estimate of the error, where p_i is the largest absolute partial sum in forming $\sum_j G_{ij}x_j - b_i$. For nonlinear $g(x)$ the stationary point condition $g_i(x) = 0$ is likely to come about in a similar way by the cancellation of a number of terms, and so the assumption is made that there exists a vector p such that

$$h_i = \varepsilon_i p_i \tag{2.23}$$

where ε_i is an independent random error from a uniform distribution in $[-\varepsilon, \varepsilon]$. Then the error in γ_i is

$$e_i = h_i^+ - h_i = (\varepsilon_i^+ - \varepsilon_i)p_i. \tag{2.24}$$

The mean of ε_i is $\mathscr{E}(\varepsilon_i) = 0$ ($\mathscr{E}$ denotes expectation) and so $\mathscr{E}(e_i) = 0$ and the variance is

$$\begin{aligned} \mathscr{E}(e_i^2) &= p_i^2\mathscr{E}(\varepsilon_i^+ - \varepsilon_i)^2 \\ &= p_i^2(\mathscr{E}(\varepsilon_i^{+2}) + \mathscr{E}(\varepsilon_i^2)) \\ &= \tfrac{2}{3}p_i^2\varepsilon^2 \end{aligned} \tag{2.25}$$

using independence and the result that the uniform distribution has variance $\frac{1}{3}\varepsilon^2$. Furthermore

$$\mathscr{E}(e_ie_j) = p_ip_j\mathscr{E}((\varepsilon_i^+ - \varepsilon_i)(\varepsilon_j^+ - \varepsilon_j)) = 0, \qquad i \neq j \tag{2.26}$$

by independence. Therefore from (2.21)

$$\mathscr{E}(\|E_{H^+}\|^2_{B^+}) = \frac{2}{\delta^T\gamma}\sum_{ij}\hat{H}_{ij}\mathscr{E}(e_ie_j) = \frac{4\varepsilon^2}{3\delta^T\gamma}\sum_i p_i^2\hat{H}_{ii} \tag{2.27}$$

and hence

$$(\mathscr{E}(\|E_{H^+}\|^2_{B^+}))^{1/2} = \varepsilon\left(\frac{4}{3\delta^T\gamma}\sum_i p_i^2\hat{H}_{ii}\right)^{1/2}. \tag{2.28}$$

It can be shown (Fletcher (1983)) that $(\mathscr{E}(z^2))^{1/2}$ is an upper bound for $\mathscr{E}(|z|)$ and that a realistic estimate is

$$\mathscr{E}(|z|) \doteq 0.8(\mathscr{E}(z^2))^{1/2}. \tag{2.29}$$

Thus (2.28) gives a close upper bound for the likely effect of cancellation errors in γ for the update.

Assuming that p is known, (2.28) is convenient for computation (if necessary the diagonal elements H_{ii} can be recurred separately), and therefore can be used in an algorithm to assess cancellation error effects. Approximately it has the form $\mathscr{E}(\|E_{H^+}\|_{B^+}) \sim \varepsilon/\|\delta\|$ showing that the errors induced by cancellation *increase without bound* as the iteration converges. The expression (2.12) for Δ is invariant under a nonsingular linear transformation of the variables x; however the way in which rounding errors arise depends on the coordinate system, so that (2.28) is not invariant under a linear transformation. Nonetheless it is invariant under diagonal scaling of the x variables, assuming that the vector p transforms correspondingly, which is the most that can be expected.

To measure the effect of rounding errors in practice, numerical experiments on some standard test problems (see Fletcher (1980)) are reported. A DEC10 computer with $\varepsilon = 2^{-27} \doteq 7.45_{10^{-9}}$ is used and p_i is estimated by the quantity

$$p_i^{(k)} = \max_{j \leqq k} |g_i^{(j)}|. \tag{2.30}$$

This is likely to be reasonable when $x^{(1)}$ is not close to x^* but the magnitudes of the elements $x_i^{(1)}$ and x_i^* for any i are similar. The results of the experiment are shown in the main part of Table 1; each row corresponds to a problem and each column to one of the last five iterations (with iteration count increasing from left to right). Each entry contains two numbers; the upper one is Δ in (2.12) and the lower one is $(\mathscr{E}(\|E_{H^+}\|_{B^+}^2))^{1/2}$ in (2.28), both for that iteration. The BFGS method is run until no further improvement in $f(x)$ can be obtained. Because all the test problems are least squares problems, with $f^* = 0$ except for Chebyquad $n = 8$, high precision in f can be obtained. This allows the iteration to continue to higher accuracy than would be the case when $f^* \neq 0$ (because each iteration requires f to decrease). The termination criterion that I would normally use for these problems is $f - f^+ \leqq 10^{-8}$. This criterion gives an indication of the maximum accuracy that can be obtained on a problem for which $f^* \neq 0$ and is indicated by an asterisk in the table. In the two rightmost columns of the table, the measure $\|H - H^*\|_{G^*}$ of the difference between the final H matrix and the true H^* matrix is given (for both termination criteria). Finally all these measures can be compared with that of H^* itself using the fact that $\|H^*\|_{G^*} = \sqrt{n}$.

The behaviour of these measures is very much as the theory predicts. The estimate $(\mathscr{E}(\|E_{H^+}\|_{B^+}^2))^{1/2}$ of cancellation errors in B is negligible on early iterations ($\sim 10^{-7}$ or so). Near the solution a rapid increase in magnitude is observed (see Table 1) in accordance with the $\sim \varepsilon / \|\delta\|$ behaviour, reaching a final value of ~ 1 in some cases. (Since equation (2.28) is based on first order approximations, not too much attention should be paid to values of ~ 1, but they do indicate that cancellation errors are substantial.) The quantity Δ which measures the size of the rank 2 correction is initially large ($B^{(1)} = I$ is taken) but drops sharply after about n iterations, presumably indicating that a good estimate of the Hessian matrix has been built up. Changes then stay at about the same level with random fluctuations, but start to decay in the neighbourhood of x^*, reflecting the result that $\Delta \to 0$ as $B \to G^*$ in exact arithmetic. However in many cases Δ is observed to rise again before termination. This only occurs when Δ and $(\mathscr{E}(\|E_{H^+}\|_{B^+}^2))^{1/2}$ are comparable and would appear to indicate that changes from cancellation errors are becoming dominant in B.

Normally when $f^* \neq 0$ the iteration would be expected to obtain an accuracy $f^{(k)} - f^* \sim \varepsilon$, with $\|x^{(k)} - x^*\| \sim \varepsilon^{1/2}$ and $\|g^{(k)} - g^*\| \sim \varepsilon^{1/2}$. This observation follows because the iteration terminates when f can no longer be improved, and by virtue of Taylor series expansions. Table 1 indicates that at this stage cancellation errors are not dominant—see the * termination point and the Chebyquad $n = 8$ case. Thus for general use it can be concluded that cancellation errors are not significant in affecting the asymptotic behaviour of the BFGS method and the resulting Hessian approximations are not significantly contaminated by cancellation. Another feature to observe is that the size of the corrections Δ is an underestimate of the true error in H, but not grossly so. Also the true error is not much less than $\sqrt{n}$, which is the measure of H itself. Thus at normal termination H may only be accurate to one or two significant figures.

When $f^* = 0$ a different picture emerges, because of the higher accuracy that can be attained in f. The trigonometric $n = 2$ example is atypical in that finite termination

TABLE 1
Assessment of cancellation errors for the BFGS *method.*

Problem	Last five iterations					$\|H-H^*\|_{B^*}$ Normal termination (*)	$\|H-H^*\|_{B^*}$ Extended termination
Rosenbrock	†$3.103_{10^{-1}}$	$3.464_{10^{-1}}$	$1.016_{10^{-1}}$	*$8.122_{10^{-2}}$	$9.618_{10^{-3}}$	$5.368_{10^{-1}}$	$8.872_{10^{-2}}$
	$8.682_{10^{-6}}$	$4.887_{10^{-5}}$	$1.050_{10^{-4}}$	$1.826_{10^{-3}}$	$2.240_{10^{-2}}$		
Chebyquad							
$n=2$	$1.209_{10^{0}}$	*$6.903_{10^{-3}}$	$7.505_{10^{-1}}$	$2.427_{10^{0}}$	$1.139_{10^{0}}$	$3.000_{10^{0}}$	$3.267_{10^{-1}}$
	$1.268_{10^{-6}}$	$4.053_{10^{-4}}$	$5.010_{10^{-2}}$	$1.050_{10^{-1}}$	$1.840_{10^{-1}}$		
$n=4$	$1.250_{10^{-2}}$	*$7.428_{10^{-1}}$	$2.565_{10^{-1}}$	$1.367_{10^{-1}}$	$2.345_{10^{-1}}$	$8.486_{10^{-1}}$	$3.729_{10^{-1}}$
	$1.237_{10^{-5}}$	$9.098_{10^{-4}}$	$1.967_{10^{-3}}$	$1.211_{10^{-2}}$	$1.309_{10^{-1}}$		
$n=6$	*$3.601_{10^{-1}}$	$1.506_{10^{-1}}$	$1.628_{10^{-1}}$	$5.444_{10^{-2}}$	$6.456_{10^{-1}}$	$7.771_{10^{-1}}$	$1.124_{10^{0}}$
	$7.479_{10^{-5}}$	$1.977_{10^{-4}}$	$1.948_{10^{-3}}$	$1.328_{10^{-2}}$	$2.863_{10^{-1}}$		
$n=8$	$6.848_{10^{-1}}$	$3.735_{10^{-1}}$	$3.636_{10^{-1}}$	$1.742_{10^{-2}}$	*$8.723_{10^{-2}}$	$1.135_{10^{0}}$	$1.193_{10^{0}}$
	$3.861_{10^{-6}}$	$2.011_{10^{-6}}$	$5.362_{10^{-6}}$	$1.449_{10^{-5}}$	$9.065_{10^{-5}}$		
Trigonometric							
$n=2$	$1.833_{10^{4}}$	$1.373_{10^{4}}$	$9.883_{10^{-2}}$	$1.709_{10^{-2}}$	$2.086_{10^{-3}}$*	$1.858_{10^{-3}}$	$1.858_{10^{-3}}$
	$4.754_{10^{-9}}$	$5.937_{10^{-9}}$	$6.793_{10^{-9}}$	$7.717_{10^{-8}}$	$6.221_{10^{-6}}$		
$n=4$	$2.362_{10^{-1}}$	$5.949_{10^{-2}}$	$4.305_{10^{-2}}$	*$3.900_{10^{-1}}$	$1.143_{10^{0}}$	$1.245_{10^{-1}}$	$8.980_{10^{0}}$
	$8.009_{10^{-5}}$	$3.140_{10^{-4}}$	$5.644_{10^{-3}}$	$1.548_{10^{-1}}$	$1.599_{10^{0}}$		
$n=6$	$2.590_{10^{-1}}$	$1.425_{10^{-1}}$	$1.275_{10^{-1}}$	$8.384_{10^{-2}}$	*$5.576_{10^{-1}}$	$6.489_{10^{-1}}$	$3.145_{10^{0}}$
	$3.988_{10^{-5}}$	$1.883_{10^{-4}}$	$1.479_{10^{-3}}$	$1.756_{10^{-2}}$	$2.938_{10^{-1}}$		
$n=8$	*$1.038_{10^{-1}}$	$1.476_{10^{0}}$	$5.331_{10^{0}}$	$1.135_{10^{1}}$	$5.660_{10^{1}}$	$1.114_{10^{0}}$	$7.996_{10^{0}}$
	$2.684_{10^{-2}}$	$3.413_{10^{-1}}$	$2.085_{10^{0}}$	$2.209_{10^{1}}$	$1.466_{10^{0}}$		
$n=10$	$1.509_{10^{-1}}$	*$3.409_{10^{-1}}$	$1.068_{10^{0}}$	$3.387_{10^{0}}$	$2.386_{10^{1}}$	$1.007_{10^{0}}$	$3.308_{10^{1}}$
	$1.057_{10^{-2}}$	$7.567_{10^{-2}}$	$2.458_{10^{-1}}$	$8.567_{10^{-1}}$	$7.099_{10^{0}}$		

* Previous iteration satisfied $f-f^+ \leqq 10^{-8}$ (normal termination).

† Upper figure is Δ and lower figure is $(\mathscr{E}(\|E_{H^+}\|^2_{B^+}))^{1/2}$ from (2.28).

occurs with $f = g_1 = g_2 = 0$ and the limiting effect is not seen. In all other cases when $f^* = 0$ in Table 1 and extended termination is allowed, cancellation errors are seen to play a major part. These errors rise to the same level as the correction measure Δ, indicating significant contamination due to cancellation. In some cases both measures continue to increase, and the true error $\|H - H^*\|_{B^*}$ is seen to deteriorate to an extent that the error in H is greater than the size of H itself. Thus cancellation errors can lead to a complete loss of significance in these circumstances and it would be prudent to cease to continue updating H before this situation is attained. These observations can also be expected to apply to quasi-Newton methods for solving equations (see § 4).

3. The no-derivative BFGS method. In this section the effect of cancellation errors is considered when the gradient vector $g^{(k)}$ in (2.2) is estimated by finite difference techniques. The VA10A program of Fletcher (1972) (modified to implement only the BFGS update) is followed in which the user is asked to supply a vector t ($t > 0$) such that t_i is the typical magnitude of x_i, and a scalar differencing increment h. Then the forward difference estimate of $g_i^{(k)}$ is

$$\hat{g}_i^{(k)} = (f(x^{(k)} + ht_i e_i) - f^{(k)})/(ht_i) \tag{3.1}$$

(e_i is a coordinate vector), and the central difference estimate is

$$\hat{g}_i^{(k)} = (f(x^{(k)} + ht_i e_i) - f(x^{(k)} - ht_i e_i))/(2ht_i). \tag{3.2}$$

In VA10A it is assumed that t and h are fixed for all k. Also the algorithm makes as much progress as possible with forward differences before switching to central differences for the remaining iterations. Only the central difference phase is analysed here but similar remarks hold when forward differences are used.

To estimate the effect of errors, it is assumed for differences taken about $x^{(k)}$ that the error in $f(x)$ is

$$\bar{f}(x) - f(x) = p^{(k)}\varepsilon_j \tag{3.3}$$

($\bar{f}$ indicates the computed value) where ε_j is independently and uniformly distributed in $[-\varepsilon, \varepsilon]$ and $p^{(k)}$ is some known constant. Using (3.3) and a Taylor series, the computed estimate of $\hat{g}_i^{(k)}$ in (3.2) is given by

$$\bar{g}_i^{(k)} = \frac{p^{(k)}(\varepsilon_{1i} - \varepsilon_{2i})}{2ht_i} + g_i^{(k)} + \tfrac{1}{6}\mathscr{G}_{iii}^{(k)}(ht_i)^2 + O(h^4) \tag{3.4}$$

where $\mathscr{G}_{iii} = \partial^3 f/\partial x_i^3$, and ε_{1i} and ε_{2i} denote two errors typified by (3.3). Thus if $\bar{\gamma}^{(k)} = \bar{g}^{(k+1)} - \bar{g}^{(k)}$ is computed then the error $e^{(k)}$ in $\bar{\gamma}^{(k)}$ is given by

$$e_i^{(k)} = \frac{p^{(k+1)}(\varepsilon_{3i} - \varepsilon_{4i}) - p^{(k)}(\varepsilon_{1i} - \varepsilon_{2i})}{2ht_i} + O(\delta^{(k)}h^2) + O(h^4). \tag{3.5}$$

Observe that a consequence of using a fixed value of h is that the third derivative terms cancel and the leading truncation errors come from fourth derivative terms. In view of this these terms are ignored and it is assumed that $e^{(k)}$ is dominated by rounding errors. In a similar manner to § 2, $\mathscr{E}(e_i) = 0$, $\mathscr{E}(e_i e_j) = 0$ and

$$\mathscr{E}(e_i^2) = \varepsilon^2(p^{(k)^2} + p^{(k+1)^2})/(6h^2 t_i^2). \tag{3.6}$$

Therefore it follows from (2.21) that

$$(\mathscr{E}(\|E_{H^+}\|_{B^+}^2))^{1/2} = \frac{\varepsilon}{h}\left(\frac{p^{(k)^2} + p^{(k+1)^2}}{3\delta^T\gamma}\sum_i \hat{H}_{ii}/t_i^2\right)^{1/2}. \tag{3.7}$$

As for (2.28) this measures the effect that cancellation errors have on the B matrix. Like (2.28) it is readily calculated, and it also has the same invariance properties. Asymptotically it suggests that $\mathscr{E}(\|E_{H^+}\|_{B^+}) \sim \varepsilon/(h\|\delta\|)$ showing not only that cancellation errors increase without bound but also that they are magnified by a factor h^{-1} from the differencing interval. Therefore cancellation effects are likely to be more severe when a no-derivative method is used.

The numerical experiments described in § 2 are repeated for the no-derivative BFGS method. Whether or not $f^* = 0$ is again an important consideration. If $f^* = 0$ then the estimation of $p^{(k)}$ in (3.3) is likely to need careful consideration, perhaps involving some elementary error analysis. However most practical applications of the no-derivative BFGS method are expected to arise when $f^* \neq 0$. In this case there will certainly be one rounding error which is realistically described by (3.3) with $p^{(k)} = |f^{(k)}|$. If we assume that this is dominant, and that we are interested in using (3.7) close to the solution so that $p^{(k)} \simeq p^{(k+1)}$, then the estimate

$$(\mathscr{E}(\|E_{H^+}\|_{B^+}^2))^{1/2} = \frac{\varepsilon|f^{(k)}|}{h}\left(\frac{2}{3\delta^T\gamma}\sum_i \hat{H}_{ii}/t_i^2\right)^{1/2} \tag{3.8}$$

suggests itself. Now the test problems referred to in § 2 are almost all least squares problems with $f^* = 0$; to make $f^* \neq 0$ so as to provide a realistic test of (3.8), the fixed nonzero quantity $\frac{1}{3}$ is added to each of the objective functions. The differencing interval $h = 10^{-3}$ is used in all cases. The results of these experiments are shown in Table 2; as before the iteration is continued until no further improvement in f can be obtained. The quantity Δ which measures the size of the rank 2 correction in (2.7) behaves very similarly to the case when analytical first derivatives are available. The estimate

TABLE 2
Cancellation errors for the no-derivative BFGS *method.*

Problem	Last five iterations				
Rosenbrock	†$6.677_{10^{-1}}$	$3.641_{10^{-1}}$	$1.422_{10^{-1}}$	$5.337_{10^{-2}}$*	$1.142_{10^{-2}}$
	$5.332_{10^{-5}}$	$1.479_{10^{-4}}$	$4.210_{10^{-4}}$	$4.713_{10^{-3}}$	$2.091_{10^{-2}}$
Chebyquad					
$n=2$	$1.637_{10^{0}}$	$2.321_{10^{-1}}$	$6.174_{10^{-3}}$	$5.830_{10^{-3}}$	$8.165_{10^{-3}}$*
	$6.950_{10^{-5}}$	$1.381_{10^{-4}}$	$3.946_{10^{-4}}$	$2.250_{10^{-4}}$	$4.882_{10^{-3}}$
$n=4$	$2.334_{10^{0}}$	$3.491_{10^{-1}}$	$3.433_{10^{-1}}$	$1.688_{10^{-1}}$*	$9.949_{10^{-2}}$
	$1.712_{10^{-4}}$	$2.049_{10^{-4}}$	$7.286_{10^{-4}}$	$2.117_{10^{-3}}$	$1.418_{10^{-2}}$
$n=6$	$7.378_{10^{-1}}$	$9.108_{10^{-1}}$	$5.088_{10^{-1}}$	$2.889_{10^{-1}}$*	$1.877_{10^{-1}}$
	$1.290_{10^{-3}}$	$2.367_{10^{-3}}$	$2.229_{10^{-3}}$	$3.799_{10^{-3}}$	$1.501_{10^{-2}}$
$n=8$	$2.802_{10^{-1}}$	$5.847_{10^{-1}}$	$2.383_{10^{-1}}$*	$3.484_{10^{-1}}$	$1.595_{10^{-1}}$
	$2.386_{10^{-4}}$	$6.386_{10^{-4}}$	$1.927_{10^{-3}}$	$1.260_{10^{-2}}$	$2.781_{10^{-2}}$
Trigonometric					
$n=2$	$1.369_{10^{4}}$	$9.734_{10^{-2}}$	$2.039_{10^{-2}}$	$5.371_{10^{-3}}$	$9.297_{10^{-4}}$
	$3.028_{10^{-8}}$	$4.088_{10^{-9}}$	$3.758_{10^{-8}}$	$6.174_{10^{-8}}$	$1.552_{10^{-7}}$
$n=4$	$2.523_{10^{-1}}$	$9.741_{10^{-1}}$	$5.842_{10^{-1}}$	$2.603_{10^{-2}}$	$2.395_{10^{1}}$*
	$6.700_{10^{-7}}$	$2.097_{10^{-6}}$	$2.543_{10^{-6}}$	$1.281_{10^{-5}}$	$4.653_{10^{0}}$
$n=6$	$7.154_{10^{-2}}$*	$5.475_{10^{-1}}$	$6.673_{10^{0}}$	$8.195_{10^{-1}}$	$5.395_{10^{-1}}$
	$6.439_{10^{-5}}$	$2.413_{10^{-3}}$	$6.370_{10^{-3}}$	$7.862_{10^{-5}}$	$6.573_{10^{-3}}$
$n=8$	$5.585_{10^{-1}}$	$4.975_{10^{-1}}$	$2.893_{10^{-1}}$	$2.608_{10^{-1}}$	$2.189_{10^{-1}}$*
	$1.842_{10^{-6}}$	$2.914_{10^{-6}}$	$5.548_{10^{-6}}$	$2.192_{10^{-5}}$	$1.011_{10^{-4}}$
$n=10$	$3.056_{10^{-1}}$	$3.453_{10^{-1}}$	$9.557_{10^{-2}}$	$2.992_{10^{-1}}$*	$1.204_{10^{-1}}$
	$6.488_{10^{-7}}$	$3.676_{10^{-6}}$	$8.169_{10^{-6}}$	$6.592_{10^{-5}}$	$1.866_{10^{-4}}$

* Previous iteration satisfied $f - f^+ \leqq 1_{10^{-6}}$.

† Upper figure is Δ and lower figure is $(\mathscr{E}(\|E_{H^+}\|_{B^+}^2))^{1/2}$ from (3.8).

$(\mathscr{E}(\|E_{H^+}\|_{B^+}^2))^{1/2}$ is again seen to be negligible initially and to increase close to the solution. However it shows a higher level of cancellation errors than in § 2. For the Rosenbrock and Chebyquad problems cancellation errors are seen to reach a magnitude comparable to the changes Δ in B. This contracts with the situation in § 2 that for this level of precision in f^*, cancellation effects are negligible. The results for the Trigonometric problems show the result less clearly. Nonetheless there is evidence that here also cancellation errors are significant; the $n=4$ and $n=6$ problems are only solved to a relative precision of 10^{-6} or less, and the values of Δ show significant increases close to the solution. A likely explanation is that the level of rounding errors in the calculation of f is much greater than $\sim \varepsilon f^{(k)}$ for the Trigonometric problems, and hence that a larger value of $p^{(k)}$ would have been appropriate. Overall there seems to be plenty of evidence for a no-derivative BFGS method that the maximum accuracy in $f^*(f^* \neq 0)$ can only be obtained at the cost of severely polluting B with cancellation errors.

There are various ideas for a no-derivative method that might be pursued as a consequence of these results. One is that when the cancellation error is seen to reach a certain level, then no further changes in B are made. Another is the well known idea (for example, Stewart (1967)) of attempting to choose h to balance truncation and cancellation effects. There are advantages in keeping h fixed (see (3.5) ff.) so this would only be done intermittently and possibly with limits to the variation allowed in h. However it may well be worth reconsidering this idea in the light of formulae of the type described here.

4. Quasi-Newton methods. This last section deals with the problem of solving a system of nonlinear equations

$$r(x)=0 \tag{4.1}$$

$(r:\mathbb{R}^n \to \mathbb{R}^n)$ when the (nonsymmetric) Jacobian matrix $A=\nabla r^T$ is not available but is estimated by a matrix J. J is updated after each iteration on the basis of differences

$$\gamma = r^+ - r \tag{4.2}$$

and

$$\delta = x^+ - x. \tag{4.3}$$

A rank one correction formula which satisfies $J^{+T}\delta = \gamma$ can be written

$$J^+ = J + \frac{w(\gamma - J^T\delta)^T}{w^T\delta} \tag{4.4}$$

for some vector w. Two well known such formulae are Barnes and Broyden's formulae, the latter corresponding to $w=\delta$ (see, for example, Fletcher (1980) for details). The main difference between here and § 2 is that here there is no invariant norm that can be used to measure the size of the errors. Accordingly much more importance is placed on the problem of equilibrating the matrix A before a common norm is used. The recent technique of Fletcher (1983) might be useful here. However if the Euclidean norm defined by

$$\|X\| = \sqrt{\operatorname{tr} X^T X} \tag{4.5}$$

is used, then analogous formulae to those in § 2 may be developed. The size of the rank one correction is seen to be

$$\Delta = \|J^T\delta - \gamma\|\,\|w\|/|w^T\delta| \tag{4.6}$$

or in the case of Broyden's formula

$$\Delta = \|J^T\delta - \gamma\|/\|\delta\|. \tag{4.7}$$

To assess cancellation errors in J, errors in r are assumed to obey the model

$$\bar{r}_i - r_i = \varepsilon_i p_i \tag{4.8}$$

where a bar denotes a computed quantity. The motivation for this is the same as for (2.23). Following (2.24) and (2.25) the error in γ_i is

$$e_i = p_i(\varepsilon_i^+ - \varepsilon_i), \tag{4.9}$$

with $\mathscr{E}(e_i) = 0$, $\mathscr{E}(e_i e_j) = 0$ and $\mathscr{E}(e_i^2) = \frac{2}{3}p_i^2\varepsilon^2$. This error in γ induces a change

$$E_{J^+} = we^T/w^T\delta \tag{4.10}$$

in J^+. Using the expected values above it follows that

$$\mathscr{E}(\|E_{J^+}\|^2) = w^T w \mathscr{E}(e^T e)/w^T\delta^2 \tag{4.11}$$

which gives

$$(\mathscr{E}(\|E_{J^+}\|^2))^{1/2} = \frac{\varepsilon\|w\|\,\|p\|\sqrt{\frac{2}{3}}}{|w^T\delta|} \tag{4.12}$$

or for Broyden's formula

$$(\mathscr{E}(\|E_{J^+}\|^2))^{1/2} = \frac{\varepsilon\|p\|\sqrt{\frac{2}{3}}}{\|\delta\|}. \tag{4.13}$$

The main conclusion as for the BFGS method is that cancellation errors increase without bound as the iteration converges. If x^* is obtained to full precision, $\|\delta\| \sim \varepsilon\|x^*\|$ and cancellation errors are ~ 1 and are therefore dominant in the estimate of J. As in § 3 it may be prudent to discontinue updating J when the quantity $(\mathscr{E}(\|E_{J^+}\|^2))^{1/2}$ in (4.13) reaches a certain level.

It is also possible to say something about the effect of the above cancellation errors on the attainable accuracy of quasi-Newton methods. A unit step iteration of a Newton-like method can be written along the lines of Moler (1967) as

$$\bar{r} = r + f, \tag{4.14}$$

$$(J+F)^T\bar{\delta} = -\bar{r}, \tag{4.15}$$

$$\bar{x}^+ = x + \bar{\delta} + g, \tag{4.16}$$

where f, F and g account for the different rounding errors that arise at each stage. If $h = x - x^*$ denotes the total error in x and $\bar{h}^+ = \bar{x}^+ - x^*$, then it follows using the identity $(J+F)^{-T} = J^{-T} - J^{-T}F^T(J+F)^{-T}$ that

$$\bar{h}^+ = h - J^{-T}r - J^{-T}f - J^{-T}F^T\bar{\delta} + g. \tag{4.17}$$

If x^+ represents the result of (4.14), (4.15) and (4.16) without error, and $h^+ = x^+ - x^*$ then

$$\bar{x}^+ - x^+ = \bar{h}^+ - h^+ = -J^{-T}f - J^{-T}F^T\bar{\delta} + g. \tag{4.18}$$

This expression gives the contribution of rounding errors to x^+ and is applicable in all the various situations of interest. For a linear system $A^Tx = b$, then to analyse iterative refinement we can set $J = A$. If f is the result of single precision rounding errors (i.e. it satisfies (4.8) for some p_i which is related to the maximum modulus

partial sum in forming r_i) then the term $-A^{-T}f$ has magnitude $\sim\|A^{-1}\|\,\|p\|\varepsilon$. We can assume $\|\bar{\delta}\|\sim\|h\|$ and so the term $-J^{-T}F^T\bar{\delta}$ is seen to be of lower order. Thus the term $-A^{-T}f$ limits the attainable accuracy in x^+. Only if higher precision is used in the calculation of r can iterative refinement be used to reduce the attainable accuracy to the level of $g(\|g\|\sim\varepsilon\|x^*\|)$.

Similar conclusions hold for a nonlinear system (with $A=A^{(k)}$, $J=J^{(k)}$ etc.). Taking $J=A$ gives Newton's method with unit step, and if we make the same assumptions (4.8) about how errors in r arise, then (4.18) is again valid. However the term $-A^{-T}f$ only gives a lower limit on attainable accuracy. A Taylor series for $r(x)$ about $x^{(k)}$ gives

$$0=r^*=r^{(k)}-A^{(k)T}h^{(k)}+O(\|h^{(k)}\|^2) \tag{4.19}$$

and if a Lipschitz condition $\|A(y)-A(x)\|\leqq\lambda\|y-x\|$ holds for all x, y in some neighbourhood of x^* then the bound $O(\|h^{(k)}\|^2)\leqq\lambda\|h^{(k)}\|^2$ is valid. It follows from (4.17) with $A=J$, and (4.19) that

$$\bar{h}^+=O(\|h\|^2)-A^{-T}f-A^{-T}F^T\bar{\delta}+g \tag{4.20}$$

where the bound $O(\|h\|^2)\leqq\lambda\|A^{-1}\|\,\|h\|^2$ is now valid. It may be that the $O(\|h\|^2)$ term is significant relative to $-A^{-T}f$ in which case it may not be possible to reach the lower limit of attainable accuracy referred to above. However if λ is such that

$$\lambda\|A^{-1}\|^2\|p\|\varepsilon<1 \tag{4.21}$$

then for any h such that $\|h\|=\|A^{-1}\|\,\|p\|\varepsilon$ it follows that $O(\|h\|^2)\leqq\lambda\|A^{-1}\|\,\|h\|^2<\|h\|$ and so Newton's method contracts h at this level of accuracy. Thus if the nonlinearity of the equations is sufficiently small in the sense of (4.21) then the level of attainable accuracy is the same as that deduced above from (4.18).

It is now possible to analyse a unit step quasi-Newton method in the same spirit. It follows from (4.17) using $J^{-T}=A^{-T}-A^{-T}(J-A)^TJ^{-T}$ that

$$\begin{aligned}\bar{h}^+&=h-A^{-T}r-A^{-T}(J-A)^T\delta-J^{-T}f-J^{-T}F^T\bar{\delta}+g\\&=O(\|h\|^2)-A^{-T}(J-A)^T\delta-J^{-T}f-J^{-T}F^T\bar{\delta}+g.\end{aligned} \tag{4.22}$$

Again $-J^{-T}f$ gives a lower limit on attainable accuracy as (4.18) is still valid, but we must additionally assume that the extra term $-A^{-T}(J-A)^T\delta$ becomes negligible at this level of accuracy. This will depend on the properties of the quasi-Newton method. In addition to this there are cancellation errors given by E_J in (4.10) which contribute to the error matrix F and in fact form the dominant contribution. (For iterative refinement and Newton's method there are no cancellation errors and F is the error in solving $A^T\delta=-r$ and is bounded by a multiple of ε.) From (4.10) it follows that $\|J^{-T}F^T\delta\|\sim\|J^{-1}\|\,\|p\|\varepsilon$ and so we have the interesting result that cancellation errors in J contribute to $\bar{h}^+-h^+$ a term of the same magnitude as that arising directly from $-J^{-T}f$. The sum of these terms gives the limit of attainable accuracy.

In fact it is possible to be more precise about the level of attainable accuracy. Let J^- (that is $J^{(k-1)}$) and x be regarded as exact. For the process of updating J^- and then forming x^+, the dominant rounding error is

$$\begin{aligned}d&=-J^{-T}f-J^{-T}E_J^T\delta\\&=-J^{-T}f-J^{-T}e^-w^{-T}\delta/w^{-T}\delta^-.\end{aligned} \tag{4.23}$$

Using (4.8) it follows that

$$d_i = -\sum_j J_{ij}^{-T} p_j((1+a)\varepsilon_j - a\varepsilon_j^-) \tag{4.24}$$

where $a = w^{-T}\delta / w^{-T}\delta^-$. Assuming the ε_j are independently taken from a uniform distribution in $[-\varepsilon, \varepsilon]$ it follows that $\mathscr{E}(d_i) = 0$ and

$$\mathscr{E}(d_i^2) = \sum (J_{ij}^{-T})^2 p_j^2 b\varepsilon^2 \tag{4.25}$$

where $b = (1+2a+2a^2)/3$. Using the notation of Fletcher (1983) that square brackets round any matrix or vector denote the operation of squaring each element, it is possible to write

$$\mathscr{E}(\|d\|) \leqq (\mathscr{E}(\|d\|^2))^{1/2} = (\mathscr{E}(e^T[d]))^{1/2} = \varepsilon(be^T[J^{-T}][p])^{1/2} \tag{4.26}$$

(here e denotes the vector $(1, 1, \cdots, 1)^T$). Expressions (4.25) and (4.26) give the expectation of the uncertainty in x^+ of the rounding errors on a single iteration, and hence a useful practical measure of attainable accuracy.

Some other recent work on the effect of errors in quasi-Newton methods is given by Dennis and Walker (1983). They assume that a bounded deterioration property holds only for the ideal updates $J^{(k)}$ which would be used if no rounding error were present. They then consider a model in which the updates actually used are allowed to differ from the ideal updates by terms which incorporate the effects of round-off errors. Their results are phrased so that certain "improvement" properties hold only as long as these terms satisfy certain conditions. However the observation in (4.11) that cancellation errors can potentially increase without bound indicates that these conditions will inevitably fail as k becomes large, and this should be taken into account when assessing their theoretical results. The thesis of this paper is that rounding errors in r (either directly or as cancellation errors in J) cause a limit of attainable accuracy of $\sim\|J^{-1}\|\,\|p\|\varepsilon$, and the only way to get higher accuracy is to evaluate r to higher precision (i.e. reduce p) as for iterative refinement. Alternatively there are certain types of problems (for example, those requiring the numerical solution of a differential equation) in which time can be saved by evaluating r to less than single precision. In such cases a higher limit of attainable accuracy could be tolerated by allowing p to increase on earlier iterations. Equations (4.25) and (4.26) would again be useful in relating the accuracy in r and x. Superficially the idea is similar to the inexact Newton approach of Dembo et al. (1982).

Acknowledgments. I am indeed grateful for the constructive comments both of the referee and of John Dennis, for pointing out some errors and misconceptions in an earlier version of this paper.

REFERENCES

M. AL-BAALI AND R. FLETCHER (1983), *Variational methods for nonlinear least squares*, Univ. of Dundee, Dept. Mathematical Sciences Report NA/71, J. Oper. Res. Soc., 36 (1985), pp. 405-421.

R. S. DEMBO, S. C. EISENSTAT AND T. STEIHAUG (1982), *Inexact Newton methods*, SIAM J. Numer. Anal., 19, pp. 400-408.

J. E. DENNIS AND R. B. SCHNABEL (1983), *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*, Prentice-Hall, Englewood Cliffs, NJ.

J. E. DENNIS AND H. F. WALKER (1983), *Inaccuracy in quasi-Newton methods: Local improvement theorems*, Dept. Mathematical Sciences Report 83-11, Rice Univ., Houston, TX.

R. FLETCHER (1972), *Fortran Subroutines for Minimization by Quasi-Newton Methods*, Harwell Report AERE-R7125.

R. FLETCHER (1980), *Practical Methods of Optimization, Volume* 1, *Unconstrained Optimization*, Wiley, Chichester.

——— (1983), *Expected conditioning*, Dept. Mathematical Sciences Report NA/63 (with addendum), Univ. Dundee, IMA, J. Numer. Anal., 5 (1985), pp. 247-273.

R. FLETCHER AND M. J. D. POWELL (1974), *On the modification of LDL^T factorizations*, Math. Comp., 28, pp. 1067-1087.

C. B. MOLER (1967), *Iterative refinement in floating point*, J. Assoc. Comput. Mach., 14, pp. 316-321.

G. W. STEWART (1967), *A modification of Davidon's minimization method to accept difference approximation of derivatives*, J. Assoc. Comput. Mach., 14, pp. 72-83.

J. H. WILKINSON (1965), *The Algebraic Eigenvalue Problem*, Oxford Univ. Press, Oxford.

SIAM J. SCI. STAT. COMPUT.
Vol. 7, No. 4, October 1986

MISCLASSIFICATION PROBABILITIES FOR QUADRATIC DISCRIMINATION*

PUSHPA LATA GUPTA†, JAMES T. RILEY‡ AND THOMAS J. WHITE‡

Abstract. Two p-variate normal populations $N_p(\mu_1, \Sigma_1)$, $N_p(\mu_2, \Sigma_2)$ with parameters known and $\mu_1 \neq \mu_2$, $\Sigma_1 \neq \Sigma_2$ are considered. A method is given to calculate, to a desired accuracy, the probabilities of misclassification (PM's) for quadratic discrimination by using Davies [1980]. Monte Carlo studies, assuming equal priors and equal costs, are conducted.

These studies demonstrate the dependence of the overall PM on a measure m of separation between the populations, used by Marks [1970], Marks and Dunn [1974], and a measure ρ of the divergence between Σ_1 and Σ_2, discussed by Rao [1977]. Tables for dimensions $p = 10$, 25, and 40 displaying the dependence are constructed from these studies. An iterative algorithm for the calculation of m is given. Application of the above results in the large sample situation is discussed.

Key words. multivariate normal populations, Mahalanobis distance, best linear discriminant hyperplane, simultaneous diagonalization, noncentral chi-square distribution, divergence measure

AMS(MOS) subject classification. 62H

1. Introduction.

In this paper we consider two p-variate normal populations $\pi_i \sim N_p(\mu_i, \Sigma_i)$, where μ_i, Σ_i, $i = 1, 2$, are the means and covariance matrices of the given populations. It is assumed that $\mu_1 \neq \mu_2$, $\Sigma_1 \neq \Sigma_2$ and all parameters are known.

It has been noticed by several authors, including Bayne and Tan [1981], that the PM's are not easy to calculate in the case of unequal covariance matrices because percentile points for linear combinations of noncentral chi-square random variables must be calculated. As a result, most researchers have examined the special case $\Sigma_2 = d\Sigma_1$ (Gilbert [1969], Clark et al. [1979]) by using Patnaik's approximation (Patnaik [1949]), which has limited application, to approximate the PM's. Marks and Dunn [1974] consider slightly more general cases, but approximate the PM's using Monte Carlo methods. Bayne and Tan [1981] approximated the PM's only for the bivariate case ($p = 2$). In this paper, a method for the calculation of the PM's to a desired accuracy is given which places no restrictions on the covariance matrices or the dimension p. It uses an algorithm of Davies [1980], and is presented in § 2.

In § 3, Monte Carlo studies, for equal priors and equal costs, are described. The main finding is that the overall PM is dependent within narrow bounds on quantities m and ρ to be defined shortly. In § 4, an iterative algorithm for the calculation of m is given. Section 5 describes how these results can be applied to actual data in large sample situations. An Appendix addresses certain technical aspects of the Monte Carlo studies.

The following notation and definitions will be used, unless otherwise specified, throughout this paper: p_i = prior probability for π_i, $i = 1, 2$; $p_1 + p_2 = 1$; $C(i|j)$ is the cost of classifying a member of π_j to π_i, $i \neq j$; $P(i|j)$ is the probability of classifying a member of π_j to π_i, $i \neq j$; $P = p_1 P(2|1) + p_2 P(1|2)$; $P(2|1)$, $P(1|2)$, and P were referred to as PM's above. $U[1, u]$ denotes the uniform distribution on the interval $[1, u]$.

DEFINITION 1.1. The parameter m is the Euclidean distance from the mean μ_1 of π_1 to the best linear discriminant hyperplane when $P_{BL}(2|1) = P_{BL}(1|2)$, where

* Received by the editors September 18, 1984, and in revised form July 8, 1985.

† Department of Mathematics, University of Maine, Orono, Maine 04469.

‡ Biomathematics Modeling Branch, USAF School of Aerospace Medicine, Brooks Air Force Base, Texas 78235.

$P_{BL}(i|j)$ denotes the probability of misclassifying a member of π_j to π_i for best linear discrimination (Marks [1970, p. 48]). In § 2 we discuss the transformation by the method of simultaneous diagonalization of $N_p(\boldsymbol{\mu}_1, \Sigma_1)$, $N_p(\boldsymbol{\mu}_2, \Sigma_2)$ into the canonical form

$$N_p(\mathbf{0}, I), \qquad N_p(\mathbf{v}, D), \tag{1.1}$$

where $\mathbf{v} = (v_1, \cdots, v_p)$, $\mathbf{d} = (d_1, \cdots, d_p)$, and $D = \text{Diag}[\mathbf{d}]$ is the diagonal matrix with diagonal entries given by $\mathbf{d} = (d_1, \cdots, d_p)$. For this form, m is the Euclidean distance from $\mathbf{0}$ to the point $\mathbf{a} = (a_1, \cdots, a_p)$ satisfying

$$v_k = a_k\left\{1 + d_k\left(\frac{\sum_{i=1}^p a_i^2}{\sum_{i=1}^p d_i a_i^2}\right)^{1/2}\right\} \tag{1.2}$$

for $k = 1, \cdots, p$; see Marks [1970, p. 132]. When $\Sigma_1 = \Sigma_2$, then $m = T/2$, where T is the Mahalanobis distance between the two populations.

Now we will define ρ, which was introduced by Hellinger [1909] and discussed by Rao [1977] and several other authors. It is also a multiple of a term in the Bhattacharya distance (Fukunaga [1972]).

DEFINITION 1.2. Define ρ, which measures the divergence in the covariance matrices Σ_1 and Σ_2, by

$$\rho = \ln\left(\frac{|\Sigma|^2}{|\Sigma_1| \cdot |\Sigma_2|}\right) \tag{1.3}$$

where $\Sigma = (\Sigma_1 + \Sigma_2)/2$. The above expression transforms to

$$\rho = 2\left[-p \ln 2 + \sum_{i=1}^p \ln\left(\sqrt{d_i} + \frac{1}{\sqrt{d_i}}\right)\right] \tag{1.4}$$

for $\pi_1 \sim N_p(\mathbf{0}, I)$, $\pi_2 \sim N_p(\mathbf{v}, D)$.

DEFINITION 1.3. The Mahalanobis distance T between two populations $\pi_1 \sim N_p(\boldsymbol{\mu}_1, \Sigma_1)$, $\pi_2 \sim N_p(\boldsymbol{\mu}_2, \Sigma_2)$ is defined by

$$T^2 = (\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)'\Sigma^{-1}(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)$$

where $\Sigma = p_1\Sigma_1 + p_2\Sigma_2$. In case $\pi_1 \sim N_p(\mathbf{0}, I)$ and $\pi_2 \sim N_p(\mathbf{v}, D)$, T^2 is given by

$$T^2 = \sum_{i=1}^p \frac{v_i^2}{p_1 + p_2 d_i}. \tag{1.5}$$

2. Derivations of expressions for $P(2|1)$ and $P(1|2)$. In this section we will express the quadratic discriminant function (QDF) as a linear combination of noncentral χ^2 variables and a standard normal variable, which is the required form in order to use Davies [1980]. It may again be emphasized here that the elements of Σ_2 are not restricted in any way.

Let $\pi_1 \sim N_p(\boldsymbol{\mu}_1, \Sigma_1)$ and $\pi_2 \sim N_p(\boldsymbol{\mu}_2, \Sigma_2)$, where $\boldsymbol{\mu}_1 \neq \boldsymbol{\mu}_2$ and $\Sigma_1 \neq \Sigma_2$. Let $f_i(\mathbf{x})$ be the density function associated with π_i, $i = 1, 2$. The QDF is given by

$$\begin{aligned} Q(\mathbf{x}) = \ln\frac{f_2(\mathbf{x})}{f_1(\mathbf{x})} &= \frac{1}{2}\ln\frac{|\Sigma_1|}{|\Sigma_2|} + \frac{1}{2}\mathbf{x}'(\Sigma_1^{-1} - \Sigma_2^{-1})\mathbf{x} \\ &\quad + (\boldsymbol{\mu}_2'\Sigma_2^{-1} - \boldsymbol{\mu}_1'\Sigma_1^{-1})\mathbf{x} + \frac{1}{2}(\boldsymbol{\mu}_1'\Sigma_1^{-1}\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2'\Sigma_2^{-1}\boldsymbol{\mu}_2). \end{aligned} \tag{2.1}$$

The allocation rule is to allocate $\mathbf{x}$ to π_2 if

$$Q(\mathbf{x}) > \ln \frac{p_1 C(2|1)}{p_2 C(1|2)} \tag{2.2}$$

and otherwise to allocate it to π_1. Using the method of simultaneous diagonalization, we can find a transformation $A'\Sigma_1^{-1/2}$, where A is orthogonal, which simultaneously converts Σ_1 into the identity matrix and Σ_2 into a diagonal matrix D. $A'\Sigma_1^{-1/2}$ can be taken as the transpose of the matrix of eigenvectors of $\Sigma_1^{-1}\Sigma_2$. More explicitly, let $\Phi = [\boldsymbol{\alpha}_1, \boldsymbol{\alpha}_2, \cdots, \boldsymbol{\alpha}_p]$ where the $\boldsymbol{\alpha}_i$'s are unit eigenvectors of Σ_1, $\Lambda = \text{Diag}[\lambda_1, \lambda_2, \cdots, \lambda_p]$, where the λ_i's are the associated eigenvalues of Σ_1, and $\Psi = [\boldsymbol{\beta}_1, \boldsymbol{\beta}_2, \cdots, \boldsymbol{\beta}_p]$, where the $\boldsymbol{\beta}_i$'s are unit eigenvectors of $\Lambda^{-1/2}\Phi'\Sigma_2\Phi\Lambda^{-1/2}$, which is a positive definite matrix since Σ_2 is positive definite and $\Lambda^{-1/2}\Phi'$ is nonsingular. It can be easily seen that $\Psi'\Lambda^{-1/2}\Phi' = \Psi'\Phi'\Sigma_1^{-1/2} = (\Phi\Psi)'\Sigma_1^{-1/2} = A'\Sigma_1^{-1/2}$, say. Here $\Sigma_1^{-1/2} = \sum_{i=1}^{p} (1/\sqrt{\lambda_i})\boldsymbol{\alpha}_i\boldsymbol{\alpha}_i'$; for more detail see Johnson and Wichern [1982, pp. 52–53]. This transforms Σ_1 into I and Σ_2 into a diagonal matrix D. Let $\mathbf{Y} = A'\Sigma_1^{-1/2}(\mathbf{x} - \boldsymbol{\mu}_1)$. Under this transformation, $\pi_1 \sim N_p(\mathbf{0}, I)$ and $\pi_2 \sim N_p(A'\Sigma_1^{-1/2}(\boldsymbol{\mu}_2 - \boldsymbol{\mu}_1), D) = N_p(\mathbf{v}, D)$, which is the canonical form (1.1).

Under the above transformation the allocation rule (2.2) becomes: Allocate $\mathbf{x}$ to π_2 if

$$\sum_{i=1}^{p} \left(\frac{1}{2}\frac{d_i - 1}{d_i} x_i^2 + \frac{v_i}{d_i} x_i - \frac{1}{2}\frac{v_i^2}{d_i} \right) \geqq \frac{1}{2} \sum_{i=1}^{p} \ln d_i + \ln \frac{p_1 C(2|1)}{p_2 C(1|2)}$$

or

$$\sum_{\substack{i \\ d_i \neq 1}} \left(\frac{1}{2}\frac{d_i - 1}{d_i} x_i^2 + \frac{v_i}{d_i} x_i - \frac{1}{2}\frac{v_i^2}{d_i} \right) + \sum_{\substack{i \\ d_i = 1}} \left(\frac{1}{2}\frac{d_i - 1}{d_i} x_i^2 + \frac{v_i}{d_i} x_i - \frac{1}{2}\frac{v_i^2}{d_i} \right)$$

$$\geqq \frac{1}{2} \sum_{\substack{i \\ d_i \neq 1}} \ln d_i + \frac{1}{2} \sum_{\substack{i \\ d_i = 1}} \ln d_i + \ln \frac{p_1 C(2|1)}{p_2 C(1|2)}.$$

Rewriting, we have

$$\sum_{\substack{i \\ d_i \neq 1}} \frac{d_i - 1}{2d_i} \left(x_i + \frac{v_i}{d_i - 1} \right)^2 + \sum_{\substack{i \\ d_i = 1}} v_i \left(x_i - \frac{1}{2} v_i \right)$$

$$\geqq \frac{1}{2} \sum_{\substack{i \\ d_i \neq 1}} \left[\ln d_i + \frac{v_i^2}{d_i(d_i - 1)} + \frac{v_i^2}{d_i} \right] + \ln \frac{p_1 C(2|1)}{p_2 C(1|2)}. \tag{2.3}$$

The LHS of (2.3) equals

$$\sum_{\substack{i \\ d_i \neq 1}} \frac{d_i - 1}{2d_i} \left(x_i + \frac{v_i}{d_i - 1} \right)^2 + \left(\sum_{\substack{i \\ d_i = 1}} v_i x_i \Big/ \sqrt{\sum_{\substack{i \\ d_i = 1}} v_i^2} \right) \left(\sqrt{\sum_{\substack{i \\ d_i = 1}} v_i^2} \right) - \frac{1}{2} \sum_{\substack{i \\ d_i = 1}} v_i^2.$$

Under π_1, $(x_i + v_i/(d_i - 1))^2$ is noncentral χ^2 with 1 degree of freedom and noncentrality parameter $\delta_i^2 = (v_i/(d_i - 1))^2$ and

$$\left(\sum_{\substack{i \\ d_i = 1}} v_i x_i \right) \Big/ \sqrt{\sum_{\substack{i \\ d_i = 1}} v_i^2} \sim N(0, 1).$$

The LHS of (2.3) can also be written as

$$\sum_{\substack{i\\ d_i\neq 1}} \frac{d_i-1}{2}\left(\frac{x_i+v_i/(d_i-1)}{\sqrt{d_i}}\right)^2+\left(\sum_{\substack{i\\ d_i=1}} v_i(x_i-v_i)\right)\Big/\sqrt{\sum_{\substack{i\\ d_i=1}} v_i^2}\sqrt{\sum_{\substack{i\\ d_i=1}} v_i^2}+\frac{1}{2}\sum_{\substack{i\\ d_i=1}} v_i^2.$$

Under π_2, $(x_i+v_i/(d_i-1)/\sqrt{d_i})^2$ is noncentral χ^2 with 1 degree of freedom and noncentrality parameter $\delta_i^2=d_iv_i^2/(d_i-1)^2$ and

$$\left(\sum_{\substack{i\\ d_i=1}} v_i(x_i-v_i)\right)\Big/\sqrt{\sum_{\substack{i\\ d_i=1}} v_i^2}\sim N(0,1).$$

Therefore,

(2.4)

$$P(2|1)=P\left(\sum_{\substack{i\\ d_i\neq 1}} \frac{d_i-1}{2d_i}\left(x_i+\frac{v_i}{d_i-1}\right)^2+\sum_{\substack{i\\ d_i=1}} v_i\left(x_i-\frac{1}{2}v_i\right)\geqq K|\mathbf{x}\in\pi_1\right)$$

$$=P\left(\sum_{\substack{i\\ d_i\neq 1}} \frac{d_i-1}{2d_i}\left(x_i+\frac{v_i}{d_i-1}\right)^2+\sqrt{\sum_{\substack{i\\ d_i=1}} v_i^2}\left(\sum_{\substack{i\\ d_i=1}} v_ix_i\Big/\sqrt{\sum_{\substack{i\\ d_i=1}} v_i^2}\right)\geqq K+\frac{1}{2}\sum_{\substack{i\\ d_i=1}} v_i^2\right)$$

where $(x_i+v_i/(d_i-1))^2\sim\chi^2(1,(v_i/(d_i-1))^2)$ and

$$\left(\sum_{\substack{i\\ d_i=1}} v_ix_i\right)\Big/\sqrt{\sum_{\substack{i\\ d_i=1}} v_i^2}\sim N(0,1).$$

Also,

$$P(1|2)=P\left(\sum_{\substack{i\\ d_i\neq 1}} \frac{d_i-1}{2d_i}\left(x_i+\frac{v_i}{d_i-1}\right)^2+\sum_{\substack{i\\ d_i=1}} v_i\left(x_i-\frac{1}{2}v_i\right)<K|\mathbf{x}\in\pi_2\right)$$

(2.5)
$$=P\left(\sum_{\substack{i\\ d_i\neq 1}} \frac{d_i-1}{2}\left(\frac{x_i+v_i/(d_i-1)}{\sqrt{d_i}}\right)^2\right.$$

$$\left.+\sqrt{\sum_{\substack{i\\ d_i=1}} v_i^2}\left(\sum_{\substack{i\\ d_i=1}} v_i(x_i-v_i)\Big/\sqrt{\sum_{\substack{i\\ d_i=1}} v_i^2}\right)<K-\frac{1}{2}\sum_{\substack{i\\ d_i=1}} v_i^2\right)$$

where

$$\left(\frac{x_i+v_i/(d_i-1)}{\sqrt{d_i}}\right)^2\sim\chi^2\left(1,\frac{v_i^2d_i}{(d_i-1)^2}\right)$$

and

$$\left(\sum_{\substack{i\\ d_i=1}} v_i(x_i-v_i)\right)\Big/\sqrt{\sum_{\substack{i\\ d_i=1}} v_i^2}\sim N(0,1).$$

Here

$$K=\frac{1}{2}\sum_{\substack{i\\ d_i\neq 1}}\left[\ln d_i+\frac{v_i^2}{d_i(d_i-1)}+\frac{v_i^2}{d_i}\right]+\ln\frac{p_1C(2|1)}{p_2C(1|2)}$$

$$=\frac{1}{2}\sum_{\substack{i\\ d_i\neq 1}}\left[\ln d_i+\frac{v_i^2}{d_i-1}\right]+\ln\frac{p_1C(2|1)}{p_2C(1|2)}.$$

TABLE 1

$p = 10$.

For each pair (ρ_0, m_0), the values given are $\bar{P}$, S_p (in parentheses), $P_{\min}$, $P_{\max}$, R_p and r.

ρ_0 \ m_0	0.250	0.500	0.750	1.000	1.250	1.500	1.750
0.500	0.299 (0.002)	0.248 (0.002)	0.190 (0.001)	0.136 (0.001)	0.092 (0.001)	0.058 (0.000)	0.035 (0.000)
	0.294 0.304	0.244 0.256	0.186 0.194	0.133 0.140	0.090 0.093	0.057 0.060	0.035 0.036
	0.010	0.012	0.007	0.007	0.003	0.002	0.002
	−0.993	−0.984	−0.977	−0.962	−0.971	−0.971	−0.975
1.000	0.239 (0.002)	0.205 (0.003)	0.160 (0.002)	0.116 (0.002)	0.080 (0.001)	0.051 (0.001)	0.031 (0.000)
	0.234 0.244	0.198 0.212	0.156 0.168	0.113 0.122	0.077 0.082	0.050 0.053	0.030 0.032
	0.010	0.014	0.013	0.010	0.005	0.004	0.002
	−0.989	−0.972	−0.964	−0.948	−0.964	−0.940	−0.957
1.500	0.197 (0.003)	0.171 (0.003)	0.136 (0.002)	0.100 (0.002)	0.069 (0.001)	0.045 (0.001)	0.027 (0.001)
	0.188 0.209	0.162 0.177	0.129 0.140	0.095 0.106	0.066 0.073	0.043 0.047	0.026 0.029
	0.021	0.014	0.011	0.011	0.007	0.004	0.003
	−0.964	−0.971	−0.965	−0.944	−0.927	−0.939	−0.887
2.000	0.165 (0.004)	0.144 (0.003)	0.116 (0.002)	0.086 (0.002)	0.060 (0.002)	0.039 (0.001)	0.024 (0.001)
	0.156 0.177	0.135 0.151	0.109 0.120	0.080 0.092	0.056 0.066	0.037 0.045	0.022 0.026
	0.021	0.015	0.011	0.011	0.010	0.007	0.003
	−0.957	−0.952	−0.962	−0.921	−0.899	−0.910	−0.908

2.500	0.139 (0.003)	0.122 (0.003)	0.099 (0.003)	0.075 (0.002)	0.052 (0.001)	0.034 (0.001)	0.021 (0.001)
	0.128 0.146	0.114 0.129	0.091 0.104	0.070 0.078	0.049 0.055	0.032 0.037	0.019 0.024
	0.018	0.016	0.012	0.009	0.007	0.005	0.005
	−0.951	−0.944	−0.939	−0.926	−0.928	−0.898	−0.849
3.000	0.119 (0.003)	0.104 (0.003)	0.085 (0.003)	0.064 (0.002)	0.046 (0.002)	0.030 (0.001)	0.019 (0.001)
	0.110 0.127	0.096 0.110	0.076 0.096	0.057 0.070	0.041 0.049	0.027 0.034	0.017 0.021
	0.017	0.014	0.019	0.012	0.008	0.006	0.004
	−0.934	−0.937	−0.893	−0.874	−0.892	−0.863	−0.827
3.500	0.100 (0.003)	0.089 (0.003)	0.073 (0.003)	0.056 (0.002)	0.040 (0.002)	0.026 (0.001)	0.016 (0.001)
	0.093 0.107	0.079 0.095	0.065 0.081	0.051 0.060	0.035 0.044	0.023 0.030	0.014 0.018
	0.014	0.016	0.015	0.010	0.009	0.007	0.004
	−0.936	−0.916	−0.904	−0.860	−0.830	−0.772	−0.850
4.000	0.086 (0.003)	0.076 (0.003)	0.063 (0.003)	0.048 (0.002)	0.035 (0.002)	0.023 (0.001)	0.014 (0.001)
	0.075 0.095	0.067 0.085	0.056 0.067	0.042 0.052	0.030 0.039	0.020 0.027	0.013 0.016
	0.020	0.018	0.012	0.010	0.009	0.007	0.004
	−0.909	−0.884	−0.883	−0.858	−0.823	−0.789	−0.824
4.500	0.073 (0.003)	0.066 (0.003)	0.054 (0.003)	0.042 (0.002)	0.030 (0.001)	0.020 (0.001)	0.013 (0.001)
	0.063 0.078	0.058 0.073	0.047 0.059	0.035 0.045	0.025 0.033	0.017 0.024	0.011 0.015
	0.016	0.015	0.012	0.010	0.008	0.007	0.004
	−0.894	−0.868	−0.854	−0.835	−0.815	−0.706	−0.789
5.000	0.063 (0.003)	0.056 (0.003)	0.047 (0.002)	0.037 (0.002)	0.026 (0.002)	0.018 (0.001)	0.011 (0.001)
	0.053 0.070	0.048 0.061	0.040 0.053	0.030 0.042	0.023 0.030	0.015 0.020	0.010 0.013
	0.018	0.014	0.013	0.012	0.007	0.006	0.004
	−0.855	−0.858	−0.818	−0.762	−0.751	−0.658	−0.724

TABLE 2

$p = 25$.

For each pair (ρ_0, m_0), the values given are $\bar{P}$, S_p, (in parentheses), $P_{\min}$, $P_{\max}$, R_p, and r.

ρ_0 \ m_0	0.250	0.500	0.750	1.000	1.250	1.500	1.750
0.500	0.293 (0.001)	0.244 (0.001)	0.186 (0.000)	0.134 (0.000)	0.090 (0.000)	0.058 (0.000)	0.035 (0.000)
	0.291 0.297	0.242 0.246	0.185 0.187	0.133 0.134	0.090 0.091	0.057 0.058	0.035 0.035
	0.005	0.004	0.002	0.002	0.002	0.001	0.000
	−0.991	−0.992	−0.989	−0.990	−0.979	−0.983	−0.987
1.000	0.233 (0.001)	0.199 (0.001)	0.156 (0.001)	0.113 (0.000)	0.077 (0.000)	0.050 (0.000)	0.030 (0.000)
	0.231 0.236	0.197 0.201	0.154 0.158	0.112 0.114	0.077 0.078	0.049 0.051	0.030 0.031
	0.005	0.004	0.003	0.002	0.002	0.001	0.001
	−0.985	−0.989	−0.978	−0.985	−0.977	−0.968	−0.974
1.500	0.191 (0.001)	0.165 (0.001)	0.131 (0.001)	0.096 (0.001)	0.066 (0.000)	0.043 (0.000)	0.026 (0.000)
	0.188 0.197	0.162 0.169	0.129 0.133	0.095 0.098	0.066 0.067	0.043 0.044	0.026 0.027
	0.009	0.006	0.004	0.003	0.002	0.001	0.001
	−0.977	−0.978	−0.982	−0.970	−0.966	−0.971	−0.955
2.000	0.158 (0.001)	0.138 (0.001)	0.111 (0.001)	0.082 (0.000)	0.057 (0.000)	0.037 (0.000)	0.023 (0.000)
	0.156 0.162	0.134 0.141	0.109 0.116	0.081 0.084	0.057 0.058	0.037 0.038	0.022 0.024
	0.006	0.007	0.007	0.003	0.001	0.001	0.001
	−0.976	−0.973	−0.966	−0.974	−0.967	−0.952	−0.937

2.500	0.133 (0.001)	0.116 (0.001)	0.094 (0.001)	0.071 (0.001)	0.049 (0.000)	0.032 (0.000)	0.020 (0.000)
	0.131 0.136	0.114 0.121	0.092 0.097	0.069 0.072	0.048 0.051	0.032 0.033	0.020 0.020
	0.005	0.007	0.005	0.003	0.003	0.002	0.001
	−0.973	−0.957	−0.955	−0.954	−0.944	−0.945	−0.938
3.000	0.112 (0.001)	0.099 (0.001)	0.080 (0.001)	0.061 (0.001)	0.043 (0.000)	0.028 (0.000)	0.017 (0.000)
	0.108 0.116	0.097 0.102	0.078 0.083	0.058 0.062	0.042 0.044	0.028 0.029	0.017 0.018
	0.008	0.005	0.005	0.004	0.003	0.001	0.001
	−0.959	−0.961	−0.938	−0.945	−0.943	−0.943	−0.930
3.500	0.095 (0.001)	0.084 (0.001)	0.069 (0.001)	0.052 (0.001)	0.037 (0.000)	0.024 (0.000)	0.015 (0.000)
	0.091 0.099	0.082 0.088	0.067 0.072	0.050 0.056	0.036 0.038	0.024 0.026	0.015 0.016
	0.008	0.006	0.005	0.006	0.002	0.002	0.002
	−0.941	−0.933	−0.946	−0.918	−0.935	−0.917	−0.885
4.000	0.081 (0.001)	0.072 (0.001)	0.059 (0.001)	0.045 (0.001)	0.032 (0.000)	0.021 (0.000)	0.013 (0.000)
	0.079 0.084	0.070 0.076	0.058 0.061	0.044 0.046	0.031 0.033	0.021 0.022	0.013 0.014
	0.006	0.006	0.004	0.002	0.003	0.002	0.001
	−0.951	−0.947	−0.948	−0.940	−0.912	−0.920	−0.904
4.500	0.069 (0.001)	0.062 (0.001)	0.051 (0.001)	0.039 (0.001)	0.028 (0.000)	0.019 (0.000)	0.012 (0.000)
	0.066 0.072	0.059 0.064	0.049 0.053	0.037 0.041	0.026 0.030	0.018 0.020	0.011 0.012
	0.006	0.005	0.004	0.004	0.003	0.002	0.001
	−0.934	−0.947	−0.942	−0.899	−0.904	−0.907	−0.904
5.000	0.060 (0.001)	0.053 (0.001)	0.044 (0.001)	0.034 (0.001)	0.024 (0.000)	0.016 (0.000)	0.010 (0.000)
	0.057 0.064	0.050 0.055	0.041 0.046	0.032 0.035	0.023 0.026	0.015 0.017	0.010 0.011
	0.007	0.004	0.004	0.003	0.003	0.001	0.001
	−0.911	−0.947	−0.924	−0.912	−0.880	−0.910	−0.899

TABLE 3

$p = 40$.

For each pair (ρ_0, m_0), *the values given are* $\bar{P}$, S_p (*in parentheses*), $P_{\min}$, $P_{\max}$, R_p, *and* r.

ρ_0 \ m_0	0.250	0.500	0.750	1.000	1.250	1.500	1.750
0.500	0.292 (0.000)	0.242 (0.000)	0.185 (0.000)	0.133 (0.000)	0.090 (0.000)	0.057 (0.000)	0.035 (0.000)
	0.291 0.293	0.242 0.243	0.185 0.186	0.133 0.133	0.090 0.090	0.057 0.058	0.035 0.035
	0.002	0.002	0.001	0.001	0.000	0.000	0.000
	−0.996	−0.994	−0.992	−0.991	−0.989	−0.987	−0.991
1.000	0.231 (0.001)	0.197 (0.001)	0.154 (0.000)	0.112 (0.000)	0.077 (0.000)	0.049 (0.000)	0.030 (0.000)
	0.229 0.234	0.196 0.200	0.153 0.155	0.112 0.113	0.076 0.077	0.049 0.050	0.030 0.030
	0.005	0.004	0.002	0.001	0.001	0.000	0.000
	−0.985	−0.967	−0.989	−0.988	−0.990	−0.985	−0.974
1.500	0.188 (0.001)	0.163 (0.001)	0.129 (0.000)	0.095 (0.000)	0.066 (0.000)	0.043 (0.000)	0.026 (0.000)
	0.187 0.193	0.162 0.166	0.128 0.131	0.095 0.096	0.065 0.067	0.042 0.043	0.026 0.026
	0.006	0.004	0.003	0.002	0.001	0.001	0.000
	−0.972	−0.971	−0.972	−0.976	−0.966	−0.973	−0.974
2.000	0.156 (0.001)	0.136 (0.001)	0.109 (0.001)	0.081 (0.000)	0.056 (0.000)	0.037 (0.000)	0.023 (0.000)
	0.155 0.160	0.135 0.139	0.108 0.111	0.080 0.083	0.056 0.057	0.036 0.038	0.022 0.023
	0.005	0.004	0.003	0.002	0.001	0.001	0.001
	−0.962	−0.970	−0.953	−0.977	−0.977	−0.959	−0.958

2.500	0.131 (0.001)	0.114 (0.001)	0.092 (0.001)	0.069 (0.000)	0.048 (0.000)	0.032 (0.000)	0.020 (0.000)
	0.130 0.136	0.113 0.118	0.092 0.095	0.069 0.071	0.048 0.049	0.031 0.032	0.019 0.020
	0.006	0.004	0.003	0.002	0.001	0.001	0.001
	−0.959	−0.959	−0.960	−0.962	−0.973	−0.965	−0.950
3.000	0.110 (0.001)	0.097 (0.001)	0.079 (0.000)	0.059 (0.000)	0.042 (0.000)	0.028 (0.000)	0.017 (0.000)
	0.109 0.113	0.095 0.100	0.078 0.081	0.059 0.061	0.041 0.043	0.027 0.028	0.017 0.017
	0.004	0.004	0.003	0.002	0.002	0.001	0.001
	−0.959	−0.942	−0.965	−0.953	−0.941	−0.963	−0.948
3.500	0.093 (0.001)	0.082 (0.001)	0.067 (0.001)	0.051 (0.000)	0.036 (0.000)	0.024 (0.000)	0.015 (0.000)
	0.091 0.096	0.081 0.085	0.066 0.071	0.050 0.052	0.035 0.037	0.024 0.024	0.015 0.015
	0.004	0.005	0.004	0.002	0.001	0.001	0.001
	−0.960	−0.943	−0.950	−0.936	−0.964	−0.952	−0.939
4.000	0.079 (0.001)	0.070 (0.001)	0.058 (0.001)	0.044 (0.000)	0.031 (0.000)	0.021 (0.000)	0.013 (0.000)
	0.078 0.082	0.069 0.072	0.056 0.060	0.043 0.045	0.031 0.032	0.020 0.021	0.013 0.013
	0.004	0.003	0.004	0.002	0.001	0.001	0.001
	−0.961	−0.946	−0.909	−0.937	−0.956	−0.948	−0.939
4.500	0.068 (0.001)	0.060 (0.000)	0.049 (0.000)	0.038 (0.000)	0.027 (0.000)	0.018 (0.000)	0.011 (0.000)
	0.066 0.070	0.058 0.061	0.049 0.051	0.037 0.039	0.027 0.028	0.018 0.019	0.011 0.012
	0.004	0.003	0.002	0.003	0.001	0.002	0.001
	−0.945	−0.966	−0.959	−0.903	−0.940	−0.897	−0.909
5.000	0.058 (0.001)	0.052 (0.001)	0.043 (0.000)	0.033 (0.000)	0.023 (0.000)	0.016 (0.000)	0.010 (0.000)
	0.057 0.061	0.049 0.055	0.042 0.045	0.032 0.033	0.023 0.024	0.015 0.016	0.010 0.010
	0.004	0.006	0.003	0.001	0.001	0.001	0.000
	−0.938	−0.923	−0.942	−0.952	−0.911	−0.913	−0.945

TABLE 4

$p = 10$.

For each pair (ρ_0, T_0), *the values given are* $\bar{P}$, S_p *(in parentheses)*, $P_{\min}$, $P_{\max}$, R_p *and* r.

ρ_0 \ T_0	0.500	1.000	1.500	2.000	2.500	3.000	3.500
0.500	0.309 (0.003)	0.280 (0.004)	0.241 (0.006)	0.200 (0.008)	0.160 (0.010)	0.123 (0.010)	0.093 (0.008)
	0.302 0.316	0.271 0.286	0.223 0.252	0.176 0.216	0.136 0.183	0.100 0.148	0.075 0.115
	0.014	0.016	0.030	0.040	0.047	0.047	0.040
	−0.990	−0.985	−0.934	−0.843	−0.391	0.524	0.832
1.000	0.246 (0.004)	0.227 (0.005)	0.200 (0.007)	0.167 (0.008)	0.136 (0.010)	0.108 (0.011)	0.081 (0.011)
	0.236 0.257	0.211 0.236	0.180 0.211	0.146 0.186	0.111 0.157	0.081 0.132	0.057 0.104
	0.021	0.025	0.031	0.040	0.045	0.051	0.046
	−0.978	−0.967	−0.919	−0.782	−0.436	0.447	0.794
1.500	0.202 (0.004)	0.187 (0.004)	0.165 (0.006)	0.141 (0.008)	0.116 (0.012)	0.091 (0.012)	0.070 (0.011)
	0.191 0.211	0.174 0.194	0.147 0.177	0.120 0.158	0.088 0.142	0.064 0.110	0.046 0.094
	0.021	0.020	0.030	0.038	0.054	0.046	0.048
	−0.970	−0.969	−0.916	−0.780	−0.261	0.414	0.768
2.000	0.168 (0.004)	0.157 (0.004)	0.141 (0.005)	0.119 (0.008)	0.098 (0.010)	0.078 (0.011)	0.063 (0.010)
	0.156 0.181	0.144 0.168	0.125 0.150	0.100 0.136	0.071 0.119	0.056 0.101	0.041 0.095
	0.024	0.024	0.026	0.036	0.048	0.045	0.054
	−0.958	−0.952	−0.892	−0.687	−0.169	0.419	0.622

2.500	0.142 (0.004)	0.133 (0.004)	0.119 (0.006)	0.102 (0.008)	0.084 (0.009)	0.067 (0.010)	0.052 (0.011)
	0.133 0.150	0.119 0.140	0.101 0.132	0.082 0.117	0.066 0.103	0.047 0.090	0.030 0.076
	0.017	0.021	0.031	0.034	0.037	0.043	0.046
	−0.953	−0.929	−0.844	−0.615	−0.264	0.354	0.732
3.000	0.120 (0.004)	0.112 (0.004)	0.101 (0.005)	0.089 (0.006)	0.073 (0.009)	0.059 (0.010)	0.045 (0.010)
	0.109 0.131	0.102 0.121	0.084 0.113	0.072 0.102	0.049 0.087	0.037 0.078	0.023 0.067
	0.022	0.020	0.029	0.031	0.038	0.041	0.044
	−0.926	−0.931	−0.855	−0.720	−0.127	0.058	0.674
3.500	0.102 (0.004)	0.096 (0.004)	0.086 (0.005)	0.076 (0.006)	0.062 (0.008)	0.052 (0.008)	0.040 (0.009)
	0.092 0.110	0.083 0.102	0.074 0.100	0.057 0.087	0.040 0.075	0.035 0.069	0.020 0.057
	0.018	0.019	0.026	0.030	0.035	0.033	0.037
	−0.920	−0.910	−0.828	−0.716	−0.239	0.158	0.627
4.000	0.087 (0.003)	0.082 (0.003)	0.075 (0.005)	0.065 (0.006)	0.055 (0.007)	0.045 (0.007)	0.035 (0.009)
	0.076 0.094	0.073 0.089	0.057 0.083	0.047 0.076	0.034 0.069	0.028 0.058	0.016 0.054
	0.019	0.016	0.026	0.029	0.036	0.030	0.039
	−0.906	−0.895	−0.815	−0.619	−0.281	0.168	0.631
4.500	0.075 (0.004)	0.070 (0.004)	0.064 (0.004)	0.056 (0.006)	0.047 (0.006)	0.037 (0.007)	0.030 (0.008)
	0.063 0.082	0.057 0.078	0.053 0.073	0.043 0.068	0.033 0.059	0.022 0.055	0.013 0.056
	0.019	0.021	0.020	0.025	0.026	0.033	0.042
	−0.853	−0.855	−0.805	−0.584	−0.277	0.325	0.608
5.000	0.064 (0.003)	0.061 (0.003)	0.055 (0.004)	0.049 (0.004)	0.041 (0.006)	0.033 (0.007)	0.026 (0.006)
	0.054 0.070	0.052 0.066	0.042 0.062	0.037 0.059	0.025 0.052	0.019 0.047	0.013 0.038
	0.016	0.014	0.021	0.022	0.026	0.028	0.025
	−0.849	−0.860	−0.799	−0.706	−0.217	0.321	0.622

Therefore $P(2|1)$ and $P(1|2)$, given by (2.4) and (2.5), respectively, are in the form required by Davies [1980].

3. Monte Carlo experiment. In this section we describe a Monte Carlo experiment, the main objective of which was to study the dependence of P on ρ in combination with either m or T. Equal priors and equal costs are assumed. After describing the structure of the experiment, we discuss the results for two cases:

Case 1. Using ρ and m, the results are given in Tables 1, 2, and 3 for $p = 10$, 25 and 40, respectively; P varies within rather narrow bounds for given ρ and m.

Case 2. Using ρ and T, the results are given in Table 4 for $p = 10$; P varies much more widely, rendering the table less useful than the previous ones. Finally, we return to Case 1 and discuss the behaviour of $P(2|1)$ and $P(1|2)$.

The Monte Carlo experiment is conducted under the canonical conditions of (1.1). We first describe Case 1 and then the modification necessary for Case 2. ρ_0 and m_0 assume the values .5, 1.0, $\cdots$, 5.0 and .25, .50, $\cdots$, 1.75, respectively. For each (ρ_0, m_0), $P = p_1 P(2|1) + p_2 P(1|2)$ is computed using Davies' algorithm with (in his notation) LIM = 10,000 and accuracy ACC = .001 for each of 100 pairs $(\mathbf{d}, \mathbf{v})$ generated in a manner to be described shortly. Then the mean $\bar{P}$, standard deviation S_p, minimum $P_{\min}$, maximum $P_{\max}$, range $R_p = P_{\max} - P_{\min}$, and sample correlation coefficient r for $P(2|1)$ and $P(1|2)$ are computed.

$\mathbf{d}$ is generated as follows:

Step 1. Randomly generate $d_1, \cdots, d_{p-1}$ such that $d_i \sim U[1, u]$ for $i = 1, \cdots, p-1$. (The choice of u is described in the Appendix.)

Step 2. Determine d_p algebraically by the requirement $\rho = \rho_0$.

Step 3. Take the reciprocal of each d_i with probability 1/2. (Note that this step does not change the value of ρ, since (1.4) is invariant under $d_i \to 1/d_i$. Since P is not invariant, the experiment is lent greater generality.) Note that if the condition

$$2 \sum_{i=1}^{k} \ln (\sqrt{d_i} + 1/\sqrt{d_i}) \leqq \rho_0 + 2(p-1)(\ln 2)$$

is not met for some k in Step 1, the equation $\rho = \rho_0$ cannot be satisfied. Thus this condition is checked each time a d_i is generated, and if it is not met, Step 1 is repeated beginning with a new d_1.

After $\mathbf{d}$ is obtained, $\mathbf{a}$ is determined by generating a vector $\mathbf{w} = (w_1, \cdots, w_p)$ such that $w_i \sim U[0, 1]$ for $i = 1, \cdots, p$, and then taking $\mathbf{a} = (m_0/\|\mathbf{w}\|)\mathbf{w}$, where $\|\mathbf{w}\| = (\sum_{i=1}^{p} w_i^2)^{1/2}$. Then $\mathbf{v}$ is determined from $\mathbf{a}$ using (1.2).

For Case 2, T plays the role of m with these changes: (1) T_0 assumes the values .5, 1.0, $\cdots$, 3.5. (2) $\mathbf{v}$ is determined from $\mathbf{u} = (1/\|\mathbf{w}\|)\mathbf{w}$ by the requirement $\mathbf{v} = T\mathbf{u}$ and the use of (1.5) with $p_1 = p_2 = 1/2$.

For Case 1, inspection of Tables 1, 2 and 3 shows that the variability in P for any (ρ_0, m_0) small enough so that P can be predicted within rather narrow bounds with reasonable confidence. Note that $\bar{P}$ decreases across each row and down each column; if this were not the case, the use of ρ and m in combination would be questionable.

A comparison of Tables 1, 2 and 3 with one another shows that corresponding values of $\bar{P}$ in the tables differ very little; one notes a very slight downward trend, most noticeable for the smaller values of ρ_0 and m_0, as p increases. The variability also decreases slightly for the smaller values of ρ_0 and m_0 as p increases. By using these tables, one can estimate values for P for the given values of ρ_0 and m_0 for $10 \leqq p \leqq 40$.

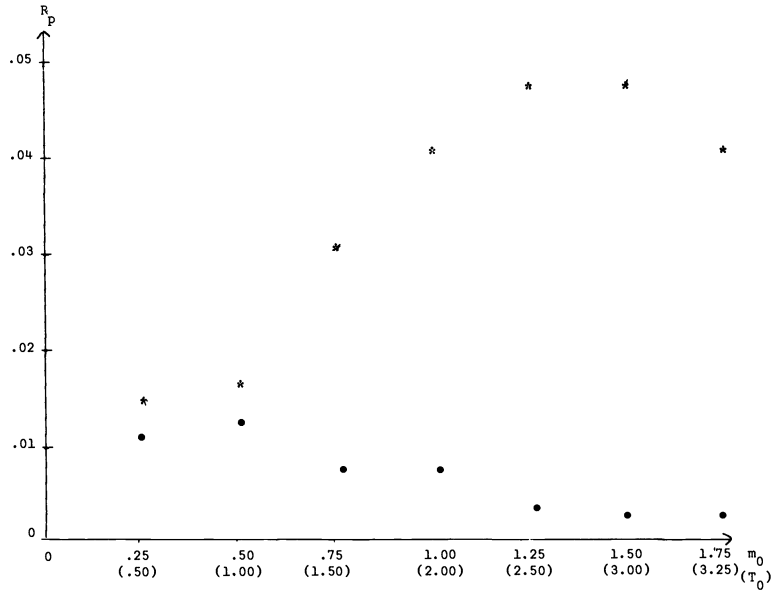


FIG. 1. $\rho_0 = .5$. •—(m_0, R_p), *—(T_0, R_p).

For Case 2, T might seem a more "natural" measure of separation than m to use with the QDF. However, the results given in Table 4 for $p = 10$ show that because of the increased variability it is much less useful in combination with ρ for the prediction of P. This increased variability is depicted in Fig. 1 for the case of $\rho_0 = .5$.

Does the dependence of P within narrow bounds on ρ and m in Case 1 result from a similar dependence of $P(2|1)$ and $P(1|2)$? It does not. Tables for $P(2|1)$ and $P(1|2)$ for $p = 10$, 25 and 40 were also computed. The mean values $\overline{P(2|1)}$ and $\overline{P(1|2)}$ were generally close to $\bar{P}$, but the variability was much greater for the smaller values of ρ_0 and m_0. For example, when $\rho_0 = .5$, $m_0 = .25$ and $p = 10$, $\bar{P}(2|1) = .288$ with s.d. $= .027$, minimum $= .244$ and maximum $= .351$. The much smaller variability for P results from the pronounced negative correlation between $P(2|1)$ and $P(1|2)$; sample values for this correlation are given in Tables 1-3.

It should be realized that the results shown in the Tables 1-4 are dependent on the choice of method of generating the vector $\mathbf{d}$.[1]

4. Computation of m. In order to use Tables 1-3, it is necessary to compute m and ρ. For m this requires that one first perform the simultaneous diagonalization described in § 2. Once this is done, m can be computed by following a simple iterative algorithm, given here as a FORTRAN subroutine; see Fig. 2.

Testing the performance of the subroutine. As part of the Monte Carlo experiment, this routine was employed to compute an estimate $\hat{m}$ of m for each point $\mathbf{a}$ generated for Tables 1-3 (a total of 7,000 per table). Let $e = \hat{m} - m$ be the error and let n be the number of iterations used in the computation of $\hat{m}$. The summary statistics given in Tables 5 and 6 indicate the excellent performance of the algorithm.

5. Applications. Under the assumptions of § 2, the QDF is the optimal discrimination algorithm. Suppose, however, that these assumptions are not met but rather that

[1] The referee has suggested that the slight dependence of $\bar{P}$ on p might be due to the way of generating the d_i.

```
      SUBROUTINE COMPM (V,DIAG,A,P,NIT,NFLAG)
      IMPLICIT REAL*8(A-H,O-Z)
      INTEGER P
      DIMENSION V(1),DIAG(1),A(1),AT(40)
      NFLAG=0
      DO 10 I=1,P
   10 A(I)=V(I)/2.DO
      NIT=0
   20 NIT=NIT+1
      SUM1=0.DO
      SUM2=0.DO
      DO 30 I=1,P
      AI=A(I)
      AT(I)=AI
      AI=AI*AI
      SUM1=SUM1+AI
   30 SUM2=SUM2+DIAG(I)*AI
      ERR=0.DO
      DO 40 I=1,P
      AI=V(I)/(1.DO+DIAG(I)*DSQRT(SUM1/SUM2))
      ERR=DMAX1(ERR,DABS(AI/AT(I)-1.DO))
   40 A(I)=AI
      IF(NIT.GT.50) GO TO 100
      IF(ERR.GT.1.D-3) GO TO 20
  100 CONTINUE
      IF(NIT.GT.50)NFLAG=1
      RETURN
      END
```

FIG. 2

TABLE 5

p	Mean error*	Mean square error†	Maximum error
10	6.0×10^{-5}	4.3×10^{-9}	5.5×10^{-4}
25	3.0×10^{-5}	9.0×10^{-10}	3.2×10^{-4}
40	2.0×10^{-5}	4.0×10^{-10}	2.2×10^{-4}

* Mean error $= (\sum e)/7{,}000$.
† Mean square error $= (\sum e^2)/7{,}000$.

TABLE 6

p	Mean for n	s.d. for n	Maximum for n
10	5.46	1.34	16
25	4.39	.79	11
40	3.98	.64	8

large samples are available from each population, permitting good estimation of μ_1, μ_2, Σ_1 and Σ_2 and also supporting the claim that the populations are approximately multivariate normal. Various studies indicate that the QDF is more sensitive than the linear discriminant function (LDF) to departures from normality and inaccuracy in the estimation of parameters.

One can compute the LDF and QDF misclassification rates for the two methods as though the estimates were the true parameters and multivariate normality were perfectly met; if the computed rates differ little, the LDF may perform at least as well as the QDF on the actual data, and then the LDF is preferred because of its smaller computational and storage requirements (especially for large p) and also because of its greater ease of geometrical interpretation. Thus one turns to the QDF only if its computed misclassification rate is markedly lower.

The computations of the rate for the LDF is straightforward and well known (Gilbert [1969]). The rate for the QDF can be computed to any desired accuracy by the method of § 2; or it can be approximated using Tables 1-3 of § 3 if $10 \leqq p \leqq 40$. Each approach requires the performance of simultaneous diagonalization; one then either applies Davies' algorithm or computes m by the method of § 4 and consults the tables. As an example of the latter, if $10 < p < 25$, one would consult both Tables 1 and 2; interpolation might be necessary. The result would only be approximate, but if the rate was far better than the computed rate for the LDF, this would be an adequate basis for the choice of the QDF.

Appendix. The values for u used in the generation of $d_1, \cdots, d_{p-1}$ in the Monte Carlo experiment were determined by a separate program. Let $\rho_i = (.5)i$, and let u_i be the value used in Step 1 for a given ρ_i, $i = 1, 2, \cdots, 10$. Since the values of u_i depend on p as well as on ρ_i, $u_1, u_2, \cdots, u_{10}$ were determined as follows for $p = 10$, 25 and 40: u was assigned the values 1.5, 1.6, $\cdots$, 9.0. For each value, 1,000 vectors $\mathbf{d}$, where $d_i \sim U[1, u]$ for all i, were generated. For each such $\mathbf{d}$, ρ was computed, and for the resulting 1,000 values, the mean $\bar{\rho}$ was computed. u_i is the first value of u such that $\bar{\rho} \geqq \rho_i$, $i = 1, 2, \cdots, 10$.

Tables A1-A3 give the values for u_i, $i = 1, 2, \cdots, 10$, for $p = 10$, 25 and 40. Since d_p is not drawn from a uniform distribution, descriptive statistics are provided for the set of d_p's exceeding u_i, $i = 1, 2, \cdots, 10$, for the runs which generate Tables 1-3.

TABLE A1
$p = 10$.

ρ_i	u_i	Q^*	d_p†			
			MEAN	SD	MIN	MAX
0.500	2.100	0.329	2.443	0.258	2.104	3.326
1.000	2.700	0.410	3.498	0.654	2.714	6.501
1.500	3.400	0.367	4.629	1.039	3.401	9.231
2.000	4.000	0.409	6.049	1.842	4.030	13.069
2.500	4.600	0.424	7.362	2.505	4.602	16.916
3.000	5.200	0.409	9.280	4.373	5.209	44.170
3.500	5.900	0.437	10.949	5.861	5.905	48.860
4.000	6.700	0.394	13.578	8.430	6.729	59.071
4.500	7.400	0.420	15.056	8.658	7.410	67.990
5.000	8.200	0.423	18.335	12.485	8.212	128.184

* Q = Proposition of cases such that $d_p > u_i$.
† Statistics computed only for those cases such that $d_p > u_i$.

TABLE A2
$p = 25$.

ρ_i	u_i	Q^*	d_p†			
			MEAN	SD	MIN	MAX
0.500	1.600	0.530	1.919	0.252	1.600	2.986
1.000	2.000	0.340	2.492	0.409	2.003	4.539
1.500	2.200	0.567	3.075	0.710	2.201	5.558
2.000	2.500	0.507	3.536	0.941	2.504	7.529
2.500	2.700	0.569	4.304	1.495	2.701	11.459
3.000	3.000	0.539	4.799	1.654	3.002	15.122
3.500	3.200	0.553	5.897	2.738	3.202	23.314
4.000	3.500	0.539	6.444	3.014	3.500	20.592
4.500	3.700	0.561	7.300	3.883	3.704	29.346
5.000	4.000	0.510	7.866	4.760	4.003	53.915

* Q = Proposition of cases such that $d_p > u_i$.
† Statistics computed only for those cases such that $d_p > u_i$.

TABLE A3
$p = 40$.

ρ_i	u_i	Q^*	d_p†			
			MEAN	SD	MIN	MAX
0.500	1.500	0.373	1.702	0.166	1.500	2.411
1.000	1.700	0.540	2.155	0.355	1.701	3.307
1.500	1.900	0.571	2.532	0.546	1.900	4.648
2.000	2.100	0.539	3.024	0.832	2.102	6.002
2.500	2.300	0.494	3.345	1.016	2.304	8.272
3.000	2.400	0.644	3.877	1.331	2.402	9.413
3.500	2.600	0.556	4.255	1.618	2.603	14.864
4.000	2.700	0.677	4.965	2.198	2.701	14.292
4.500	2.900	0.606	5.150	2.372	2.901	19.165
5.000	3.100	0.563	5.732	2.970	3.102	34.710

* Q = Proposition of cases such that $d_p > u_i$.
† Statistics computed only for those cases such that $d_p > u_i$.

Note. Davies' algorithm appeared in ALGOL. We had to translate it into FORTRAN for our Monte Carlo work.

Acknowledgment. The authors are thankful to the referees for some valuable suggestions which enhanced the presentation of the paper.

REFERENCES

C. K. BAYNE AND W. Y. TAN (1981), *QDF misclassification probabilities for known population parameters*, Comm. Statist. Theor. Math. A, 10(22), pp. 2315-2326.

W. R. CLARKE, P. A. LACHENBRUCH AND B. BROFFITT (1979), *How non-normality affects the quadratic discrimination function*, Comm. Statist. Theory Math. A, 8(13), pp. 1285-1301.

R. DAVIES (1980), *The distribution of a linear combination of χ^2 random variables*, Applied Statistics, 29, 3, pp. 323-333.

K. FUKUNAGA (1972), *Introduction to Statistical Pattern Recognition*, Academic Press, New York.

E. S. GILBERT (1969), *The effect of unequal variance-covariance matrices on Fisher's linear discriminant function*, Biometrics, 25, pp. 505-515.

E. HELLINGER (1909), *Neue bergrundung der theorie quadratisher formen von unendlichvielen veranderlichen*, J. Reine Angew Math., 136, pp. 210ff.

R. A. JOHNSON AND D. W. WICHERN (1982), *Applied Multivariate Statistical Analysis*, Prentice-Hall, Englewood Cliffs, NJ.

S. MARKS (1970), *Discriminant functions when covariance matrices are unequal*, Ph.D. dissertation, Dept. Public Health, Univ. California, Los Angeles, CA.

S. MARKS AND O. J. DUNN (1974), *Discriminant functions when covariance matrices are unequal*, J. Amer. Statist. Assoc., 69, 346, pp. 555-559.

P. B. PATNAIK (1949), *The non-central χ^2 and F-distributions and their approximations*, Biometrika, 36, pp. 202-232.

C. R. RAO (1977), *Cluster analysis applied to a study of race mixture in human populations*, Classification and Clustering, J. Van Ryzin, ed., Academic Press, New York, pp. 175-197.