

## GLOBAL ERROR ESTIMATES FOR ODEs BASED ON EXTRAPOLATION METHODS\*

L. F. SHAMPINE† AND L. S. BACA†

**Abstract.** It is shown how to exploit the powerful codes based on extrapolation of the midpoint rule so as to solve the initial value problem for a system of ordinary differential equations (ODEs) and to provide estimates of the global (true) error in the solution values. Particular attention is given to the reliability of the estimates and to the overall efficiency of the integration.

**Key words.** ODE, global error, true error, extrapolation, midpoint rule

**AMS(MOS) subject classifications.** 65L05

**CR categories.** 5.17

**1. Introduction.** Typical codes for the numerical integration of the initial value problem for a system of ordinary differential equations,

$$y' = f(x, y), \quad a \leq x \leq b, \quad y(a) \text{ given,}$$

step through the interval producing approximate solutions  $y_m \doteq y(x_m)$  on a mesh  $a = x_0 < x_1 < \dots < x_N = b$ . At each  $x_m$  the code adjusts the step size  $h = x_{m+1} - x_m$  so as to produce a solution  $y_{m+1}$  satisfying an accuracy requirement specified by the user of the code. A quantity called the local error is controlled at each step. Although natural from a technical point of view, local error is only indirectly related to the global, or true, error,  $y(x_{m+1}) - y_{m+1}$ , which is of most interest to the user. For this reason there has been quite a lot of research devoted to the question of estimating global errors. Excellent surveys by Prothero [4] and Stetter [11] provide an overview of this research.

There exist production-grade codes which solve nonstiff initial value problems and provide global error estimates. Although these codes are quite useful, there is a need for more reliable and more efficient procedures. We shall present an approach based on extrapolation of a modified midpoint rule. The approach could be based on any of the popular codes of this kind, but we shall take advantage of certain aspects of the effective code DIFEX1 of Deuffhard [2]. One of the features of our approach is the use of variation of order to achieve efficiency over a wide range of tolerances. Another is stringent testing of our basic hypothesis so as to result in a reliable estimate of the global error. This reliability holds rather well even at the extremes of crude and stringent tolerances which strain all the approaches to estimating global error (or integrating the equation, for that matter).

In the next section we describe how our approach has evolved from others and make general observations as to what quality estimate we might expect. In §3 we describe how to carry out the basic idea in the context of extrapolation and point out a novel solution to a surprising difficulty. Sections 4 and 5 are devoted to difficulties arising from stiffness and precision, respectively. A selection of illustrative examples follows. Besides illuminating facets of our approach, comparative results for several codes are presented. The paper concludes with a few final observations.

---

\* Received by the editors March 29, 1983, and in revised form July 25, 1983. This work was performed at Sandia National Laboratories supported by the U.S. Department of Energy under contract DE-AC04-76DP00789.

† Numerical Mathematics Division, Sandia National Laboratories, Albuquerque, New Mexico 87185.

**2. Some methods for global error estimation.** A number of methods for approximating the global error of a numerical solution of

$$(2.1) \quad y' = f(x, y), \quad a \leq x \leq b,$$

$$(2.2) \quad y(a) \text{ given,}$$

generate two approximate solutions  $\{y_n\}$ ,  $\{y_n^*\}$  in order to estimate the error of one. Let  $\{y_n^*\}$  be the more accurate solution. A time honored scheme is to generate  $\{y_n\}$  for a tolerance  $\tau$  and then to generate  $\{y_n^*\}$  for a tolerance  $r\tau$  with  $r < 1$ . If one assumes that the global error is approximately proportional to the local error tolerance, then

$$y(x_n) - y_n^* \doteq r(y(x_n) - y_n)$$

whence one can estimate the global error of the more accurate result  $y_n^*$  in terms of  $y_n$  and  $y_n^*$ . Tests by Enright [3] show that the better items of mathematical software do rather well at producing proportional error behavior, at least for routine problems. Shampine [7] has emphasized that this behavior is at most a secondary objective in the construction of the software and Stetter [12] has detailed the technical difficulties in achieving proportionality. In our considerable experience with this approach, the proportionality is not good enough to produce a reliable estimate of the more accurate result.

A much more reliable way to estimate global errors with two independent integrations is to estimate the global error of the less accurate integration by  $y_n^* - y_n$ . This requires only that  $y_n^*$  be more accurate than  $y_n$ . Seemingly this is almost certain to be so. The difficulty is that for technical reasons both results might pass the local error test at each step but that the local error of  $y_m^*$  is actually larger than that of  $y_m$  for a significant number of steps  $x_m$  and as a consequence  $y_n^*$  is less accurate than  $y_n$ . Nonetheless, this method, which we call reintegration, has important virtues and is reasonably reliable.

It is a virtue that the two integrations are completely independent, but unfortunately the results are not closely enough related to yield truly reliable global error estimates. A natural idea is to couple the integrations more closely. The classical development of the asymptotic behavior of the global error of a one-step method of order  $p$ , which for fixed step size  $h$  has the form

$$(2.3) \quad y(x_n) = y_n + h^p e(x_n) + O(h^{p+1}),$$

suggests estimation of  $h^p e(x_n)$  by two parallel integrations done with step sizes  $h$  and  $h/2$ . Shampine and Watts [10] have considered the many practical details involved in turning the idea into a production-grade code GERK. Here the two integrations, which are done simultaneously, are so closely coupled that one again estimates the global error of the more accurate result. GERK is an effective code which we shall use as a standard in our numerical examples.

There are two ways in which GERK could be improved. It has been found that variation of the order of formula is important to efficiency. The fixed order of GERK is a good choice for general use, but is too low for efficient integration at stringent accuracies. It is not clear how one could accommodate variation of order with global extrapolation. One of the virtues of reintegration is that it is applicable to variable order codes. We present in this paper a scheme built on the variable order (one-step) method resulting from extrapolation of the midpoint rule. Another difficulty with global extrapolation is its relatively restricted applicability. A lack of smoothness of  $f$  at one point  $x$  can destroy the relation (2.3) at later points. At crude tolerances and

in the presence of mild stiffness the leading term in (2.3) might not dominate sufficiently to get a reliable, much less accurate, estimate. At stringent tolerances the relation (2.3) is ruined by the effects of finite precision. Dekker and Verwer [1] have considered how to recognize when the global extrapolation is invalid and how accurate the estimate is. Some results from their code RGERK will be quoted later. This is a valuable development of the global extrapolation approach, but it does not address the question of a more broadly applicable method and is comparatively expensive.

To fully appreciate our approach, it is worth mentioning another technique described by Shampine and Watts. Global extrapolation pairs a single step from  $x_n$  to  $x_{n+1}$  to generate  $y_{n+1}$  with two half steps by the same method to generate  $y_{n+1}^*$ . It is supposed that using half the step size results in a more accurate solution  $y_{n+1}^*$ . An alternative is to generate  $y_{n+1}^*$  by a single step taken with a higher order method. Shampine and Watts call this “order extrapolation”. For this procedure one estimates the error of the less accurate result  $y_n$  because the exact relationship of the global errors of the two integrations is not known. The point we wish to make is that if the  $\{y_n\}$  represent the “primary” integration, a more accurate result  $\{y_n^*\}$  could be generated by using a smaller step size or a higher order. Our new approach in effect does both. Unfortunately it is not always the case that a more accurate result is obtained in this way. One of the key ideas in our attempt to get reliability is that we test the hypothesis that the secondary integration produce a result with smaller local error. By coupling the integrations and by doing this test, we avoid the principal source of unreliability with the method of reintegration.

We shall produce two sequences  $\{y_n\}$ ,  $\{y_n^*\}$  for which the estimated local error at each step in the construction of the secondary sequence  $\{y_n^*\}$  has a norm no greater than half that of the primary sequence  $\{y_n\}$ . This is a way to generate a  $\{y_n^*\}$  more reliably than the alternatives, but it is not *certain* to yield smaller errors. A little reflection about the nature of local error shows how it could happen that  $y_m^*$  have a larger global error than  $y_m$ . We shall give numerical examples.

The global error of  $y_n$  is to be estimated by comparison to  $y_n^*$ ,

$$ge_n = y(x_n) - y_n \doteq y_n^* - y_n.$$

Let us compare this estimate to the true global error:

$$\text{ratio} = \frac{y_n^* - y_n}{ge_n} = \frac{(y(x_n) - y_n) - (y(x_n) - y_n^*)}{ge_n} = \frac{ge_n - ge_n^*}{ge_n}.$$

Obviously

$$1 - \frac{|ge_n^*|}{|ge_n|} \leq \text{ratio} \leq 1 + \frac{|ge_n^*|}{|ge_n|}.$$

These bounds on the quality of the estimated global error depend, as might be expected, on  $y_n^*$  being more accurate than  $y_n$ . Notice that if  $\text{ratio} < 0$  or  $\text{ratio} > 2$ , then  $|ge_n^*| > |ge_n|$ . In our tests we describe such a value of ratio as “unacceptable” because the basic hypothesis that  $|ge_n^*| < |ge_n|$  is not true. This is despite the fact that a ratio  $> 2$  might be perfectly satisfactory for practical purposes. Considering that we ask only that the local error associated with  $y_n^*$  be half that associated with  $y_n$ , we really cannot expect any more than  $|ge_n^*| \leq 0.5 |ge_n|$ , in general. This implies  $0.5 \leq \text{ratio} \leq 1.5$ . Such a ratio will be described as “good”. It might well turn out that, say,  $|ge_n^*| \leq 0.9 |ge_n|$  which would imply  $0.1 \leq \text{ratio} \leq 1.9$ . We would describe this as “adequate”. Our construction tends to result in  $y_n^*$  with a local error much smaller than that of  $y_n$ . For this reason

$0.9 \leq \text{ratio} \leq 1.1$  is not unusual and can be fairly described as saying that we have an "excellent" estimate of the true global error. These simple observations give us an idea what to expect of the numerical realization of our approach to global error estimation.

**3. The practical process.** Our idea for a variable order code with global error estimates is developed for implementations of the extrapolated midpoint rule. The code used for our experiments is a modification of the April 30, 1981, version of the code DIFEX1 of Deuffhard. The idea is generally applicable to extrapolation codes, but certain aspects of Deuffhard's order and step size algorithm [2] will be exploited. It will be necessary first to review the basic integration scheme in order to explain how we proceed. Suppose one has an approximation  $y_0$  to the solution  $y(x_0)$  of (2.1), (2.2) and wishes to produce an approximation to  $y(x_0 + H)$ . An integer  $n_i$  is chosen, a step size  $h_i = H/n_i$  is defined, and a subintegration to  $x_0 + H$  is done:

$$\begin{aligned}\eta_0 &= y_0, \\ \eta_1 &= \eta_0 + h_i f(x_0, \eta_0), \\ \eta_{j+1} &= \eta_{j-1} + 2h_i f(x_0 + jh_i, \eta_j), \quad j = 1, \dots, n_i - 1.\end{aligned}$$

The smoothed value

$$S(h_i) = \frac{1}{2}[\eta_{n_i} + \eta_{n_i-1} + h_i f(x_0 + H, \eta_{n_i})]$$

is taken as the result of the subintegration. In polynomial extrapolation a tableau is now formed by the recipe

$$\begin{aligned}T_{i,1} &= S(h_i), \quad i = 1, 2, \dots, \\ T_{i,k} &= T_{i,k-1} + \frac{T_{i,k-1} - T_{i-1,k-1}}{(n_i/n_{i-k+1})^2 - 1}, \quad k = 2, \dots, i.\end{aligned}$$

This amounts to a convenient way of forming a family of Runge-Kutta formulas  $T_{i,k}$ . A standard result [2] is that the local error of  $T_{i,k}$  is asymptotically

$$(3.1) \quad e_{i,k} = \frac{1}{(n_{i-k+1} \cdots n_k)^2} \tau_k H^{2k+1} + O(H^{k+2}),$$

so that  $T_{i,k}$  is a formula of order  $2k$ . To form any element  $T_{i,k}$  in row  $i$  of the extrapolation tableau, subintegrations have to be made with  $h_1, h_2, \dots, h_i$ . The evaluation of  $f(x_0, \eta_0)$  can be saved and used in all the subintegrations. The cost, in terms of the number of evaluations of  $f$ , of forming  $T_{i,k}$  is  $A_i$  where

$$(3.2) \quad A_1 = n_1 + 1, \quad A_{j+1} = A_j + n_{j+1}, \quad j = 1, 2, \dots$$

The code DIFEX1 takes  $\{n_i\} = \{2, 4, 6, 8, 10, 12, 14, 16, 18, 20\}$ .

An important improvement due to Deuffhard (see [9] for just how much) is to take a subdiagonal element  $T_{k+1,k}$  as the one to be used from row  $k+1$  and to estimate its local error by

$$(3.3) \quad T_{k+1,k+1} - T_{k+1,k}.$$

If the local error of  $T_{k+1,k}$  is not satisfactory, one has the choice of raising the order by generating row  $k+2$  of the extrapolation tableau and considering  $T_{k+2,k+1}$  or of reducing  $H$  and trying again. As the error expression (3.1) makes clear, considering  $T_{k+2,k}$  corresponds essentially to reducing the step size, and considering  $T_{k+2,k+1}$

corresponds to reducing the step size and raising the order. It is not necessary for us to go into the various step size and order selection algorithms.

When the local error of  $T_{k+1,k}$  is acceptable, there is implicitly an assumption that  $T_{k+1,k+1}$  has a smaller local error so that the local error estimate (3.3) is valid. All the current codes actually advance the integration with the value  $T_{k+1,k+1}$  (local extrapolation). This value is presumably more accurate, but it is not known how much. We must deviate from standard practice here and advance the integration with  $T_{k+1,k}$ . It is essential in our approach that we have an *estimate* of the local error of the value *accepted* rather than a sort of bound on its local error.

Suppose that the primary integration is advanced from  $x_n$  to  $x_{n+1}$  and

$$y_{n+1} = T_{k+1,k}, \quad le \doteq T_{k+1,k+1} - T_{k+1,k} = \text{est}.$$

In the secondary integration we go to (at least) row  $k+2$  in the extrapolation tableau, thereby in effect resorting to a smaller step size. We also raise the order. We attempt to take

$$(3.4) \quad y_{n+1}^* = T_{k+2,k+1}^*.$$

A key point is that we approximate the local error in the secondary integration

$$le^* \doteq T_{k+2,k+2}^* - T_{k+2,k+1}^* = \text{est}^*$$

and insist that this error be smaller than that made in the primary integration. Specifically we require that

$$(3.5) \quad \|\text{est}^*\| \leq 0.5 \|\text{est}\|.$$

If all is going well, the secondary integration should be quite a lot more accurate. As pointed out, the code for the primary integration is attempting to adjust  $k$  and  $H$  so that  $T_{k+1,k+1}$  is more accurate than  $T_{k+1,k}$ . According to (3.1), the value  $T_{k+2,k+1}$  is asymptotically more accurate than  $T_{k+1,k+1}$  by a factor of  $(n_1/n_{k+2})^2$ , therefore ordinarily quite a lot more accurate. A local error expression like (3.1) holds for the secondary integration too. The only term depending on the problem is  $\tau_k$ . If the secondary integration is producing values close to those of the primary integration, i.e., the global errors of  $\{y_n\}$  are not “too” large, the factors should be similar. This means that the local error of  $T_{k+2,k+1}^*$  is then likely to be comparable to that of  $T_{k+2,k+1}$  hence quite a lot smaller than that of  $T_{k+1,k+1}$ , which is expected to be smaller than that of  $T_{k+1,k}$ . Furthermore, it is expected that the higher order result  $T_{k+2,k+2}^*$  will have a smaller local error than  $T_{k+2,k+1}^*$ . Putting this all together, it is *likely* that (3.4) yield a much more accurate solution locally than  $y_{n+1}$ . In any case, we *test* the hypothesis and insist that  $y_{n+1}^*$  be more accurate according to the standard local error estimates.

Suppose  $y_{n+1}^*$  is not accurate enough to pass the test (3.5). What can we do then? An obvious possibility is to raise the order still more by going to  $T_{k+3,k+2}^*$ . This is not very appealing because the work increases significantly on raising the order. Experiments brought to our attention a fact well-known, namely that the algorithms often result in a  $y_{n+1}$  which is much more accurate than required. Generally speaking this does no harm but here we may have difficulty in achieving still more accuracy in  $y_{n+1}^*$ . We were led to an action we have never seen before.

If (3.5) is not passed, we consider the possibility of changing  $y_{n+1}$  to make it *less* accurate. Specifically, we consider the use of the formula

$$(3.6) \quad y_{n+1}(\alpha) = \alpha T_{k+1,k} + (1-\alpha) T_{k+1,k+1} = T_{k+1,k+1} - \alpha \text{est}.$$

An estimate for its local error is

$$\|\text{est}(\alpha)\| = \|T_{k+1,k+1} - y_{n+1}(\alpha)\| = |\alpha| \|\text{est}\|.$$

The least that the local error of  $y_{n+1}$  can be increased and pass (3.5) is for  $\alpha$  such that

$$\|\text{est}^*\| = 0.5 \|\text{est}(\alpha)\|,$$

hence

$$(3.7) \quad \alpha = 2 \|\text{est}^*\| / \|\text{est}\|.$$

It is permissible to increase the local error of the primary integration as long as the local error test with the specified tolerance  $\tau$  can still be passed:

$$(3.8) \quad \|\text{est}(\alpha)\| = 2 \|\text{est}^*\| \leq \tau.$$

Thus we proceed as follows: The code generates  $y_{n+1} = T_{k+1,k}$  and in a parallel integration computes (3.4). If (3.5) is true, we go on to the next step. If it is not true, we test (3.8). When (3.8) holds, we ‘‘spoil’’ the accuracy of the step by accepting  $y_{n+1}(\alpha)$  from (3.6), (3.7) instead of  $y_{n+1}$ . This solution for the primary integration does pass the required test on the local error, and  $y_{n+1}^*$  is sufficiently more accurate that we can proceed on to the next step.

If  $y_{n+1}^*$  is not accurate enough, and if we cannot recover by decreasing the accuracy of  $y_{n+1}$ , we go to the expense of raising the order in the secondary integration and try

$$y_{n+1}^* = T_{k+3,k+2}^*.$$

If (3.5) then holds, we go on to the next step. If it does not, we try again to recover by decreasing the accuracy of  $y_{n+1}$ . If this is not possible, an error return is made from the code.

The primary and secondary integrations are coupled in our approach so that they produce results at the same points. The essentially new feature is that we manipulate the order and step size in the secondary integration and possibly even later the primary integration so as to verify that the estimated local error of the secondary integration is no more than half that of the primary integration. Error returns are made when this is not possible. In succeeding sections we shall study circumstances leading to such returns. Arguments already made suggest that in normal circumstances (3.4) not only suffices, it produces a  $y_{n+1}^*$  with a local error much smaller than that of  $y_{n+1}$ .

The bulk of the storage needed in an extrapolation code is to hold intermediate results of the extrapolation tableau. For this reason the maximum number of subintegrations is limited. In DIFEX1 this limit is 10. Because of our algorithm for global error estimation we limit the number of subintegrations in the primary integration to 8 and in the secondary integration to 10. The secondary integration is advanced a step only after the primary integration achieves a successful step. In this way no effort is wasted on an unsuccessful step in the primary integration. Furthermore, the secondary integration can use the same storage area for its extrapolation tableau and so avoid any significant increase in storage.

If the primary integration does  $k$  subintegrations and is successful, the step costs  $A_k$  function evaluations. The secondary integration normally does  $k + 1$  subintegrations at a cost of  $A_{k+1}$  evaluations. From this we can get an idea of the cost of the global error estimate. When a successful step of the basic code costs  $A_k$  evaluations, the code with global error estimate will cost normally  $A_k + A_{k+1}$  evaluations. There must be at least  $k = 2$  subintegrations. With the restriction  $k \leq 8$  and the sequence  $\{n_i\}$  used in DIFEX1 this cost ranges from a factor of 2.86 larger at  $k = 2$  down to 2.25 at  $k = 8$ .

Of course the handling of failed steps in the primary integration and the possibility of having to raise the order in the secondary integration means that these factors are a very rough guide. Still, the cost is comparable to that of other procedures for estimating global errors.

Advantage can be taken of Deuffhard's order and step size selection algorithm [2]. Unlike, say, Adams codes, the cost of a step depends on the order used and the sequence  $\{n_i\}$  implemented. Deuffhard properly accounts for the cost in the selection algorithm in terms of costs normally set according to the recipe (3.2). If we realize that our scheme normally does  $A_k + A_{k+1}$  evaluations instead of  $A_k$  in carrying out  $k$  subintegrations, a simple alteration of the setting of the costs in DIFEX1 allows the code to take into account the secondary integration when selecting the most efficient step size and order. This is quite a nice aspect of Deuffhard's algorithm from our point of view.

It is not our intention to offer a replacement for the codes, such as DIFEX1, being used to solve ODEs routinely, rather to offer a capability for the occasional assessment of true errors or for the solution of problems with special requirements of reliability. Our code will ordinarily be significantly more expensive than DIFEX1 for two reasons: It does more work at each step because of the secondary integration. It produces a less accurate result because it does not perform local extrapolation and may produce a still less accurate result as a by-product of the process for getting a reliable global error estimate. We have given an indication of the difference in cost arising from the first consideration and then immediately blurred the picture by reducing this cost through manipulation of the order chosen. It is not at all clear what the net difference in cost of the original DIFEX1 code and our modification would be, but this is not important in our use of global error estimation. The important question, which we will address, is how the effectiveness of our code compares to that of other codes which estimate the true error.

**4. Stiffness.** Shampine and Watts [10] have pointed out certain practical and theoretical difficulties associated with global extrapolation in the presence of stiffness. This is to be expected because the postulated asymptotic behavior is disturbed. Perhaps surprisingly, there are analogous difficulties with our approach.

A practical matter requiring some care with global extrapolation is that the step size is selected in one integration and a multiple of it is simply used in the other integration. One must be sure that if the step size is selected so as to keep the one integration stable, the other integration will also be stable. The same is true with our scheme. By computation of the regions of absolute stability, we have verified that if the step size chosen in the primary integration to advance with the formula  $T_{k+1,k}$  is stable, then it is also stable with the formulas  $T_{k+2,k+1}$  and  $T_{k+3,k+2}$  used in the secondary integration. This fact is not surprising in view of the smaller step size(s) used in the additional subintegration(s) to raise the order. Because of the rather uniform improvement, there is little point in our reproducing the plots here. We comment that this issue was brought to our attention by the observation of instability in a cheaper implementation of our basic idea that we tried out first.

It has been argued [6] that in the presence of stiffness, reasonable step size selection algorithms for one-step methods will lead to step sizes which correspond to being on the boundary of the stability region, on the average. This is what disturbs the expected asymptotic behavior of global extrapolation. However, it also disturbs our scheme in a perhaps unexpected way. We obtain reliable estimates of the global error by comparing the size of the estimated local errors in two parallel integrations. The thing that

causes the step size to fluctuate about the value corresponding to the boundary of the stability region is the fact that the local error estimates become large outside the region. This may be due to a large local error, but it may also be due to an inaccurate estimate. If the local error estimates are not reliable, neither is our global error estimate.

As with global extrapolation, if the step size is artificially restrained by a specified maximum step size, or output, so as to correspond to being well inside the region of absolute stability, the global error estimate will be reliable. In our case this is because the local error estimate is then reliable.

**5. Precision difficulties.** In this section we take up several issues loosely connected by their relation to the accuracy tolerance. At crude tolerances the integral curves followed by the primary and secondary integrations might be quite different and so present differing degrees of difficulty for their integration over a step. It can happen that the secondary integration is unable to achieve the smaller local error required even when all the devices described in § 3 are brought to bear on the task. An error return is made. This was unexpected for us, but we have seen a number of examples and now do not think it rare. A suitable action is to reduce the tolerance and try again to complete the integration.

At the opposite end of the tolerance range, we expect that at very stringent tolerances it will be difficult to produce  $y_n^*$  with a smaller local error than  $y_n$  and near machine unit roundoff, impossible. In addition, the estimates of the local error are so disturbed by roundoff near limiting precision that they cannot be believed and the global error estimate is unreliable. This is the analogue of the asymptotic error behavior being disturbed for global extrapolation. In [5] a couple of very simple devices are presented which recognize when the tolerances approach limiting precision. They have proven very useful and were taken into account in GERK. DIFEX1 has one device but not the other. We modified the code to add it. DIFEX1 implements an error control which is (almost) a pure relative error test. We modified it as described in [8] to get a sound pure relative error test, which is convenient in the present context but not necessary. The device added then is very simple. The code accepts an input tolerance  $\varepsilon$  and internally requires a norm of the estimated local error to be less than or equal to  $\varepsilon/4$ . This will be required of  $y_{n+1}$ . We ask in the secondary integration that  $y_{n+1}^*$  have a local error no more than half as big, hence certainly no more than  $\varepsilon/8$ . However, to estimate the local error of  $y_{n+1}^*$ , a still more accurate value  $y_{n+1}^{**}$  is formed. The fundamental limit is that this last value cannot be more accurate than the correctly rounded solution, that is, its local error is at least as great as a unit roundoff  $u$  in the true solution. It is more realistic to say that we cannot ask for a local error less than several units of roundoff in the true solution. Tracing our way back we see that the relative error tolerance  $\varepsilon$  must be a fairly large multiple of the unit roundoff  $u$  on the machine used if the results are to be meaningful. We therefore insist that the relative error tolerance  $\varepsilon \geq 100u$  and return an error message if it is not. This is simple and obvious when a pure relative error test is used. To compare with other codes we also altered DIFEX1 to provide it with a pure absolute error test. The complication this causes is that the precision requirement must be tested at every stage. When the code has a tentative solution, the absolute error test is converted to an equivalent relative error test and it is asked if the relative error to be imposed is less than  $100u$ .

Even when the code is working with a pure absolute error control, the device described has a negligible cost. In experiments without it, we found that we might well get a failure return because the secondary integration could not be made accurate enough. However, we often got very poor global error estimates without any sign of



difficulty because the local error estimates were disturbed. Adding the device greatly enhanced the reliability at stringent tolerances.

There is a difficulty in interpretation of global error estimates which must be understood to assess properly the performance of methods. It is reasonable to consider the ratio of the estimated global error to the true global error, as we proposed in § 2, at points where the global error is comparatively large. It is not, however, reasonable to do this near points where the global error changes sign. Perhaps the matter is clearest with the method of global extrapolation which uses

$$y(x_n) = y_n + h^p e(x_n) + O(h^{p+1})$$

and approximates

$$y(x_n) - y_n \doteq h^p e(x_n).$$

Obviously this approximation breaks down where  $e(x)$  vanishes. Near a change of sign of the global error it is clear that we must expect, in general, that  $e(x)$  not always have even the correct sign. Less dramatic are the large relative (but not absolute) errors in the estimation of the global error which will be observed. These facts make it difficult to present simple but meaningful statistics at all mesh points. They also pose a practical difficulty. When should the user ignore the global error estimate as being potentially of the wrong sign or wrong order of magnitude? We can proceed much as with the local error estimates. For the global error estimate to be any good,  $y_n^*$  must be more accurate than  $y_n$ . Obviously we cannot expect it to be correct to more than a few units of roundoff and correspondingly  $y_n$  must be still less accurate. This says that an estimated global error in  $y_n$  less than a modest number of units of roundoff in  $y_n$  cannot be trusted. In our tests we disregard estimates less than  $100u|y_n|$  to provide a measure of protection. This could be implemented in the form of a warning to the user, but in our research code we have left this issue to the user who, after all, has all the data needed to test the values. Of course, if one were to focus merely on the absolute error of the global error estimate there would be no difficulty of this kind. This is not a very satisfactory remedy since for both intrinsic and technical reasons, the true global error can range over several orders of magnitude in a normal integration.

**6. Numerical examples.** A number of authors have considered the problem

$$(6.1) \quad \begin{aligned} y_1' &= \frac{1}{2} \frac{y_1}{(x+1)} - 2xy_2, & y_1(0) &= 1, \\ y_2' &= \frac{1}{2} \frac{y_2}{(x+1)} + 2xy_1, & y_2(0) &= 0, \end{aligned}$$

on various intervals. The solution components are

$$y_1(x) = \sqrt{x+1} \cos(x^2), \quad y_2(x) = \sqrt{x+1} \sin(x^2).$$

As  $x$  increases, this problem becomes increasingly difficult to solve because the components oscillate with increasing frequency. Prothero [4] has plotted the true global errors and their estimates for the first component when the equation is integrated on  $[0, 3]$  with a pure absolute error test using two codes. One code is the GERK code based on global extrapolation. In 1971 Stetter proposed a way of estimating global errors by the use of formulas having exact principal error equations. This approach is the basis of a code written by Merluzzi and Brosilow which was also tested by Prothero. The two plots give some feeling as to the issue of the quality of the estimate near a change of sign of the true error. It is difficult to quantify the quality of the estimates.

In a general way, the plots show the estimates to be surprisingly good even though their relative error is large in places. Plotting is very attractive, but scaling plots appropriately is difficult. Dekker and Verwer [1] have compared GERK and RGERK on (6.1) with the interval  $[0, 8]$ . They measure the ratio of the estimated to the true global error.

We wanted a fairly difficult example, so we solved (6.1) on  $[0, 10]$ . This was done for a wide range of tolerances to study possible variation of reliability and the effect on the cost of our variable order code. The same computations were done with GERK so as to contrast our results with a fixed order code. As described in § 2, we measured the ratio of the estimated global error to the true global error except when the estimate is less in magnitude than 100 units of roundoff in the solution itself (for reasons presented in § 5). For reasons given there we describe a value of ratio such that

ratio  $< 0$  or ratio  $> 2$  as unacceptable—*U*,

$0.1 \leq \text{ratio} \leq 1.9$  adequate—*A*,

$0.5 \leq \text{ratio} \leq 1.5$  good—*G*,

$0.9 \leq \text{ratio} \leq 1.1$  excellent—*E*.

Recall that a ratio  $> 2$  might be acceptable for practical use, but we describe it as “unacceptable” for the technical reason that the estimate is not justified by our usual argument. This may not be a reasonable label for the results of the GERK code. Table 1 gives the percentage of steps which gave an estimate of the specified quality for the first solution component as a function of the pure absolute error tolerance “tol”. Table 2 gives the same quantity for the second solution component. Notice that we have not rated  $0 \leq \text{ratio} < 0.1$ , which does occur at tolerance  $10^{-4}$  with component 2, nor  $1.9 < \text{ratio} \leq 2$ , which does occur at tolerance  $10^{-11}$  with component 2.

TABLE 1  
Quality of estimates of solution of first component of (6.1) on  $[0, 10]$ .

$-\log_{10} \text{tol}$	extrapolation					GERK				
	U	A	G	E	FCN	U	A	G	E	FCN
1	1	100	98	85	2,202	53	44	27	8	1,088
2	1	100	99	97	4,248	19	63	48	10	1,674
3	0	100	100	100	4,388	11	78	63	16	2,551
4	0	100	100	100	4,546	5	86	73	26	4,019
5	0	100	98	96	5,340	4	91	83	39	6,326
6	0	100	100	96	6,198	2	85	81	48	10,011
7	2	98	98	93	7,414	1	67	65	47	15,856
8	0	100	100	98	8,614	1	58	56	41	25,111
9	0	100	100	99	10,196	7	43	35	15	39,619
10	0	100	97	88	12,072	23	21	10	3	60,013
11	8	91	81	44	14,308	27	10	3	0	77,887
12	quit—tolerance too small					not attempted				

We think the results for our approach are extraordinarily good. The high percentage of steps with “excellent” error estimates means that the local errors made in the secondary integration are significantly smaller than those in the primary integration, as we thought likely. The computations were all performed on a computer with about 14 decimal digits precision. The quality of the estimates is disturbed at the absolute

TABLE 2  
Results corresponding to Table 1 for the second solution component.

extrapolation						GERK		
$-\log_{10} \text{tol}$	U	A	G	E	U	A	G	E
1	1	99	99	85	49	47	27	3
2	0	100	100	98	30	67	49	14
3	0	100	99	98	20	79	64	20
4	0	98	98	94	14	86	76	27
5	0	100	100	96	8	90	83	42
6	0	100	100	96	4	85	81	49
7	0	100	100	93	3	67	65	47
8	0	100	100	100	2	59	57	41
9	0	100	100	100	12	43	35	15
10	0	100	96	88	26	21	10	4
11	10	89	81	44	28	10	3	1
12*								

error tolerance  $10^{-11}$  but we were successful at avoiding the poor estimates which would have been seen at  $10^{-12}$ . The quality of the estimates is notably insensitive to the tolerance. In contrast, the quality of the estimates provided by GERK is seriously disturbed at both crude and stringent tolerances. It is never of quality comparable to the new code.

Its authors emphasize that the fixed order code GERK will not be efficient at stringent tolerances. The costs, measured by the number of function evaluations FCN, displayed in Table 1 illustrate this. The higher orders allowed by our code are of great advantage at stringent tolerances. On the other hand, GERK is significantly cheaper at crude tolerances. In part this is because it does not strive so hard for reliability, but in part it is intrinsic to the approaches. We consider the reliability worth the extra cost since, after all, this is the range of tolerances where the total cost is least.

The problem

$$y' = -32xy \ln 2, \quad y(-1) = 2^{-10}, \quad -1 \leq x \leq 1,$$

has the peaked solution

$$y(x) = 2^{6-16x^2}.$$

For  $x < 0$  the problem is unstable so that the global error grows rapidly from the initial point to  $x = 0$ . For  $x > 0$  the problem becomes increasingly stable and the global error decays rapidly. Dekker and Verwer [1] present some results for GERK and RGERK for integration with the pure relative error tolerance  $10^{-4}$ . They display in their Table 5 the values of “ratio” at only 8 points and state that the results are similar at other points. By our characterization of the quality, GERK gave good results at all the points reported and excellent results at 3 of the 8 points. The integration cost 810 function evaluations. RGERK gave excellent results at all 8 points reported at a cost of 1584 evaluations. Our code solved this problem in 15 steps. The results were good at all steps and excellent at 14 of the 15 steps. The cost was 986 evaluations.

The problem

$$y' = 10(y - x^2), \quad y(0) = 0.02, \quad 0 \leq x \leq 2,$$

TABLE 3  
*Solution of mathematically unstable problem.*

GERK			RGERK		extrapolation		
$-\log_{10} \text{tol}$	ratio	FCN	ratio	FCN	ratio	max/min	FCN
1	.11	91	.77	181	.95	1.0/.95	332
2	.38	150	.96	294	.94	1.0/.94	482
3	.68	314	1.00	602	.97	1.0/.97	554
4	.83	517	1.00	1,003	.99	1.0/.99	580
5	.90	771	1.00	1,491	.98	1.0/.98	764
6	.94	1,021	1.00	2,011	.99	1.0/.99	850
7	.96	1,348	1.00	2,680	.99	1.0/.99	978
8	.97	2,050	1.00	4,084	1.0	1.0/1.0	1,176
9	.98	3,228	.95	6,450	.99	1.0/.97	1,306
10	.93	5,136	.49	10,266	.99	1.0/.97	1,634
11	.80	6,522	.14	13,038	1.0	1.0/.97	1,798

is mathematically unstable as the general solution of the differential equation

$$y(x) = 0.02 + 0.2x + x^2 + c e^{10x}$$

makes clear. It has been used as a test problem by a number of authors. In Table 3 we present results for GERK, RGERK, and our extrapolation code. The results for GERK and RGERK are taken from [1]. They were obtained on the same kind of computer as we used for our code; indeed, we verified the results for GERK. The ratio of the estimated to true global error is computed only at the endpoint  $x = 2$ . To provide a better assessment of the quality of the estimate, we report the maximum and minimum over all the steps of the integration for this ratio with our code. GERK does commendably well. RGERK was intended to provide a better, though more expensive, estimate and this is observed to be the case here. The costs of the integrations cannot be taken at face value. All the codes are "tuned" differently so that, for example, the solution by RGERK is roughly an order of magnitude more accurate than that of GERK. Our intention is to show the general size of the costs, especially the behavior as a function of the tolerance. For most practical purposes all the codes produce adequate estimates of the global error at  $x = 2$ . The most interesting point about the quality is that our approach produced significantly better estimates at both extremes of the range of tolerances. The results for the maximum and minimum value of ratio show that our approach is uniformly good. The dependence of the cost in function evaluations, FCN, on the tolerance is illuminating. Our variable order code has a cost much less sensitive to the tolerance and because it can resort to much higher orders, the cost is much lower at stringent tolerances. Considering the cost of the lowest order procedure in the variable order code, it is not surprising that RGERK, and especially GERK, are cheaper at crude tolerances. However, this difference in cost is partly due to the additional attention given to reliability.

Dekker and Verwer [1, p. 14ff] present some results of RGERK applied to the mildly stiff problem

$$(6.2) \quad y' = -100 \left( y - \frac{x}{x+1} \right) + \frac{x}{(x+1)^2}, \quad y(0) = 0.$$

The general solution is

$$y(x) = \frac{x}{x+1} + y(0) e^{-100x}$$

so that the initial condition has been chosen so as to eliminate the initial transient. They considered the integration of this problem with a pure absolute error tolerance of  $10^{-3}$ . We also did this on  $[0, 2]$  for several initial step sizes. With the initial step size  $10^{-1}$ ,  $10^{-2}$ ,  $10^{-3}$ , our code successfully completed 0, 2, 1 steps, respectively, before returning with the message that it was unable to achieve sufficient accuracy in the secondary integration. This is not surprising. With the special initial condition, the character of integral curves close to  $y(x)$  differ markedly from  $y(x)$  itself. An unfortunate aspect of the problem specification is the pure absolute error criterion. In view of the fact that  $y(0) = 0$ , the accuracy control is very lax at  $x = 0$  which causes codes difficulty in adjusting a guessed initial step size to an appropriate value. We also solved this problem with the step size limited to 0.02, which assures absolute stability. Our initial step size was 0.01. The final step size was also less than 0.02 in order to hit  $x = 2$ . The other 99 steps were of maximum size. The ratio of the estimated to true global error at the first step was .97, excellent, and at the last step .81, which we call good. The ratio was remarkably constant on the other 99 steps since it ranged from .68 to only .70, which represents a good quality estimate.

We also solved the equation of (6.2) with the initial condition changed to  $y(0) = 1$ . With initial step size  $10^{-2}$ , the first part of the integration is then not particularly stiff because of the initial transient. As a consequence the code reached  $x = 1.64$  in 45 steps before returning with a message that the secondary integration was insufficiently accurate. The ratio of estimated to true global error had a quality characterized as follows: 33% of the values of ratio were excellent, 84% were good, 98% were adequate, and only one estimate (2%) was unacceptable.

**7. Conclusions.** The simple theoretical foundations of our approach to global error estimation promise a reliable estimate at a cost relatively insensitive to the accuracy desired. The numerical results of the last section confirm this promise. The performance of our research code is gratifyingly good in comparison with codes based on other approaches.

Although our approach to global error estimation could be grafted on to any code based on extrapolation, we have taken significant advantage of certain aspects of Deuffhard's step and order selection algorithms [2]. As a consequence, the order selected in the modified code is not necessarily the same as that selected in the original code DIFEX1. In addition, we had to take out the local extrapolation which is standard for such codes with the consequence that the result of a step is not the same in modified and original code even if both choose the same order. The importance of these observations is that often users want to make only spot checks on the global error. They would prefer to turn off this feature for production runs if so doing would significantly reduce the cost. Although our code is an efficient one, it is significantly more expensive than the original DIFEX1. It would be nice if both codes gave the same results so that one could use the cheaper code when global error estimates are not required. Unfortunately this goal was not realized. We point out that essentially the same can be said of the global extrapolation code GERK which also does not produce the same results as its "original" Runge-Kutta code RKF45.

## REFERENCES

- [1] K. DEKKER AND J. G. VERWER, *Estimating the global error of Runge–Kutta approximations*, Rept. NW 130/82, Math. Centrum, Amsterdam, 1982.
- [2] P. DEUFLHARD, *Order and stepsize control in extrapolation methods*, Numer. Math., to appear.
- [3] W. H. ENRIGHT, *Using a testing package for the automatic assessment of numerical methods for O.D.E.'s*, in Performance Evaluation of Numerical Software, L. D. Fosdick, ed., North-Holland, Amsterdam, 1979, pp. 199–213.
- [4] A. PROTHERO, *Estimating the accuracy of numerical solutions to ordinary differential equations*, in Computational Techniques for Ordinary Differential Equations, I. Gladwell and D. K. Sayers, eds., Academic Press, London, 1980, pp. 103–128.
- [5] L. F. SHAMPINE, *Limiting precision in differential equation solvers*, Math. Comp., 28 (1974), pp. 141–144.
- [6] ———, *Stiffness and nonstiff differential equation solvers, II: detecting stiffness with Runge–Kutta methods*, ACM Trans. Math. Software, 3 (1977), pp. 44–53.
- [7] ———, *Discussion of session on performance evaluation in ordinary differential equations*, in Performance Evaluation of Numerical Software, L. D. Fosdick, ed., North-Holland, Amsterdam, 1979, pp. 215–217.
- [8] ———, *Robust relative error control*, Rept. SAND82-2320, Sandia National Laboratories, Albuquerque, NM, 1982.
- [9] ———, *Efficient extrapolation methods for ODEs*, Rept. SAND83-0041, Sandia National Laboratories, Albuquerque, NM, 1983.
- [10] L. F. SHAMPINE AND H. A. WATTS, *Global error estimation for ordinary differential equations*, ACM Trans. Math. Software, 2 (1976), pp. 172–186.
- [11] H. J. STETTER, *Global error estimation in ODE-solvers*, in Lecture Notes in Mathematics 630, Springer, Berlin, 1978, pp. 179–189.
- [12] ———, *Tolerance proportionality in ODE-codes*, in Seminarberichte Nr. 32 Sektion Math., R. März, ed., Humboldt Univ., Berlin, 1980, pp. 109–123.

## THE TUNNELING ALGORITHM FOR THE GLOBAL MINIMIZATION OF FUNCTIONS\*

A. V. LEVY† AND A. MONTALVO‡

**Abstract.** This paper considers the problem of finding the global minima of a function  $f(x): \Omega \subset \mathcal{R}^n \rightarrow \mathcal{R}$ . For this purpose we present an algorithm composed of a sequence of cycles, each cycle consisting of two phases: (a) a minimization phase having the purpose of lowering the current function value until a local minimizer is found and, (b) a tunneling phase that has the purpose of finding a point  $x \in \Omega$ , other than the last minimizer found, such that when employed as starting point for the next minimization phase, the new stationary point will have a function value no greater than the previous minimum found.

In order to test the algorithm, several numerical examples are presented. The functions considered are such that the number of relative minima varies between a few and several thousand; in all cases, the algorithm presented here was able to find the global minimizer(s). When compared with alternate procedures, the results show that the new algorithm converges more often to the global minimizer(s) than its competitors; additionally, it becomes more efficient than the other procedures for problems with increasing density of relative minima.

**Key words.** global optimization, nonlinear programming, unconstrained minimization

**1. Introduction.** One of the most interesting research areas in nonlinear programming is that of finding the global minimizer of a function defined in an  $n$ -dimensional linear space. Until recent years this problem has been almost completely ignored and only few numerical methods have been developed to solve it. Among these, three approaches have been used: (a) In the hill climbing approach, all the extremal points are found sequentially in a deterministic fashion and comparing the values of  $f(x)$  at *all* the local minimizers, the global is identified (Brannin [3]; Hardy [6]). (b) In the multiple random start approach, a very large number of starting points are given and any minimization algorithm is used to find the corresponding local minimizer. By comparing the values of  $f(x)$  at *all* the local minimizers the global is identified (Anderssen [1]). (c) In the function modification approach, an auxiliary function is defined as whose function value at the local minimizer is lower than the value of the function at all the local minimizers previously found. Using this algorithm recursively, the global could be approached (Goldstein [5]).

Most of the algorithms so far developed along these lines, are practical only for low dimensionality problems. In this paper we present a new algorithm which retains the best properties of each approach, while avoiding their main disadvantages, such as unpredictable performance while approaching the global minimizer(s), large computing time and the evaluation of higher order derivatives. The new method (Tunneling Algorithm) consists of two phases, a minimization phase and a tunneling phase. These phases are used sequentially to approach the global minimizer of  $f(x)$ . In the minimization phase, for a given starting point  $x^0$ , any minimization algorithm with a local descent property on  $f(x)$  can be used to find a local minimum of  $f(x)$ , say at  $x^*$ .

In the tunneling phase, an auxiliary function  $T(x)$ , is defined, where  $T$ , the tunneling function, is a scalar function with continuous first derivatives, whose zero-set

---

\* Received by the editors March 15, 1982, and in revised form August 23, 1983. This work was partially supported by the National Science Foundation Grant NSF-MCS-76-21657.

† Postdoctoral Fellow in Aero-Astronautics, Rice University, Houston, Texas, Professor of IIMAS, Universidad Nacional Autónoma de Mexico, Mexico City, DF, Mexico.

‡ Research Associate, IIMAS, Universidad Nacional Autónoma de Mexico, Apdo. Postal 20-726, Deleg. A. Obregón, 01000 Mexico, DF, Mexico.

coincides with the set where  $f(x) = f(x^*)$  and which depends on a set of parameters that are chosen automatically by the algorithm with the purpose of stabilizing the method. The objective of this phase is the following: starting at any point in a neighborhood of  $x^*$ , we seek a new point  $x^0$  such that  $T(x^0) \leq 0$ .

By using these two phases alternately, the sequence of minimizers found are such that the function value at each of these minimizers is never greater than any of the function values at the previously found minimizers; that is, the function value at the local minimizers is nonincreasing.

Numerical results are presented for sixteen examples which are solved by the present method and by two versions of the multiple random start approach. These examples vary from two to ten variables, from one strict global minimum to eighteen global minima and up to several thousands of stationary points.

The numerical results show that the present method is usually faster than the other methods and more important, it converges more often to the global minimizer than the other methods. In particular, the numerical results indicate that these relative advantages increase with the density of relative minima.

**2. Derivation of the algorithm.** Let  $f(x)$  be a twice continuously differentiable function on the set  $\Omega = \{x \in \mathbb{R}^n : a \leq x \leq b\}$ , with  $a$  and  $b \in \mathbb{R}^n$ . We will assume that all the minima of  $f(x)$  are isolated minima and that there is a finite number of them.

In this paper we consider the problem of finding all the global minimizers of  $f(x)$  in  $\Omega$ . For solving this problem, we will present an algorithm that consists of two separate phases. The objective of the first phase is to find a local minimizer  $x^*$  of  $f(x)$ ; the second phase is designed to move from this point to a point  $x^0 \neq x^*$  with  $f(x^0) \leq f(x^*)$ . These two phases are described next.

**2.1. Minimization phase.** Given a starting point  $x^0$  we use any minimization algorithm to find a local minimizer of  $f(x)$ , say  $x^*$ . This algorithm can be any unconstrained minimization method with descent property, e.g., steepest descent, conjugate gradient, Newton's method. We will assume that at the end of this phase a local minimizer has been found.

**2.2. Tunneling phase.** We start this phase at  $x^*$ , the exit point of the minimization phase, and its purpose is to find a point  $x^0 \in \Omega$  such that

$$(1) \quad \begin{aligned} f(x^0) &\leq f(x^*), \\ x^0 &\neq x^*. \end{aligned}$$

This can be formally stated as follows: find  $x^0 \in Z = \{x \in \Omega - \{x^*\} : f(x) \leq f(x^*)\}$ . We now have the following dichotomy, whenever  $Z$  is not empty:

- i)  $x^0$  is also a local minimizer of  $f(x)$  and  $x^0 \neq x^*$ ;
- ii) if  $x^0$  is not a minimizer of  $f(x)$ ; then a further descent on the function value can be achieved, e.g., by moving along the direction given by  $-f_x(x^0)$ .

In the former case we have located a new minimizer of  $f(x)$  where the function value is *not higher* than the previous one,  $f(x^*)$ ; in the latter case we have found a point that can be used to start a new minimization phase that will locate a new minimizer with a function value lower than  $f(x^*)$ . (In the case when  $Z = \emptyset$ , then the algorithm goes to the boundary of  $\Omega$ .)

In order to move from  $x^*$ , consider the function

$$(2) \quad T(x) = \frac{f(x) - f(x^*)}{[(x - x^*)'(x - x^*)]^\eta}$$



(where ' denotes transposition) which has a pole at  $x^*$  for  $\eta$  sufficiently large. Note that all  $x^0 \neq x^*$  satisfying  $T(x^0) \leq 0$  are contained in  $Z$ . Therefore, we will consider the problem of finding a nonpositive minimum of  $T(x)$ . In practice, after several applications of the two phases the function  $T(x)$  will have the general form (Appendix I)

$$(3) \quad T(x) = \frac{f(x) - f^*}{\{\prod_{i=1}^l [(x - x_i^*)'(x - x_i^*)]^{\eta_i}\} [(x - x_m)'(x - x_m)]^{\lambda_0}}$$

The purpose of each of the terms in (3) is the following: The difference in the numerator eliminates as possible solutions of this phase all those points  $x$  satisfying  $f(x) > f^*$ . The first term in the denominator prevents the algorithm to locate as solutions of this phase all previous minimizers found at  $x_i^*$ ,  $i = 1, 2, \dots, l$ , with a function value  $f(x_1^*) = f(x_2^*) = \dots = f(x_l^*) = f^*$ ; that is, the algorithm will not cycle between previous solutions with the same function value (*Remark.*  $l$  is always taken as one whenever the new minimizer found produces a function value strictly lower than the previous one, and is increased by one if the new minimizer produces a function value equal to the previous one). The second term in the denominator of (3) is designed to smooth out, in an adaptive fashion, any irrelevant local minimizer of  $T(x)$  that might attract any particular minimization algorithm during the search for  $x^0$  (points where  $T_x(x) = 0$  and  $T(x) > 0$ ).

Finally, the tunneling phase can be actually implemented by using any minimization algorithm with descent property on  $T(x)$ . The rules needed to determine the correct value of all the parameters involved in  $T(x)$  are presented in Appendix I.

**2.3. Minimization algorithms.** Next, we present three different algorithms employed in the minimization phase; namely, ordinary gradient, conjugate gradient and Newton's methods as well as a modified version of Newton's method as employed in the tunneling phase.

**2.3.1. Ordinary gradient** (Himmelblau [7]). For this method, the displacements  $\Delta x$  are generated according to

$$\Delta x = -\alpha f_x(x)$$

where  $\alpha$ , the step size, is chosen such that

$$f(x + \Delta x) < f(x)$$

by using a bisection procedure on  $\alpha$ , starting at  $\alpha = 1$  and up to  $N_b$  bisections.

**2.3.2. Conjugate gradient.** This method was implemented following the Fletcher-Reeves method (Himmelblau [7]) as follows:

- a. Set  $k = 0$  and  $\Delta x^0 = -f_x(x^0)$  as the first search direction;
- b. find  $\alpha$ , the step size, such that

$$\frac{df(x^k + \alpha \Delta x^k)}{d\alpha} = 0;$$

- c.  $x^{k+1} = x^k + \alpha \Delta x^k$ ;
- d. The new search direction is evaluated according to

$$\Delta x^{k+1} = -f_x(x^{k+1}) + \frac{f'_x(x^{k+1})f_x(x^{k+1})}{f'_x(x^k)f_x(x^k)} \Delta x^k;$$

- e.  $k = k + 1$  and repeat the procedure from step b. Whenever  $k = n + 1$ , the entire procedure is restarted at step a, by taking  $x^0 = x^k$ .

Finally, to accomplish step b, the minimum of  $f(x)$  along the search direction  $\Delta x^k$ , is first bracketed; once this is done, the optimum value of  $\alpha$  is found by using quadratic interpolation.

**2.3.3. Newton's method** (Himmelblau [7]). This method generates displacements  $\Delta x$  according to

$$\Delta x = -\alpha \rho f_{xx}^{-1}(x) f_x(x)$$

where  $\alpha$ , the step size is determined as in the ordinary gradient method and  $\rho$ , a direction factor, is defined according to

$$\rho = \begin{cases} 1 & \text{if } f(x - 2^{-N} \rho f_{xx}^{-1}(x) f_x(x)) < f(x), \\ -1 & \text{if } f(x - 2^{-N} \rho f_{xx}^{-1}(x) f_x(x)) > f(x). \end{cases}$$

**2.3.4. Modified Newton's method** (Miele [8]). The displacements  $\Delta x$ , in the tunneling phase are generated according to

$$\Delta x = -\alpha \frac{T(x)}{T'_x(x) T_x(x)} T_x(x)$$

and  $\alpha$ , the step size, is determined as in the ordinary gradient and Newton's methods.

In all four methods described above, provisions must be taken in order to prevent that any point generated by the procedures lies outside the admissible region.

**3. Some properties of the tunneling algorithm.** Using the material presented in the previous section, we now exhibit the following descent property of the tunneling algorithm obtained by successive applications of minimization and tunneling phases.

a) If any of the minimization phases uses  $x_i^0$  as a starting point, a local minimizer of  $f(x)$  will be found, say  $x_i^*$ , whose function values satisfy

$$(4) \quad f(x_i^*) \leq f(x_i^0).$$

b) Similarly, if any of the tunneling phases uses  $x_i^*$  as starting point, the solution point of the corresponding tunneling function  $T(x)$ , say  $x_{i+1}^0$ , satisfies

$$(5) \quad f(x_{i+1}^0) \leq f(x_i^*), \quad x_{i+1}^0 \neq x_i^*.$$

c) Combining (4) and (5), and suppressing the intermediate solution of the tunneling phases, the descent property of the algorithm is given by

$$(6) \quad f(x_1^*) \geq f(x_2^*) \geq \dots \geq f(x_G^*) \geq f(x_G^0)$$

where  $x_G^*$  corresponds to the global minimizer of  $f(x)$  and where  $x_i^* \neq x_j^*$  for  $i \neq j$ .

**4. Numerical experiments.** It is the purpose of this section to illustrate the characteristics of the Tunneling Algorithm as well as the methods already mentioned in § 1, namely, two versions of the random starting method. For this purpose, sixteen numerical examples were solved in a CDC-6400 computer, using single precision arithmetic, with all the algorithm coded in standard FORTRAN IV.

#### 4.1. Stopping conditions.

**4.1.1. Tunneling algorithm.** The value assigned to the different parameters involved in the minimization phase were the following: the convergence criteria were chosen as

$$\varepsilon_1 = 10^{-9}$$

and the nonconvergence criteria were set to

$$N_b = 20.$$

The values assigned to those parameters involved in the tunneling phase are given in Appendix I.

**4.1.2. Multiple random start method (MRS).** This algorithm is made-up of a random number generator which gives the starting point of any minimization phase. The actual implementation of this technique is the following. For a given minimization algorithm we ran it for different starting points, selected at random, and stored the obtained local minima of  $f(x)$  and their location. (Note that in this method neither the number of starting points nor their location are known in advance.)

The stopping condition employed in this algorithm to denote convergence at any local minimum was

$$(7) \quad f'_x(x) f_x(x) \leq \varepsilon_1 = 10^{-9}.$$

Nonconvergence is defined to occur whenever the number of bisections in a given minimization step is greater than 20 or when the limiting computing time is reached.

**4.1.3. Modified multiple random start method (MMRS).** This algorithm is in the same line as the previous one. The only modification involved in MMRS is as follows: Let  $x^*$  be the location where a minimum of  $f(x)$  occurs, say  $f(x^*)$ . To continue the search for additional minima, a random point  $x^0$  is generated and used as the location to start the next minimization phase provided

$$(8) \quad f(x^0) - f(x^*) \leq 10^{-3}$$

is satisfied. If for a given  $x^0$ , (8) is not satisfied, a new point  $x^0$  is generated at random and satisfaction of (8) is tested; this procedure is repeated until inequality (8) is satisfied or a time limit is reached. (As in the MRS method neither the number of starting points nor their location are known in advance.)

**4.2. Measurement of success.** Each of the problems presented in § 4.4 was solved several times starting each run at different points. Let  $N_G$  denote the number of global minima exhibited by a particular problem ( $N_G = 1$  for problems with one strict global minimum);  $N_r$  the number of starting points employed in each problem and  $M_i$ ,  $i = 1, 2, \dots, N_r$  the number of different global minima found when starting at each one of the  $N_r$  starting points; under these conditions we define a quantity  $p$ , a measure of success, as

$$(9) \quad p = \frac{\sum_{i=1}^{N_r} M_i}{N_G N_r}.$$

From this equation we observe that  $0 \leq p \leq 1$ , and the larger value of  $p$ , the more robust is the algorithm since more often finds all the  $N_G$  global minimizers.

**4.3. Running time.** Each of the problems presented in the following section was first solved using the tunneling algorithm. Let  $t_i$ ,  $i = 1, 2, \dots, N_r$  denote the time consumed by the tunneling algorithm, in CPU secs, from the beginning of the first minimization phase up to the end of the last minimization phase. With these times, we define an average running time as

$$(10) \quad t_{av} = \frac{\sum_{i=1}^{N_r} t_i}{N_r}.$$

This average time,  $t_{av}$ , was employed as limiting time when running the alternate methods, giving this a basis to compare all different methods.

**4.4. Examples.** In this section we describe the sixteen examples employed to compare the algorithms. The region of interest, where the examples were considered as well as the starting points employed in each example, is given.

*Example 1. "Two-dimensional Shubert function" (Shubert [10]).*

$$(11) \quad f(x_1, x_2) = \left\{ \sum_{i=1}^5 i \cos [(i+1)x_1 + i] \right\} \left\{ \sum_{i=1}^5 i \cos [(i+1)x_2 + i] \right\},$$

$$-10 \leq x_i \leq 10, \quad i = 1, 2.$$

This function exhibits 760 local minima in the region considered, eighteen of which are also global minima. The value of  $f(x)$  at these global minima is  $f(x_1, x_2) = -186.73091$  and the coordinates of each one of the minimizers are the following:

$$\begin{aligned} &(-7.08350, -7.70831), (-0.80032, -7.70831), (5.48286, -7.70831), \\ &(-7.70831, -7.08350), (-1.42513, -7.08350), (4.85805, -7.08350), \\ &(-7.08350, -1.42513), (-0.80032, -1.42513), (5.48286, -1.42513), \\ &(-7.70831, -0.80032), (-1.42513, -0.80032), (4.85805, -0.80032), \\ &(-7.08350, 4.85805), (-0.80032, 4.85805), (5.48286, 4.85805), \\ &(-7.70831, 5.48286), (-1.42513, 5.48286), (4.85805, 5.48286). \end{aligned}$$

The starting points for this example were  $(7, 7)$ ,  $(7, -7)$ ,  $(-7, 7)$ ,  $(0, 0)$ .

*Example 2. "Two-dimensional Shubert function".*

*Case 1.*

$$(12) \quad f(x_1, x_2) = \left\{ \sum_{i=1}^5 i \cos [(i+1)x_1 + i] \right\} \left\{ \sum_{i=1}^5 i \cos [(i+1)x_2 + i] \right\}$$

$$+ \frac{1}{2}[(x_1 + 1.42513)^2 + (x_2 + 0.80032)^2], \quad -10 \leq x_i \leq 10, \quad i = 1, 2.$$

This function exhibits the same characteristics as Example 3 with only one global minimum located at

$$x_1 = -1.42513, \quad x_2 = -0.80032$$

with a function value  $f(x) = -186.73091$ . The starting points were the same as those in Example 1.

*Example 3. "Two-dimensional Shubert function".*

*Case 2.*

$$(13) \quad f(x_1, x_2) = \left\{ \sum_{i=1}^5 i \cos [(i+1)x_1 + i] \right\} \left\{ \sum_{i=1}^5 i \cos [(i+1)x_2 + i] \right\}$$

$$+ (x_1 + 1.42513)^2 + (x_2 + 0.80032)^2, \quad -10 \leq x_i \leq 10, \quad i = 1, 2.$$

This example has the same characteristics of Examples 3 and 4, with only one global minimum located at

$$x_1 = -1.42513, \quad x_2 = -0.80032$$

with a function value  $f(x) = -186.73091$ . The starting points were the same as for Example 1.

*Example 4. "Six-hump camelback function" (Hardy [6]).*

$$(14) \quad f(x_1, x_2) = [4 - 2.1x_1^2 + \frac{1}{3}x_1^4]x_1^2 + x_1x_2 + [-4 + 4x_2^2]x_2^2, \quad -3 \leq x_1 \leq 3, \quad -2 \leq x_2 \leq 2.$$

This function exhibits 6 local minimizers, two of which are also global, located at  $(-0.08983, 0.7126)$  and  $(0.08983, -0.7126)$  and the function value is  $f(x) = -1.0316285$ .

The starting points employed in this example were  $(-2.9, -1.9)$ ,  $(-2.9, 1.9)$ ,  $(2.9, -1.9)$ , and  $(2.9, 1.9)$ .

Examples 5 through 7 are taken from the general formula

$$(15) \quad f(x) = \frac{\pi}{n} \left\{ k \sin^2(\pi y_1) + \sum_{i=1}^{n-1} [(y_i - A)^2 (1 + k \sin^2(\pi y_{i+1}))] + (y_n - A)^2 \right\},$$

$$y_i = 1 + 0.25(x_i - 1), \quad -10 \leq x_i \leq 10, \quad i = 1, 2, \dots, n$$

where the constants  $k$  and  $A$  were fixed at 10 and 1 respectively, and  $n$  denotes the dimensionality of the problem.

The function given by (15) exhibits many local minima but only one of these is also a global minimum. The location of this global minimizer is fixed at

$$x_i = 1, \quad i = 1, 2, \dots, n$$

and the function attains the value  $f(x) = 0$ , irrespective of the dimensionality of the problem.

*Example 5. Equation (15),  $n = 2$ .* The starting points used in this example were  $(-8, 8)$ ,  $(8, 8)$ ,  $(-5, 5)$ , and  $(-8, 8)$ .

*Example 6. Equation (15),  $n = 3$ .* The starting points used in this example were  $(8, 8, 8)$ ,  $(-5, 5, -5)$ ,  $(8, -8, 8)$ , and  $(-8, -8, -8)$ .

*Example 7. Equation (15),  $n = 4$ .* The starting points used in this example were  $(-5, -5, -5, -5)$ ,  $(5, 5, 5, 5)$ ,  $(-5, -5, 5, 5)$ , and  $(5, 5, -5, -5)$ .

Examples 8 through 10 are taken from the general formula

$$(16) \quad f(x) = \frac{\pi}{n} \left\{ k \sin^2(\pi x_1) + \sum_{i=1}^{n-1} [(x_i - A)^2 (1 + k \sin^2(\pi x_{i+1}))] + (x_n - A)^2 \right\},$$

$$-10 \leq x_i \leq 10, \quad i = 1, 2, \dots, n,$$

where the constants  $k$  and  $A$  were fixed at 10 and 1 respectively and  $n$  denotes the dimensionality of the problem.

*Example 8. Equation (16),  $n = 5$ .* The starting points used for this example were  $(8, 8, 8, 8, 8)$ ,  $(-8, -8, 0, 8, 8)$ ,  $(8, 8, 0, -8, -8)$ , and  $(-8, -8, -8, -8, -8)$ .

*Example 9. Equation (16),  $n = 8$ .* The starting points used in this example were

a)  $x_i = 8, \quad i = 1, 2, \dots, 8,$

b)  $x_i = -8, \quad i = 1, 2, \dots, 6, \quad x_i = 0, \quad i = 7, 8,$

c)  $x_i = 8(-1)^i, \quad i = 1, 2, \dots, 6, \quad x_i = 0, \quad i = 7, 8,$

d)  $x_i = 0, \quad i = 1, 2, \dots, 8.$

*Example 10. Equation (16),  $n = 10$ .* The starting points used in this example were

a)  $x_i = 0, \quad i = 1, 2, \dots, 10,$

b)  $x_i = 2, \quad i = 1, 2, \dots, 10,$

c)  $x_i = 6, \quad i = 1, 2, \dots, 10,$

d)  $x_i = -1, \quad i = 1, 2, \dots, 10.$

Examples 11 through 16 have been taken from the general formula

$$(17) \quad f(x) = k_1 \sin^2 \pi l_0 x_1 + k_1 \sum_{i=1}^{n-1} [(x_i - A)^2 (1 + k_0 \sin^2 \pi l_0 x_{i+1})] \\ + k_1 (x_n - A)^2 (1 + k_0 \sin^2 \pi l_0 x_n),$$

where the constants in this equation have been fixed as follows:  $k_0 = 1$ ,  $k_1 = 0.1$ ,  $A = 1$ ,  $l_0 = 3$ , and  $l_1 = 2$ . The examples generated by using (17) exhibit many local minimizers, but only one of these is also global located at  $x_i^* = 1$ ,  $i = 1, 2, \dots, n$  and the function value at this point is  $f(x^*) = 0$ .

*Example 11.* Equation (17),  $n = 2$ ,  $-10 \leq x_i \leq 10$ ,  $i = 1, 2$ . The starting points for this example were  $(9, 9)$ ,  $(-9, -9)$ ,  $(-9, 9)$ , and  $(9, -9)$ .

*Example 12.* Equation (17),  $n = 3$ ,  $-10 \leq x_i \leq 10$ ,  $i = 1, 2, 3$ . The starting points were  $(5, 5, 5)$ ,  $(5, -5, 5)$ ,  $(-5, 5, -5)$ , and  $(-5, -5, -5)$ .

*Example 13.* Equation (17),  $n = 4$ ,  $-10 \leq x_i \leq 10$ ,  $i = 1, 2, 3, 4$ . The starting nominal points for this example were  $(5, 5, 5, 5)$ ,  $(5, 5, -5, -5)$ ,  $(-5, -5, 5, 5)$ , and  $(-5, 0, 0, 5)$ .

*Example 14.* Equation (17),  $n = 5$ ,  $-5 \leq x_i \leq 5$ ,  $i = 1, 2, \dots, 5$ . The starting points for this example were

- a)  $x_i = 3$ ,  $i = 1, 2, \dots, 5$ ,
- b)  $x_i = 3(-1)^{i-1}$ ,  $i = 1, 2, \dots, 5$ ,
- c)  $x_i = 3(-1)^i$ ,  $i = 1, 2, \dots, 5$ ,
- d)  $x_i = -3$ ,  $i = 1, 2, \dots, 5$ .

*Example 15.* Equation (17),  $n = 6$ ,  $-5 \leq x_i \leq 5$ ,  $i = 1, 2, \dots, 6$ . The starting points used for this example were

- a)  $x_i = 3$ ,  $i = 1, 2, \dots, 6$ ,
- b)  $x_i = 3$ ,  $i = 1, 2, 3$ ,  $x_i = -3$ ,  $i = 4, 5, 6$ ,
- c)  $x_i = -3$ ,  $i = 1, 2, 3$ ,  $x_i = 3$ ,  $i = 4, 5, 6$ ,
- d)  $x_i = -3$ ,  $i = 1, 2, \dots, 6$ .

*Example 16.* Equation (17),  $n = 7$ ,  $-5 \leq x_i \leq 5$ ,  $i = 1, 2, \dots, 7$ . The starting points for this example were

- a)  $x_i = -3$ ,  $i = 1, 2, \dots, 7$ ,
- b)  $x_i = -3$ ,  $i = 1, 2, 3$ ,  $x_4 = 0$ ,  $x_i = 3$ ,  $i = 5, 6, 7$ ,
- c)  $x_i = 3$ ,  $i = 1, 2, 3$ ,  $x_4 = 0$ ,  $x_i = -3$ ,  $i = 5, 6, 7$ ,
- d)  $x_i = 3$ ,  $i = 1, 2, \dots, 7$ .

**5. Analysis of the numerical results.** In this section we present the results obtained for the sixteen examples mentioned in the previous section when solved using the various methods previously described. Additionally we present the result obtained when running the tunneling algorithm coupled to several local minimization techniques.

Next, we present the results obtained by the tunneling, MRS and MMRS methods, using in their corresponding minimization phases the ordinary gradient method to find the local minimizers (Montalvo [9]).

With the purpose of having as much information as possible to have a better idea on the performance of the methods, we report the following data for each one of the sixteen examples.

- a) Total time required for reaching the global minimizers.
- b) Total number of functions and gradient evaluations.
- c) Partial time spent in the minimization phases.
- d) Total number of functions and gradient evaluation during the minimization phases.

e) Minimization phases required to reach the global minimizers.  
 f) Measurement of success (p). Here we have to differentiate the following situations:

f.1) For functions exhibiting only one global minimum, the reported value for the tunneling algorithm varies between zero and one, since this value corresponds to the number of times that, in the average, the global minimum was reached starting at four different points. For MRS and MMRS methods, the reported values are either zero or one depending whether or not the global minimum was located or missed.

f.2) For functions exhibiting more than one global minimum the reported value is obtained as follows: for each one of the runs starting at different points, we counted the number of different global minima found and then an average between these four numbers is computed; the final number is obtained dividing the latter by the total number of global minima exhibited by the function.

From the results shown in Table 1, we can make the following observations.

a) In all sixteen examples, the tunneling algorithm was the only one in locating all global minimizers, except in Example 1 where the method located seventeen out of the eighteen globals.

b) On Examples 5, 6, 7, 11 and, 12, the MMRS algorithm out-performed the tunneling algorithm as it was able to find the global minimizers faster than the tunneling algorithm.

c) On Examples 2, 3, 8, 10, and 13–16 both MRS and MMRS were not able to locate the global minimizers, while the tunneling algorithm did; these examples include the one with  $n = 10$ .

Taking into account the above statements we reach the conclusion that the tunneling algorithm is usually faster and more often converges to the global minimizers than the other methods; this advantage increases with the density of relative minima. In particular, the starting point generated by the tunneling algorithm (exit point generated by the tunneling phase), can be considered as a “highly educated” random starting point for initiating each of the minimization phases.

*Use of alternate local minimization techniques.* In Table 2 we present the results obtained when running the tunneling algorithm coupled to different local minimization techniques, namely: (a) ordinary gradient, (b) conjugate gradient and (c) Newton’s method.

For the ten examples analyzed (Examples 1–10) we have focused the comparison on the time spent by the algorithms in (a) the minimization phases and (b) reaching the global; that is, we allowed the program to reach the global minimizer. For each case we report the above mentioned two times as well as the average number of minimizations performed and the average number of times the global minimizer was located.

We observe that in all cases the tunneling algorithm was always able to locate the global, independently of the local minimization technique being used. However, one can observe the following anomalous results in Table 2.

a) The ordinary gradient method took the least total times for reaching the global in Examples 1, 3, 8, and 10.

b) The conjugate gradient method was the best in Examples 2, 3, and 9.

c) Newton’s method became the best method for Examples 4, 5, 6, and 7.

Another important fact that can also be observed in Table 2, is that in seven of the ten examples, the ordinary gradient method required the least minimization phases. It is the authors’ opinion that this characteristic was probably due to the fact that even during the search for local minimizers the minimization method was also “tunneling”;

TABLE 1  
Summary of numerical results for the tunneling, MRS, and MMRS methods.

Ex/dim.	Method	Total time (CPU-secs)	Total evaluations		Minimization time (CPU-secs)	Evaluations during minimization phase		Number of minimizations	$p$
			Functions	Gradient		Functions	Gradient		
1/2	Tunnel	87.045	12,160	1,731	2.113	1,047	97	17	0.9445
	MRS	88.089	35,479	9,572	87.845	35,479	9,572	244	0.5
	MMRS	87.066	50,462	7	0.148	75	7	1	0.0555
2/2	Tunnel	8.478	2,912	390	1.045	525	53	3	1
	MRS	5.197	*	*	*	*	*	10	0
	MMRS	4.313	*	*	*	*	*	2	0
3/2	Tunnel	5.984	2,180	274	1.409	710	69	3	1
	MRS	2.094	*	*	*	*	*	4	0
	MMRS	6.018	3,350	19	0.292	136	19	1	0
4/2	Tunnel	1.984	1,496	148	0.033	57	17	2	1
	MRS	0.036	61	19	0.034	61	19	2	1
	MMRS	2.036	6,000	10	0.016	32	10	1	0.5
5/2	Tunnel	3.238	2,443	416	0.947	1,116	273	2.75	1
	MRS	64.0	*	*	*	*	*	1	0
	MMRS	1.062	1,241	287	0.997	1,195	287	3	1
6/3	Tunnel	12.915	7,325	1,328	4.435	3,823	848	3.25	1
	MRS	3.516	2,861	784	3.485	2,861	784	3	1
	MMRS	4.024	3,443	726	3.231	2,647	726	3	1
7/4	Tunnel	20.450	4,881	1,371	1.91	1,126	376	4.25	1
	MRS	3.391	*	*	*	*	*	6	0
	MMRS	4.335	3,076	457	2.289	1,369	457	2	1
8/5	Tunnel	11.885	7,540	1,122	9.32	6,471	881	2	1
	MRS	8.107	*	*	*	*	*	1	0
	MMRS	11.925	9,458	171	2.112	1,506	171	1	0
9/8	Tunnel	45.474	19,366	2,370	35.644	16,138	2,011	2.5	1
	MRS	38.091	17,229	2,143	38.091	17,229	2,143	2	1
	MMRS	45.535	22,193	1,771	30.757	14,126	1,771	1	0
10/10	Tunnel	68.22	23,982	3,272	67.283	22,191	3,135	2.5	1
	MRS	192.0	*	*	*	*	*	1	0
	MMRS	68.26	25,966	2,913	62.47	23,093	2,913	1	0
11/2	Tunnel	4.364	2,613	322	0.762	736	158	2.5	0.5
	MRS	6.308	6,851	876	6.308	6,851	876	5	0
	MMRS	1.792	1,867	250	1.406	1,441	250	4	1
12/3	Tunnel	12.378	6,955	754	3.927	3,142	479	4.25	0.75
	MRS	13.291	10,566	1,652	13.227	10,566	1,652	9	0
	MMRS	2.975	2,316	359	2.139	2,316	359	3	1
13/4	Tunnel	8.35	3,861	588	3.076	1,863	390	3.75	0.75
	MRS	9.851	6,659	740	9.821	6,659	740	2	0
	MMRS	8.376	6,234	273	2.294	1,419	273	2	0
14/5	Tunnel	28.33	10,715	1,507	7.249	3,565	797	5.5	0.75
	MRS	51.707	28,347	4,002	51.712	28,347	4,002	7	0
	MMRS	28.362	17,339	1,098	11.150	5,746	1,098	3	0
15/6	Tunnel	33.173	12,786	1,777	17.282	7,839	1,329	3.5	1
	MRS	41.065	19,301	2,784	41.028	19,301	2,784	2	0
	MMRS	33.231	18,985	132	1.263	479	132	2	0
16/7	Tunnel	71.981	16,063	2,792	15.350	6,142	1,013	7.5	0.75
	MRS	92.615	38,483	5,411	92.546	38,483	5,411	8	0
	MMRS	72.027	36,195	435	5.977	2,132	435	2	0

\* Failure in convergence.



TABLE 2  
*Results for the tunneling algorithm coupled with alternate local minimization procedures.*

Ex/Dim.	Minimization phase	$t_{av}$	$t_{min}$	Total no. of minimizations	Globals found
1/2	Ord. grad.	87.045	2.113	17	17
	Conj. grad.	105.474	4.283	19.75	17.25
	Newton	128.656	3.210	18	17
2/2	Ord. grad.	8.478	1.054	3	1
	Conj. grad.	4.656	0.064	6.75	1
	Newton	8.173	2.157	7	1
3/2	Ord. grad.	5.984	1.409	3	1
	Conj. grad.	5.944	0.446	4.75	1
	Newton	10.466	1.699	7	1
4/2	Ord. grad.	1.984	0.033	2	2
	Conj. grad.	0.586	0.070	2	2
	Newton	0.43	0.061	3.5	2
5/2	Ord. grad.	3.283	0.947	2.75	1
	Conj. grad.	1.052	0.091	3.75	1
	Newton	0.327	0.110	2.125	1
6/3	Ord. grad.	12.915	4.435	3.25	1
	Conj. grad.	7.218	0.187	3	1
	Newton	5.863	0.377	2.75	1
7/4	Ord. grad.	20.45	1.910	4.25	1
	Conj. grad.	11.782	0.323	3.125	1
	Newton	5.29	3.480	2	1
8/5	Ord. grad.	11.885	9.320	2	1
	Conj. grad.	15.850	1.593	5.25	1
	Newton	46.195	6.829	11.25	1
9/8	Ord. grad.	45.474	35.644	2.5	1
	Conj. grad.	24.773	6.001	4.75	1
	Newton	157.830	78.941	15	1
10/10	Ord. grad.	68.220	67.283	2.5	1
	Conj. grad.	118.737	7.890	11.25	1
	Newton	230.381	208.570	6.75	1

this means that the ordinary gradient method was not easily trapped by local minimizers as it probably occurred to the alternate procedures. (Obviously, this depends on the one-dimensional search being used.) However, this behavior has not yet been satisfactorily explained.

**6. Final comments.** Based on the material previously presented we conclude that the Tunneling Algorithm has the following characteristics that make it very attractive for finding global minima.

a) It is superior to several other methods previously reported since generally, it locates the global minimizer(s) faster than do the other methods. It should be pointed out that even in the case when the tunneling algorithm is not highly efficient (for several examples) it still locates them.

b) It was experimentally observed that the efficiency  $\cap$  the tunneling algorithm is not affected by the density of relative (nonglobal) minimizers, while the other methods tested are strongly affected, in a negative direction, by this property of the objective function.

c) When comparing different local minimization methods coupled to the tunneling algorithm, no one of these showed to be more efficient than the others. However, it should be pointed out that when the tunneling algorithm was coupled to the ordinary gradient method, this version required the least amount of minimization phases.

### Appendix I.

*Determination of the tunneling function.* In this appendix we present the necessary steps to determine the actual parameters involved in the definition of  $T(x)$ , namely

$$[l, (x_i^*, i = 1, 2, \dots, l), (\eta_i, i = 1, 2, \dots, l), x_m, \lambda_0, f^*]$$

where  $T(x)$  was defined as

$$(A.0) \quad T(x) = \frac{f(x) - f^*}{\left\{ \prod_{i=1}^l [(x - x_i^*)'(x - x_i^*)]^{\eta_i} \right\} [(x - x_m)'(x - x_m)]^{\lambda_0}}$$

*Step 1. Determination of  $f^*$ .* Once any local minimum has been found at  $x^*$ , we want the tunneling algorithm to ignore all the local minima whose function value are higher than  $f(x^*)$ , since they are irrelevant for approaching the global minimizer. This objective can be achieved by letting  $f^*$  be the lowest function value obtained so far. With this selection of  $f^*$  we accomplish our objective of tunneling below irrelevant local minima even if we do not know how many they are nor their location.

*Step 2. Determination of  $\eta_i$  and  $x_i^*$ ,  $i = 1, 2, \dots, l$ .* To clearly illustrate this step, let us assume that  $l = 1$  that is the algorithm has found only one local minimum at  $x_1^*$ , with a function value  $f^* = f(x_1^*)$ . We employ Equation (A.0) with  $\lambda_0 = 0$ , thus the tunneling function simplifies to ( $\lambda_1$  replacing  $\eta_1$ )

$$(A.1) \quad T(x) = \frac{f(x) - f^*}{[(x - x_1^*)'(x - x_1^*)]^{\lambda_1}}$$

where  $x = x_1^* + \varepsilon$ . The correct value of  $\eta_1$  is found iteratively starting from  $\lambda_1 = 1$ , until the following descent property holds.

$$(A.2) \quad T'_x(x)\Delta x < 0$$

provided

$$(A.2a) \quad \varepsilon'\Delta x > 0$$

where  $\varepsilon$  is a random vector with  $\|\varepsilon\| \ll 1$ ,  $\Delta x$  is the displacement produced by the tunneling algorithm using the trial value of  $\lambda_1$  in (A.2). It can easily be proved that (A.2) and (A.2a) are simultaneously satisfied provided  $T(x)$  has a pole at  $x = x_1^*$ .

If for a given  $\lambda_1$ , inequalities (A.2) are not satisfied, increase  $\lambda_1$  by  $\Delta\lambda_1$  until the above descent property is satisfied. For subsequent steps the actual value of  $\eta_1$  is determined according to

$$(A.3) \quad \eta_1 = \begin{cases} 0 & \text{if } \gamma \geq 1 + \varepsilon_2, \\ \lambda_1 & \text{if } \gamma \leq 1 - \varepsilon_2, \\ (\lambda_1/2)[1 + (1 - \gamma)/\varepsilon_2] & \text{if } 1 - \varepsilon_2 \leq \gamma \leq 1 + \varepsilon_2 \end{cases}$$

where  $\gamma = \|x - x^*\|$  and  $\varepsilon_2$  is a small prescribed number. Note that Equation (A.3) is

a ramp function which continuously switches the denominator in Equation (A.1), when crossing the unit circle centered at  $x_i^*$ . This switching procedure is necessary because for some functions, as the tunneling algorithm iterates, the denominator in Equation (A.1) might become very large when  $\|x - x_i^*\| > 1$ , forcing the tunneling function to become very flat and close to zero, thus slowing down the convergence of the tunneling algorithm.

Let us now consider the case of  $l > 1$ , that is, when the function has multiple local minima at the level  $f^*$ , and  $l$  successive applications of the minimization phase with  $(l-1)$  tunneling phases have identified the points  $x_i^*$ ,  $i = 1, 2, \dots, l-1$  as local minimizers of  $f(x)$  at the level  $f^*$ . As each  $x_i^*$ ,  $i = 1, 2, \dots, l-1$  is found and having computed  $\eta_i$ ,  $i = 1, 2, \dots, l-1$ ,  $\eta_l$  is computed employing the above iterative procedure using the function

$$(A.4) \quad T(x) = \frac{f(x) - f^*}{\prod_{i=1}^{l-1} [(x - x_i^*)'((x - x_i^*))^{\eta_i}[(x - x_i^*)'(x - x_i^*)]^{\lambda_i}]}$$

The actual value of  $\eta_l$  is determined as in (A.3), that is

$$(A.5) \quad \eta_l = \begin{cases} 0 & \text{if } \gamma_l \geq 1 + \varepsilon_2, \\ \lambda_l & \text{if } \gamma_l \leq 1 - \varepsilon_2, \\ (\lambda_l/2)[1 + (1 - \gamma_l)/\varepsilon_2] & \text{if } 1 - \varepsilon_2 \leq \gamma_l \leq 1 + \varepsilon_2 \end{cases}$$

where  $\gamma_l = \|x - x^*\|$ .

*Remark.* The value of  $l$  is reset to  $l = 1$  whenever the new local minimum found has a lower value than the previous one.

*Step 3. Determination of  $x_m$  and  $\lambda_0$ .* For the first iteration of each tunneling phase we set

$$(A.6) \quad x_m = x_l^* \quad \text{and} \quad \lambda_0 = 0.$$

For subsequent iterations within the tunneling phase the position  $x_m$  and pole strength  $\lambda_0$  of the movable pole are automatically computed in order to cancel out any undesirable relative minimum that the function  $T(x)$  might have. Once any of these minima is detected we proceed as follows.

Let  $x$  be the present point and  $\hat{x}$  the previous point generated by the tunneling algorithm. Then the position of  $x_m$  is determined as follows

$$(A.7) \quad x_m = \begin{cases} \hat{x} & \text{if } \|\hat{x} - x\| < 1, \\ \xi\hat{x} + (1 - \xi)x & \text{if } \|\hat{x} - x\| \geq 1 \end{cases}$$

where  $0 < \xi \leq 1$  is chosen such that  $\|x - x_m\| < 1$ . We note that the movable pole thus located will always be within the unit circle centered at the current position  $x$  of the zero finding algorithm, so we are free to choose the pole strength as large as it is required to cancel out any undesirable minimum that  $T(x)$  might have in the neighborhood of  $x$ . To achieve this, let  $\lambda_0$  be the pole strength value employed when moving from the previous point  $\hat{x}$  to  $x$ : let  $\Delta x$  be the displacement produced by the tunneling algorithm leading from  $x$  to the new point  $\tilde{x}$  and consider the inner product

$$(A.8) \quad u = (x - \hat{x})' \Delta x.$$

If  $u > 0$ , we can retain the pole strength while moving from  $x$  to  $\tilde{x}$ . If  $u < 0$ , the value of  $\lambda_0$  is no longer appropriate, and a new  $\lambda_0$  is found iteratively starting from  $\lambda_0 = \hat{\lambda}_0$  and increasing it by  $\Delta\lambda_0$  until  $u > 0$ .

Once the tunneling algorithm has moved away from the local minimum of  $T(x)$

which motivated the increase of the pole strength, one could return to a simpler tunneling function with a simpler geometry, by resetting the movable pole strength  $\lambda_0$  to zero.

A heuristic rule to detect when the tunneling algorithm has left the undesirable neighborhood of the local minimum of  $T(x)$  is as follows. Compute the displacement that would be produced by the tunneling algorithm, for two values of  $\lambda_0$ , namely, for the current value of  $\lambda_0$  we compute  $\Delta x(\lambda_0)$ ; for  $\lambda_0 = 0$  we compute  $\Delta x(0)$ . Next, we measure the angle between these two displacements and update the value of the pole strength  $\lambda_0$  according to the following rule:

$$(A.9) \quad \lambda_0 = \begin{cases} 0 & \text{if } \Delta x'(0)\Delta x(\lambda_0) > 0, \\ \lambda_0 & \text{if } \Delta x'(0)\Delta x(\lambda_0) \leq 0. \end{cases}$$

*Step 4. Stopping condition for the tunneling phase.*

*Case 1.* Recall that the tunneling algorithm should be stopped whenever  $T(x) \leq 0$ . In the actual implementation of this phase this condition has been replaced by

$$(A.10) \quad T(x) \leq \varepsilon_3.$$

We note that any  $x$  satisfying (A.10) is an acceptable starting point for the next minimization phase.

*Case 2.* This case occurs when for a given  $\varepsilon$ -vector, in a given number of iterations,  $N_s$ , the zero finding algorithm fails to locate  $x^0 \in \Omega$  such that  $T(x^0) \leq \varepsilon_3$ . If this is the case we are in the position of selecting another  $\varepsilon$ -vector, different from the previous one, and restart the tunneling phase for this new  $\varepsilon$ -vector, until satisfaction of inequality (A.10) is achieved; this procedure is repeated  $N_\varepsilon$  times.

Finally, without changing any of the parameters of the tunneling function  $T(x)$ , we initiate the tunneling phase from any starting point selected at random  $x \in \Omega$ , that is, no longer within an  $\varepsilon$ -neighborhood of the last minimum found; this procedure is repeated  $N_R$  times.

The search for any point  $x^0$  such that  $T(x^0) \leq 0$  was implemented using the so-called Restoration Algorithm (Miele [8]), as described in § 2.3.4, to produce the displacements  $\Delta x$ . The values assigned to the parameters involved in this phase were the following:

The convergence criteria were chosen as

$$\varepsilon_3 = 10^{-3};$$

the  $\eta$ -switching parameter was fixed at

$$\varepsilon_2 = 10^{-5};$$

and the nonconvergence criteria were the following:

$$: \lambda_{\max} = 5$$

$$: \lambda_0^* = 1$$

$$: \beta_R = 2^{-20}$$

$$\text{Number of bisections: } N_b = 20$$

$$\text{Number of search steps: } N_s = 100$$

$$\text{Number of } \varepsilon\text{-vectors: } N_\varepsilon = 2n$$

$$\text{Number of final random vectors: } N_R = 2n$$

where  $n$  denotes the dimensionality of the problem.

## REFERENCES

- [1] R. S. ANDERSSEN AND P. BLOOMFIELD, *Properties of the random search in global optimization*, J. Optim. Theory Appl., 16 (1975), pp. 383–398.
- [2] M. AVRIEL, *Nonlinear Programming Analysis and Methods*, Prentice-Hall, Englewood Cliffs, NJ, 1976.
- [3] F. H. BRANNIN, JR., *Widely convergent method for finding multiple solutions of simultaneous nonlinear equations*, IBM J. Res. Dev. (September 1972), pp. 504–522.
- [4] L. DIXON AND G. P. SZEGO, eds., *Towards Global Optimization*, North-Holland, Amsterdam, 1975.
- [5] A. A. GOLDSTEIN AND J. F. PRICE, *On descent from local minima*, Math. Comp., 25 (1971), pp. 569–574.
- [6] J. HARDY, *An implemented extension of Brannin's method*, in *Towards Global Optimization*, North-Holland, Amsterdam, 1975.
- [7] D. M. HIMMELBLAU, *Applied Nonlinear Programming*, McGraw-Hill, New York, 1972.
- [8] A. MIELE, H. Y. HUANG AND J. C. HEIDEMAN, *Sequential gradient-restoration algorithm for the minimization of constrained functions—ordinary and conjugate gradient versions*, J. Optim. Theory Appl., 4 (1969), pp. 213–243.
- [9] A. MONTALVO, *Desarrollo de un nuevo algoritmo para la minimización global de funciones*, Ph.D. thesis, School of Engineering, National Autonomous University of Mexico, 1979.
- [10] B. O. SHUBERT, *A sequential method seeking the global maximum of a function*, SIAM J. Numer. Anal., 9 (1972), pp. 379–388.

## ON FITTING EXPONENTIALS BY NONLINEAR LEAST SQUARES\*

J. M. VARAH†

**Abstract.** This paper is concerned with the problem of fitting discrete data or a continuous function by least squares using exponential functions. We examine the questions of uniqueness and sensitivity of the best least squares solution, and provide analytic and numerical examples showing the possible nonuniqueness and extreme sensitivity of these solutions.

**Key words.** exponential fitting, nonlinear least squares

**1. Introduction.** The discrete problem of fitting data  $(t_i, y_i)$ ,  $i = 1, \dots, n$  by a sum of exponentials  $y(t) = \sum_{j=1}^m a_j e^{b_j t}$  in the best least squares sense so as to minimize

$$I(\mathbf{a}, \mathbf{b}) = \sum_{i=1}^n (y(t_i) - y_i)^2,$$

is well known to be difficult computationally. The extreme sensitivity of the exponential powers  $\{b_j\}$  (assumed throughout to be real and negative) was pointed out by Lanczos (1956, pp. 276ff), who showed for a particular example that various parameter values can give near-optimal results. Since then, special algorithms have been proposed (for example Osborne (1975)) and problems of this type have been used to test more general nonlinear least squares algorithms (Golub and Pereyra (1973), Kaufman (1978)).

We are attracted to this problem because of its relation to the general problem of fitting parameters in differential equations. Indeed, if we consider a constant coefficient model

$$y^{(m)} + c_{m-1}y^{(m-1)} + \dots + c_0 = 0,$$

then the general solution  $y(t)$  is as above, with the  $\{b_j\}$  the roots of the associated polynomial (assumed real and negative). Notice that we must also allow for the possibility of multiple roots or confluences, as pointed out by Rice (1968).

Recently, there has been interest in this problem in the more restrictive setting  $a_i > 0$ ,  $i = 1, \dots, m$  (the completely monotone case). Here a convex cone characterization is possible, some uniqueness results are known, and special algorithms have been proposed. See Evans et al. (1980), Kammler (1979), and Ruhe (1980). Although this positivity assumption is reasonable for some applications, we feel that in the context of parameter estimation it is more reasonable to allow the  $\{a_i\}$  to vary freely.

In § 2, we consider this (discrete) problem and show that the example of Lanczos is not at all pathological. For a given data set, and only 2 exponentials, not only can the coefficients of the best approximation be very sensitive to the data, but there can easily be more than one local or global best approximation.

This discrete problem is closely related to the continuous problem of fitting an exponential sum to a given function  $f(t)$  so as to minimize

$$I(\mathbf{a}, \mathbf{b}) = \frac{1}{2} \int_0^\infty (y(t) - f(t))^2 dt.$$

\* Received by the editors November 30, 1982, and in revised form August 1, 1983.

† Department of Computer Science, University of British Columbia, Vancouver, British Columbia, Canada V6T 1W5.

This problem is more amenable to analysis than the discrete one, and in § 3 we examine the question of uniqueness, and give examples of nonunique solutions, extending the work of Kammler (1979). Then in § 4 we treat the special case where  $f(t)$  is itself an exponential sum; in this case one can be more explicit about nonuniqueness, and we give several examples. Finally in § 5 we try to draw some conclusions and give some recommendations for treating such problems in general.

**2. The discrete problem.** We consider only two exponentials, since all the aspects of the problem can be seen in this case, and the necessary extensions to three or more exponentials are easily seen.

Thus the problem can be stated easily enough: we have a least squares function

$$(2.1) \quad I(\mathbf{a}, \mathbf{b}) = \sum_i (a_1 e^{b_1 t_i} + a_2 e^{b_2 t_i} - y_i)^2 \equiv \sum_i r_i^2$$

with first order minimum conditions

$$(2.2) \quad \begin{aligned} \frac{\partial I}{\partial a_1} &= 2 \sum_i r_i e^{b_1 t_i} = 0, & \frac{\partial I}{\partial b_1} &= 2 \sum_i r_i t_i a_1 e^{b_1 t_i} = 0, \\ \frac{\partial I}{\partial a_2} &= 2 \sum_i r_i e^{b_2 t_i} = 0, & \frac{\partial I}{\partial b_2} &= 2 \sum_i r_i t_i a_2 e^{b_2 t_i} = 0 \end{aligned}$$

or  $J^T \mathbf{r} = 0$ , where  $J$  is the Jacobian matrix  $J_{ij} = \partial r_i / \partial \alpha_j$ ,  $\alpha = (a_1, a_2, b_1, b_2)$ . The  $\partial I / \partial a_j = 0$  equations can be used to solve for the linear parameters  $a_1, a_2$ , and these substituted to give  $I$  as a function of  $b_1, b_2$  only, but this is messy and does not provide much insight.

Some idea of the nature of the critical point(s) arrived at by solving (2.2) can be obtained by evaluating the  $4 \times 4$  Hessian matrix  $H$ ,

$$(2.3) \quad \begin{aligned} H_{kj} &= \frac{1}{2} \frac{\partial^2 I}{\partial \alpha_k \partial \alpha_j} = \sum_i \left( \frac{\partial r_i}{\partial \alpha_k} \right) \left( \frac{\partial r_i}{\partial \alpha_j} \right) + \sum_i r_i \frac{\partial^2 r_i}{\partial \alpha_k \partial \alpha_j} \\ &= J^T J + G. \end{aligned}$$

Most of  $G$  is zero at a solution of (2.2), the only nonzero terms being

$$G_{33} = a_1 \sum_i r_i t_i^2 e^{b_1 t_i}, \quad G_{44} = a_2 \sum_i r_i t_i^2 e^{b_2 t_i}.$$

We should mention here that we are assuming  $b_1 \neq b_2$ ; in the confluent case  $b_1 = b_2$  we must adjust  $I$  accordingly. We will consider this more explicitly when dealing with the continuous problem in § 3 and 4.

This natural splitting of  $H$  into  $J^T J + G$  has been used by several authors as a way of measuring *intrinsic* and *parameter-dependent* sensitivity. Ramsin and Wedin (1977) define for this purpose curvatures  $\mu_i$  as generalized eigenvalues of  $Gx = \mu J^T Jx$ , and Bates and Watts (1980) define similarly normal and tangential curvatures as measures of intrinsic and parameter-dependent sensitivities. The idea is that with a different choice of parameterization, the ill-condition of  $H$  caused by  $J^T J$  might be removed. This is of course interesting and useful; however, we take the view here that we are specifically concerned with the parameters  $\{a_i, b_i\}$  as given, so that we are "stuck" with the ill-condition caused by the particular parameterization using powers of exponentials. Hence in particular we feel that the eigenvalues of the Hessian matrix  $H$  do provide a reasonable measure of the local sensitivity of the problem, that is, of the least squares function  $I(\mathbf{a}, \mathbf{b})$ .

This preamble on sensitivity is important because even the simple problem (2.1) can be very ill-conditioned. We give some indication of this using two well-known

examples: that of Lanczos mentioned earlier, and one due to Osborne (1972). The Osborne data are given in Table 1, after subtracting the asymptotic constant 0.3754. The Lanczos data (see Lanczos (1956, p. 276)) are generated from three exponentials,

$$(2.4) \quad .0951 e^{-t} + .8607 e^{-3t} + 1.5576 e^{-5t}$$

using  $\Delta t = .05$  and generating 24 points and truncating the values to two decimal places. We give this in Table 1 as well. Both data sets appear to have unique minima, given in Table 2.

TABLE 1

Osborne data				Lanczos data			
<i>t</i>	<i>y</i>	<i>t</i>	<i>y</i>	<i>t</i>	<i>y</i>	<i>t</i>	<i>y</i>
0.0	.4686	1.7	.1826	0.0	2.51	.60	.27
0.1	.5326	1.8	.1626	.05	2.04	.65	.23
0.2	.5566	1.9	.1466	.10	1.67	.70	.20
0.3	.5606	2.0	.1306	.15	1.37	.75	.17
0.4	.5496	2.1	.1146	.20	1.12	.80	.15
0.5	.5326	2.2	.1036	.25	0.93	.85	.13
0.6	.5056	2.3	.0916	.30	0.77	.90	.11
0.7	.4746	2.4	.0816	.35	0.64	.95	.10
0.8	.4426	2.5	.0726	.40	0.53	1.00	.09
0.9	.4086	2.6	.0626	.45	0.45	1.05	.08
1.0	.3756	2.7	.0556	.50	0.38	1.10	.07
1.1	.3426	2.8	.0486	.55	0.32	1.15	.06
1.2	.3096	2.9	.0446				
1.3	.2826	3.0	.0336				
1.4	.2526	3.1	.0356				
1.5	.2276	3.2	.0306				
1.6	.2046						

TABLE 2

	$a_1$	$a_2$	$b_1$	$b_2$	$I$	$\lambda_1(H)$	$\lambda_2(H)$
Osborne	1.93	-1.46	-1.29	-2.22	$.55 \times 10^{-4}$	.00011	.06
Lanczos	0.40	2.11	-1.81	-4.57	$1.0 \times 10^{-4}$	.00027	.03

Some idea of the ill-condition of these problems can be seen from the size of the Hessian eigenvalues; however, this is very local information and an understanding of the global behavior is really more helpful. Thus we give as well in Figs. 1 and 2 a geometrical picture with a 3-dimensional plot of the function  $\sqrt{I(b_1, b_2)}$  using the appropriate linear parameters  $a_1, a_2$  for each  $b_1, b_2$ . The ranges involved were  $-0.4 \cong b_1, b_2 \cong -7.0$  for Osborne,  $-0.4 \cong b_1, b_2 \cong -10.0$  for Lanczos. Of course the plots are symmetric about the line  $b_1 = b_2$ . We add that although the actual *location* of the minimum point is sensitive to the data, the global nature of the least squares surface is not; thus the same kind of surface is obtained, for example, using exact values of the Lanczos data (i.e. not truncated to 2 decimal places).

Notice that the Osborne data gives rise to a rather narrow valley to the minimum and beyond. This results from the fact that one exponential fits the data surprisingly well. The Lanczos plot is rather different: the valley is broader, in a different direction,



and not so easily distinguished. Both surfaces appear *convex* in the region  $b_1 > b_2$  with unique minima, although verifying convexity appears to be difficult, and we have not been able to characterize those data which lead to convex surfaces, or even give sufficient conditions. We shall return to this problem in the continuous case.

To appreciate the ill-condition involved here, we can try to measure the sensitivity of the parameters to changes in the data. For example, suppose the data values are in error by no more than  $\delta$ . Then we can tolerate a change in  $\sqrt{I}$  of  $\varepsilon = \delta\sqrt{N}$ . Following Bard (1974, p. 171), this gives rise to an uncertainty region about the minimum which, assuming  $I$  is locally quadratic, is the ellipse  $(\delta b)^T H(\delta b) \leq 2\varepsilon^2$ , where  $H$  is the  $2 \times 2$  Hessian using only  $b_1, b_2$  as variables (not the  $4 \times 4$  Hessian used earlier). Thus  $\|\delta b\|$  can be as large as  $\varepsilon/\sqrt{\lambda_1(H)}$ . With  $\delta = .001$  in the above cases, this uncertainty region is quite large; we have not computed it precisely but we know it contains the points in Table 3.

TABLE 3

	$b_1$	$b_2$	$b_1$	$b_2$
Osborne	-1.2	-2.5	-1.4	-1.9
Lanczos	-1.6	-4.4	-2.1	-4.7

Yet these data sets are not in the least pathological; other data sets give comparable results. Indeed, one can devise data sets where the situation is much worse, i.e. with a flatter, yet nonconvex surface, merely by forming a different exponential sum in (2.4). We consider one specific example briefly here, and return to it in § 4 where we discuss the continuous approximation problem. We use the general sum

$$(2.5) \quad \alpha_1 e^{-t} + \alpha_2 e^{-3t} + \alpha_3 e^{-5t},$$

choosing the special case  $\alpha = (.1, .4, -.3)$ .

The difference in the nature of the surface  $I(b_1, b_2)$  can be seen by examining the confluent case  $b_1 = b_2$ . Along this line the Osborne and Lanczos data appear to give rise to a *convex* function  $I(b_1, b_1)$ , with a unique minimum at some finite (negative) value of  $b_1$ . This point is in fact a *saddle point* of the surface, with the surface decreasing in value away from  $b_1 = b_2$  until the (apparent) global minimum is reached (see Figs. 1 and 2). However the new example is *not* convex along  $b_1 = b_2$ .

Algebraically, the confluent case has the form  $(c + dt_i) e^{bt_i}$ , with

$$I(b) = \sum_1^N [(c + dt_i) e^{bt_i} - y_i]^2.$$

The first order conditions for a critical point are

$$\sum_1^N [(c + dt_i) e^{bt_i} - y_i] t_i^k = 0, \quad k = 0, 1, 2.$$

If we define

$$s_k = \sum t_i^k e^{2bt_i}, \quad k = 0, 1, 2, 3,$$

$$z_k = \sum y_i t_i^k e^{bt_i}, \quad k = 0, 1, 2,$$

and solve for the linear parameters  $c$  and  $d$ , we get a single equation for  $b$ :

$$(2.6) \quad (s_2^2 - s_1 s_3) z_0 + (s_0 s_3 - s_1 s_2) z_1 = (s_0 s_2 - s_1^2) z_2.$$

This equation is rather nasty, and it appears very difficult in general to give criteria for a unique solution, so as to make  $I(b)$  convex. We *conjecture* that this will imply a unique global minimum for  $b_1 \leq b_2$ , much as in Figs. 1 and 2.

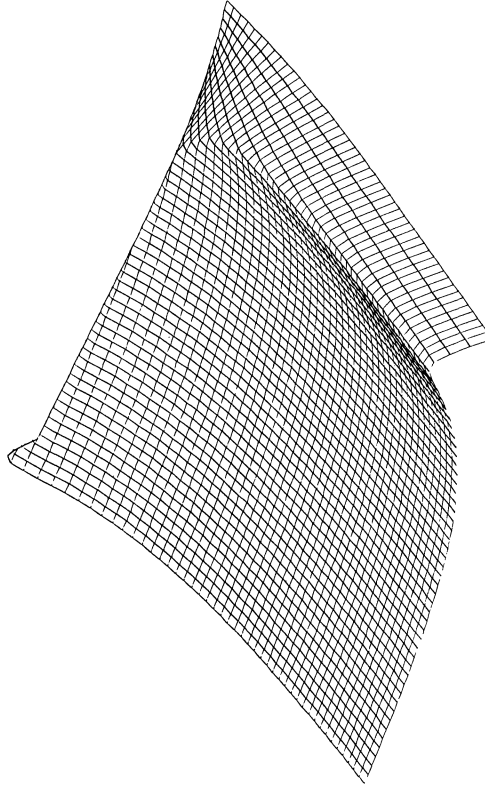


FIG. 1. Discrete least squares surface  $(I(b_1, b_2))^{1/2}$  for Osborne data.  
Range:  $-0.4 \geq b_1, b_2 \geq -7.0$ , vertical scale: (0.0, 1.0).

Now consider the special case mentioned earlier. For  $\alpha = (.1, .4, -.3)$ , we generate 33 data points from (2.5) using  $t_1 = 0$  and  $\Delta t = 0.1$ . Examination of the confluent case reveals 3 critical points, as shown in Table 4.

This case has two distinct minima; one as above at  $b_1 = b_2 = -.98$ , and the other (global) minimum at  $b_1 = -1.55, b_2 = -10.5$  with  $I(b_1, b_2) = .15 \times 10^{-3}$ . The  $I$ -function is very flat near both minima, with the  $4 \times 4$  Hessian having an eigenvalue of  $.33 \times 10^{-5}$  in the latter case. The 3-D plot of the surface  $[I(b_1, b_2)]^{1/2}$ , for  $-.4 \geq b_1, b_2 \geq -10.0$  is given in Fig. 3. In the neighborhood of the local minimum  $(-.98, -.98)$ ,  $I$  is very flat: for  $-.9 \geq b_1, b_2 \geq -1.5, 1.68 \times 10^{-3} \leq I \leq 2.02 \times 10^{-3}$ . Moreover, near the global minimum  $(-1.55, -10.5)$ , it is also flat; if we again allow .001 error in each data point, we find the uncertainty region contains  $(-1.45, -20)$  and  $(-1.7, -5.7)$ .

A similar situation occurs for very many data sets; because of this we feel that the fitting of exponentials must be attempted with great care. Moreover, there seems to be little correlation between this sensitivity and monotonicity of the data. In the following sections, we shall discuss the continuous problem in greater analytic detail.

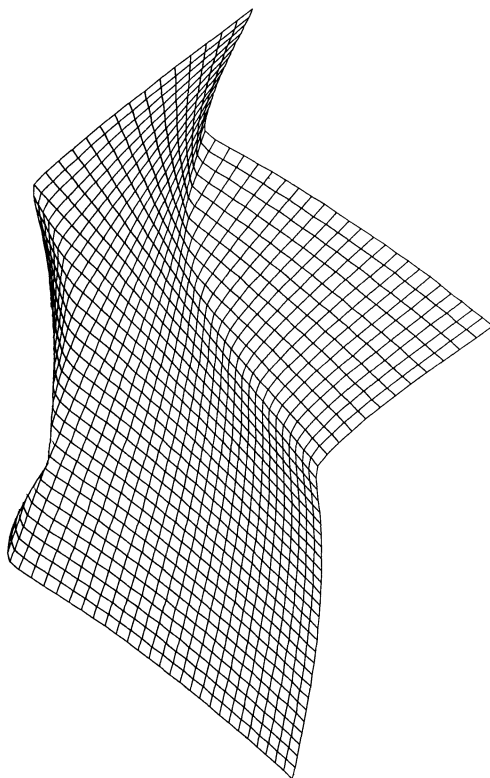


FIG. 2. Discrete least squares surface  $(I(b_1, b_2))^{1/2}$  for Lanczos data. Range:  $-0.4 \cong b_1, b_2 \cong -10.0$ , vertical scale: (0.0, 1.0).

TABLE 4

$\alpha = (.1, .4, -.3)$			
<b>b</b>	- .98	- 1.34	- 2.38
<i>I</i> (b)	$1.68 \times 10^{-3}$	$1.85 \times 10^{-3}$	$0.78 \times 10^{-3}$
nature	min	saddle	saddle

**3. The continuous problem.** Here we are given a function  $f(t)$ ,  $0 \leq t < \infty$ , and wish to approximate it by an exponential sum  $y(t) = \sum_{j=1}^m a_j e^{b_j t}$  so as to minimize

$$(3.1) \quad I(\mathbf{a}, \mathbf{b}) = \frac{1}{2} \int_0^\infty (y(t) - f(t))^2 dt.$$

We assume the exponentials are decaying, i.e.  $b_j < 0, j = 1, \dots, m$ . The first-order conditions for a minimum (or more generally for any critical point) are, for  $j = 1, \dots, m$ ,

$$\frac{\partial I}{\partial a_j} = \int_0^\infty (y - f) e^{b_j t} dt = 0$$

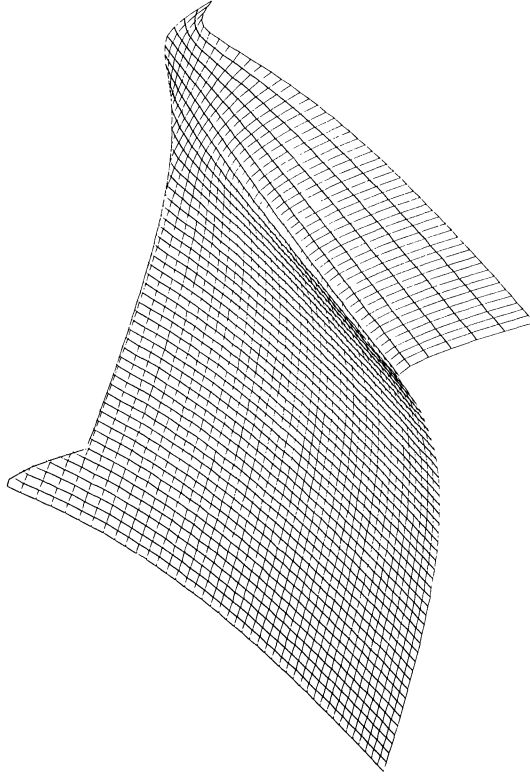


FIG. 3. Discrete least squares surface  $(I(b_1, b_2))^{1/2}$  for exponential data.  
Range:  $-0.4 \cong b_1$ ,  $b_2 \cong -10.0$ , vertical scale: (0.0, 1.0).

and

$$\frac{\partial I}{\partial b_j} = a_j \int_0^{\infty} (y-f)t e^{bt} dt = 0.$$

If we define what are essentially the Laplace transforms of  $y$  and  $f$ ,

$$z(b) = \int_0^{\infty} y(t) e^{bt} dt = -\sum_1^m \frac{a_i}{b_i + b}$$

and

$$g(b) = \int_0^{\infty} f(t) e^{bt} dt,$$

then these first-order conditions are equivalent (assuming no  $a_j = 0$ ) to the functions  $z(b)$ ,  $z'(b)$  interpolating  $g(b)$ ,  $g'(b)$  at the solution points  $\{b_j\}$ ,  $j = 1, \dots, m$ ;

$$(3.2) \quad \begin{aligned} z(b_j) &= -\sum_{i=1}^m \frac{a_i}{b_i + b_j} = \int_0^{\infty} f(t) e^{b_j t} dt = g(b_j), \\ z'(b_j) &= \sum_{i=1}^m \frac{a_i}{(b_i + b_j)^2} = \int_0^{\infty} t f(t) e^{b_j t} dt = g'(b_j). \end{aligned}$$

These equations are sometimes called the Aigrain/Williams equations (see Kammler (1979)) and of course make the error  $(y-f)$  orthogonal to  $e^{b_j t}$  and  $t e^{b_j t}$  over  $[0, \infty)$ .

They are linear in the  $\{a_j\}$ , but nonlinear in the  $\{b_j\}$ , so the existence and uniqueness of solutions is not clear, and may vary with the function  $f(t)$ .

It is of interest to compute the Hessian matrix  $H$  of second partial derivatives of  $I$ ; we get

$$\begin{aligned}\frac{\partial^2 I}{\partial a_j \partial a_k} &= \int_0^\infty e^{b_j t} e^{b_k t} dt = \frac{-1}{b_j + b_k}, \\ \frac{\partial^2 I}{\partial a_j \partial b_j} &= \int_0^\infty (y-f)t e^{b_j t} dt + a_j \int_0^\infty t e^{2b_j t} dt = 0 + \frac{a_j}{(2b_j)^2} \quad (\text{at a critical point}), \\ \frac{\partial^2 I}{\partial a_j \partial b_k} &= a_k \int_0^\infty t e^{b_j t} e^{b_k t} dt = \frac{a_k}{(b_j + b_k)^2}, \\ \frac{\partial^2 I}{\partial b_j \partial b_k} &= a_j a_k \int_0^\infty t^2 e^{b_j t} e^{b_k t} dt = \frac{-2a_j a_k}{(b_j + b_k)^3}, \\ \frac{\partial^2 I}{\partial b_j^2} &= a_j^2 \int_0^\infty t^2 e^{2b_j t} dt + a_j \int_0^\infty (y-f)t^2 e^{b_j t} dt = \frac{-2a_j^2}{(2b_j)^3} + g_j\end{aligned}$$

Thus, as in the discrete case, the Hessian consists of two parts,

$$H = H_0 + G,$$

with  $H_0$  depending explicitly on the  $\{a_j, b_j\}$  and  $G$  explicitly on  $f(t)$ . If we order the variable  $a_1, \dots, a_m, b_1, \dots, b_m$ , then  $H_0$  has the form

$$(3.3) \quad H_0 = \left( \begin{array}{c|c} \frac{-1}{b_j + b_k} & \frac{a_k}{(b_j + b_k)^2} \\ \hline \frac{a_j}{(b_j + b_k)^2} & \frac{-2a_j a_k}{(b_j + b_k)^3} \end{array} \right), \quad j, k = 1, \dots, m.$$

If we factor out the  $\{a_j\}$  by a diagonal congruency transformation, the remaining matrix is the Gram matrix for the functions  $e^{b_j t}, t e^{b_j t}, j = 1, \dots, m$ . Thus  $H_0$  is positive definite and the nature of a particular critical point depends on  $G$ . If the solution  $y(t)$  is a good fit, so that the terms  $g_j$  are small,  $H$  will be very close to  $H_0$ , and in this case the sensitivity of the solution parameters will depend effectively on the eigenvalues of  $H_0$ .

However,  $H_0$  can be *very* ill-conditioned: for  $b_j = j$ , the top left block is the Hilbert matrix of order  $m$ , and  $H_0$  is in fact much more ill-conditioned than this. Even for  $m = 2$ ,  $\lambda_1(H) < 10^{-4}$ ; for  $m = 3$ ,  $\lambda_1(H) < 10^{-7}$ , and for  $m = 4$ ,  $\lambda_1(H) < 10^{-10}$ . In practice, at least for our examples, we have found that the Hessian eigenvalues are indeed very close to those of  $H_0$  (at the minimum point), so that our problem is intrinsically very ill-conditioned. More on this can be found in Ruhe (1980).

Moreover, the situation is really much worse than this: there is the strong possibility of multiple solutions of (3.2) for a given function  $f(t)$ , even in very simple cases. We illustrate this by extending an example of Kammler (1979), fitting two exponentials by one,

$$(3.4) \quad f(t) = e^{-t} + \alpha e^{bt}, \quad y(t) = a e^{bt}.$$

We assume  $\alpha$  and  $\beta$  are given,  $a$  and  $b$  are to be found, and that  $b$  and  $\beta$  are negative. The equations (3.2) are as follows:

$$(3.5) \quad \begin{aligned} \frac{\partial I}{\partial a} = 0 &\Rightarrow \frac{a}{2b} = \frac{1}{b-1} + \frac{\alpha}{b+\beta}, \\ \frac{\partial I}{\partial b} = 0 &\Rightarrow a = 0, \quad \text{or} \\ \frac{a}{4b^2} &= \frac{1}{(b-1)^2} + \frac{\alpha}{(b+\beta)^2}. \end{aligned}$$

However, using (3.4) gives

$$(3.6) \quad I = \frac{a^2}{2b} + \frac{1}{2} - \frac{\alpha^2}{2\beta} - \frac{2\alpha}{\beta-1}.$$

So  $I$  is always smaller if  $a \neq 0$ , i.e.  $a = 0$  never gives a minimum. Using (3.4) to define  $a$ , and substituting in (3.5) gives

$$(3.7) \quad (b+1)(b+\beta)^2 + \alpha(b-\beta)(b-1)^2 = 0$$

which is a cubic for  $b = b(\alpha, \beta)$  with one or three real solutions. To see when three real solutions can exist, express  $I$  as a function of  $b$  alone (using (3.4)):

$$(3.8) \quad I(b) = 2b \left( \frac{1}{b-1} + \frac{\alpha}{b+\beta} \right)^2 + \frac{1}{2} - \frac{\alpha^2}{2\beta} - \frac{2\alpha}{\beta-1}.$$

Now  $dI/db = 0$  gives  $a = 0$  or (3.7) as above; however at solutions of (3.7)

$$\frac{d^2 I}{db^2} = \frac{4(\beta+1)^2 b}{(b-\beta)^2 (b-1)^5 (b+\beta)} [3b^2 + b - b\beta - 3\beta]$$

whose *sign* is completely determined by the quantity in square brackets. A plot of this in the  $(b, \beta)$ -plane is shown in Fig. 4.

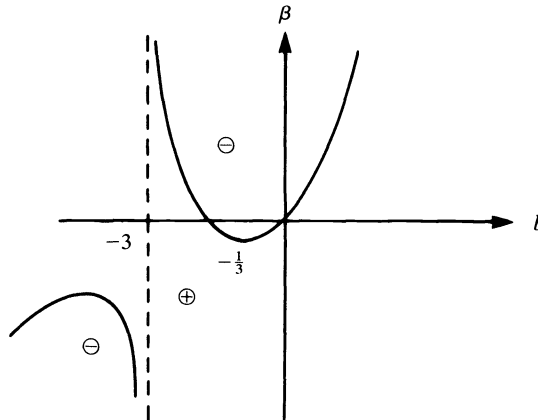


FIG. 4

Notice that the only chance for having three real roots is for  $(b, \beta)$  to be in the  $\ominus$  region (i.e. with  $I''(b) < 0$ ) for one of the roots  $b$ , so that a local maximum is achieved. Now consider  $b, \beta$  as our free parameters (not  $\alpha, \beta$ ) with  $\alpha$  given by (3.7). We can restrict our attention to the small  $\ominus$  region  $R$  near zero and below the  $b = 0$

axis; the other  $\ominus$  region is obtained by a simple transformation:  $b \rightarrow 1/b$ ,  $\beta \rightarrow 1/\beta$ ;  $\alpha \rightarrow \alpha/\beta$ , the other  $b$ -roots  $\rightarrow$  reciprocals,  $a \rightarrow -a/b$ , and  $I \rightarrow I$ . That is, for each  $(b, \beta)$  pair in  $R$ , there is a reciprocal pair  $(1/b, 1/\beta)$  with the same solution. However in  $R$ , we have *three*  $b$ -solutions; there is one  $b$  for which  $I''(b) < 0$ , i.e. a local maximum. However we must have two other local minima (say  $x_1, x_2$ ) with  $-\infty < x_1 < b < x_2 < 0$ , since  $I'(b) > 0$  for  $b \rightarrow 0^-$ , and  $I'(b) \rightarrow 0^-$  for  $b \rightarrow -\infty$ . Actually, this holds for interior points of  $R$  only; on the boundary, we get a double root (say  $x_1 = b$ ) and even a triple root at the minimum of the  $b - \beta$  curve (i.e.  $b = \sqrt{8-3}$ ).

Of particular interest are cases where the two local minima  $x_1, x_2$  have identical  $I$ -values, so we in fact have a nonunique *global* minimum. This occurs for a continuum of values  $(b, \beta)$  inside  $R$  given by  $\beta = -b^2$  (notice this curve leaves  $R$  at the minimum  $b = \sqrt{8-3}$ ). On this curve, the other  $b$ -roots  $x_1, x_2$  are given by (for  $\sqrt{8-3} < b < 0$ )

$$x^2 + (b^2 + 4b + 1)x + b^2 = 0$$

or

$$x_{1,2} = \frac{-(b^2 + 4b + 1) \pm (b + 1)\sqrt{b^2 + 6b + 1}}{2}.$$

At these  $x$ -values, the corresponding  $a$ -values from (3.4) are

$$a(x) = 2x \left( \frac{1}{x-1} - \frac{b}{x-b^2} \right) = 2 \left( \frac{1}{1-1/x} - \frac{b}{1-b^2/x} \right).$$

Notice that if  $x \rightarrow b^2/x$ ,  $a(x) \rightarrow a(b^2/x) = 2b(1/(x-1) - b/(x-b^2)) = (b/x)a(x)$ , and hence

$$\frac{a^2(x)}{2x} = \frac{a(x)a(b^2/x)}{2b} = \frac{a^2(b^2/x)}{2(b^2/x)}.$$

Thus since  $x_2 = b^2/x_1$  for the two roots, we have  $a^2(x_1)/2x_1 = a^2(x_2)/2x_2$  and thus from (3.6),  $I(x_1) = I(x_2)$ .

Thus we can have multiple solutions and nonunique minima even in this relatively simple case; as Kammler (1979) notes, such problems are extremely ill-conditioned numerically, particularly for  $b$  near the triple root.

From now on, we again specialize to the case of two exponentials, i.e.  $y(t) = a_1 e^{b_1 t} + a_2 e^{b_2 t}$ . Then the variational equations are (see (3.2))

$$(3.9) \quad \begin{aligned} \frac{-a_1}{2b_1} + \frac{-a_2}{b_1 + b_2} &= \mathbf{g}(b_1) \equiv g_1, & \frac{-a_1}{b_1 + b_2} + \frac{-a_2}{2b_2} &= \mathbf{g}(b_2) \equiv g_2, \\ \frac{a_1}{4b_1^2} + \frac{a_2}{(b_1 + b_2)^2} &= \mathbf{g}'(b_1) \equiv g'_1, & \frac{a_1}{(b_1 + b_2)} + \frac{a_2}{4b_2^2} &= \mathbf{g}'(b_2) \equiv g'_2. \end{aligned}$$

The first two equations define  $a_1, a_2$  via  $B\mathbf{a} = \mathbf{g}$ , with

$$B = \begin{pmatrix} -1 & -1 \\ 2b_1 & b_1 + b_2 \\ -1 & -1 \\ b + b_2 & 2b_2 \end{pmatrix}.$$

Then the remaining two are

$$(3.10) \quad \begin{aligned} \left( \frac{1}{2b_1} - \frac{2b_2}{b_1^2 - b_2^2} \right) g_1 + \frac{b_2}{b_1(b_1 - b_2)} g_2 &= -g'_1, \\ \frac{-b_1}{b_2(b_1 - b_2)} g_1 + \left( \frac{1}{2b_2} + \frac{2b_1}{b_1^2 - b_2^2} \right) g_2 &= -g'_2. \end{aligned}$$

Moreover, the functional

$$I(a(b), b) \equiv \int_0^\infty (y-f)^2 dt = \int_0^\infty f^2 dt - \mathbf{g}^T \mathbf{B}^{-1} \mathbf{g}$$

which after some manipulation can be expressed as

$$(3.11) \quad I = \int_0^\infty f^2 dt + 2(b_1 + b_2) \left[ \left( \frac{b_1 g_1 - b_2 g_2}{b_1 - b_2} \right)^2 + b_1 b_2 \left( \frac{g_1 - g_2}{b_1 - b_2} \right)^2 \right].$$

This last form is particularly useful, as it holds in the *confluent* case ( $b_1 = b_2$ ) as well if limits are used. If  $b_1 = b_2 = b$ , then  $y(t) = (ct + d) e^{bt}$ , and the variational equations give

$$c = -4b^2(g + 2bg'), \quad d = -4b(g + bg')$$

where  $g = g(b)$ ,  $g' = g'(b)$  as before. Then in terms of  $b$  only,

$$(3.12) \quad I = \int_0^\infty f^2 dt + 4bg^2 + 8b^2gg' + 8b^3(g')^2$$

which indeed is the limit of (3.11) as  $b_2 \rightarrow b_1 = b$ . The conditions for a critical point in this case boil down to one equation:

$$I'(b) = 4[g^2 + 6bgg' + 8b^2(g')^2 + 2b^2gg'' + 4b^3g'g''] = 0.$$

Because of the symmetry in  $I(b_1, b_2)$  across  $b_1 = b_2$ , any solutions of this equation are critical points of  $I$  in the  $(b_1 - b_2)$ -plane, and may be local minima, maxima, or saddle points.

In principle, for a given function  $f(t)$ , equations (3.10) could be used to find solutions and (3.11) could be differentiated to find the Hessian. However this appears to be difficult in any specific practical case, and we prefer to try to understand the problem by plotting the surface  $I(b_1, b_2)$ .

As an example, consider  $f(t) = t^2 e^{-t}$ . Then

$$g(b) = \frac{2}{(1-b)^3}, \quad g'(b) = \frac{6}{(1-b)^4}, \quad g''(b) = \frac{24}{(1-b)^5}.$$

The variational equations (3.10) are difficult to analyze; however the confluent case  $b_1 = b_2 = b$  is somewhat easier:

$$I(b) = \frac{3}{4} + 4b \frac{4}{(1-b)^6} + 8b^2 \frac{12}{(1-b)^7} + 8b^3 \frac{36}{(1-b)^8}$$

and it is easy to check that  $I'(b) = 0$  at three points:

$$b^{(1)} = \frac{-5 + \sqrt{12}}{13} \cong -.118, \quad I(b^{(1)}) \cong .202 \quad (\text{local minimum}),$$

$$b^{(2)} = -0.2, \quad I(b^{(2)}) \cong .214 \quad (\text{saddle point}),$$



$$b^{(3)} = \frac{-5 - \sqrt{12}}{13} \cong -.651, \quad I(b^{(3)}) \cong .0125 \quad (\text{global minimum}).$$

There appear to be no critical points for  $b_1 \neq b_2$ , and we plot the surface  $[I(b_1, b_2)]^{1/2}$  in Fig. 5, with  $-.05 \cong b_1, b_2 \cong -2.0$ .

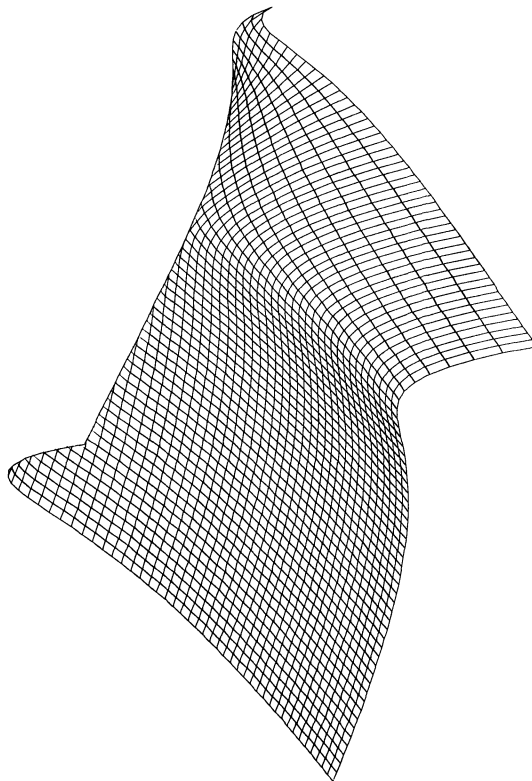


FIG. 5. Continuous least squares surface  $(I(b_1, b_2))^{1/2}$  for  $f(t) = t^2 e^{-t}$ .  
Range:  $-.05 \cong b_1, b_2 \cong -2.0$ , vertical scale:  $(0.0, 1.0)$ .

Other choices of  $f(t)$  will give very different surfaces, of course, and in the next section we consider the special case where  $f(t)$  is itself an exponential sum.

**4. Exponential data.** Here we consider fitting two exponentials  $y(t) = a_1 e^{b_1 t} + a_2 e^{b_2 t}$  to  $f(t) = \sum_{j=1}^n \alpha_j e^{\beta_j t}$  as we did in the discrete case in § 2. Thus  $g(b) = \sum_1^n -\alpha_j / (b + \beta_j)$  and the variational equations (3.10) can, after some manipulation, be written

$$(4.1) \quad \begin{aligned} \sum_{i=1}^n \alpha_j \frac{(b_1 - \beta_i)(b_2 - \beta_i)}{(b_1 + \beta_i)^2 (b_2 + \beta_i)} &= 0, \\ \sum_{j=1}^n \alpha_j \frac{(b_1 - \beta_i)(b_2 - \beta_i)}{(b_1 + \beta_i)(b_2 + \beta_i)^2} &= 0. \end{aligned}$$

Clearly, some results about the location of roots  $(b_1, b_2)$  can be inferred from (4.1). For example, if the  $\alpha_j > 0$  for all  $j$ , and  $\beta_n < \dots < \beta_1 < 0$ , then at least one of  $(b_1, b_2)$  must lie inside  $(\beta_n, \beta_1)$ .

However, it is very difficult to give any more general results about the nature of the solutions  $(b_1, b_2)$ : even if the  $\beta_j$  are fixed, the nature of the surface  $I(b_1, b_2)$  varies

tremendously with the choice of  $\alpha = (\alpha_1, \dots, \alpha_n)$ . From (3.11), we can express  $I(b_1, b_2)$  as

$$(4.2) \quad I = \alpha^T C \alpha - \mathbf{g}^T B^{-1} \mathbf{g} = \alpha^T A \alpha, \quad c_{ij} = \frac{-1}{\beta_i + \beta_j}$$

with  $A = DCD$ , and  $D$  diagonal with

$$d_i = \frac{(b_1 - \beta_i)(b_2 - \beta_i)}{(b_1 + \beta_i)(b_2 + \beta_i)}$$

Moreover, we can express the Hessian similarly:

$$H = \begin{pmatrix} \alpha^T \frac{\partial^2 A}{\partial b_1^2} \alpha & \alpha^T \frac{\partial^2 A}{\partial b_1 \partial b_2} \alpha \\ \alpha^T \frac{\partial^2 A}{\partial b_1 \partial b_2} \alpha & \alpha^T \frac{\partial^2 A}{\partial b_2^2} \alpha \end{pmatrix}.$$

One can explicitly compute the partial derivatives; indeed,

$$\frac{\partial^2 A}{\partial b_1^2} = 4b_1^3(\bar{D}G\bar{D})$$

where  $\bar{D}$  is diagonal,

$$\bar{d}_i = \frac{b_2 - \beta_i}{(b_2 + \beta_i)(b_1 + \beta_i)^3},$$

and

$$g_{ij} = 1 - 3 \frac{\beta_i \beta_j}{b_1^2} + \frac{\beta_i \beta_j}{b_1^2} \left( \frac{\beta_i}{b_1} + \frac{\beta_j}{b_1} \right).$$

However, to show uniqueness of a minimum point,  $(b_1, b_2)$  for example, we need convexity of  $I(b_1, b_2)$ , i.e.  $H$  positive definite for all  $b_1 < 0, b_2 < 0$ , for some particular choice of  $\alpha$  and  $\beta$ . This seems to be very difficult: the matrix  $G$  is unfortunately indefinite over much of the region  $b_1 < 0$ .

Even the confluent case is intractable, although interesting: one can readily express the function  $I(b)$  from (4.2) and its derivatives; for example

$$\frac{dI}{db} = -4\alpha^T \hat{D} \hat{G} \hat{D} \alpha,$$

with  $D$  diagonal,  $\hat{d}_i = (b - \beta_i)/(b + \beta_i)^3$ , and  $\hat{g}_{ij} = b^2 - \beta_i \beta_j$ . Again, however, the nature of the function varies tremendously with  $\alpha$ . For some  $\alpha$ ,  $dI/db = 0$  at only one point (a minimum), and for these it appears that  $I(b_1, b_2)$  has a unique minimum as well; for others however, the confluent case admits 3 or more solutions and the full function  $I(b_1, b_2)$  can have several minima. As well,  $I(b_1, b_2)$  can be very flat over a large range of  $b_1, b_2$ . We can illustrate these different aspects with examples, all taken with  $n = 3$  and  $\beta_1 = -1, \beta_2 = -3, \beta_3 = -5$ .

*Example 1 (the Lanczos data).*  $\alpha = (.0951, .8607, 1.5576)$ .

Here the confluent case has one minimum, which is a saddle point for the full function  $I(b_1, b_2)$ . This in turn has a unique minimum for  $b_1 \cong b_2$  at  $(-1.47, -4.42)$  with  $I = .85 \times 10^{-5}$  and  $\lambda_{\min}(H) = .0021$ . The surface is very similar to that of the discrete problem, given in Fig. 2.

*Example 2.*  $\alpha = (.1, .4, -.3)$ .

Here (as in the discrete problem) the confluent case has three critical points, one a local minimum and two saddle points for the full problem. In addition, the full problem has a minimum for  $b_1 > b_2$ . The minima are:

$$(-1.17, -1.17), \quad I = .99 \times 10^{-4}$$

$$(-1.54, -11.2), \quad I = .18 \times 10^{-4}, \quad \lambda_{\min}(H) = .48 \times 10^{-6}.$$

Again the surface is similar to that in Fig. 3.

Notice also that this surface is very flat: indeed for the whole region  $-1 \cong b_1, b_2 \cong -3$ , we have

$$.06 \times 10^{-3} \cong I(b_1, b_2) \cong .25 \times 10^{-3}.$$

*Example 3.*  $\alpha = (.14, -.70, .70)$ .

Here the confluent case has 5 critical points, listed in Table 5.

TABLE 5

<b>b</b>	-0.20	-0.33	-0.73	-2.11	-7.2
$I(\mathbf{b})$	$.1165 \times 10^{-2}$	$.1174 \times 10^{-2}$	$.1124 \times 10^{-2}$	$.1456 \times 10^{-2}$	$.078 \times 10^{-2}$
nature	minimum	saddle	minimum	maximum	minimum

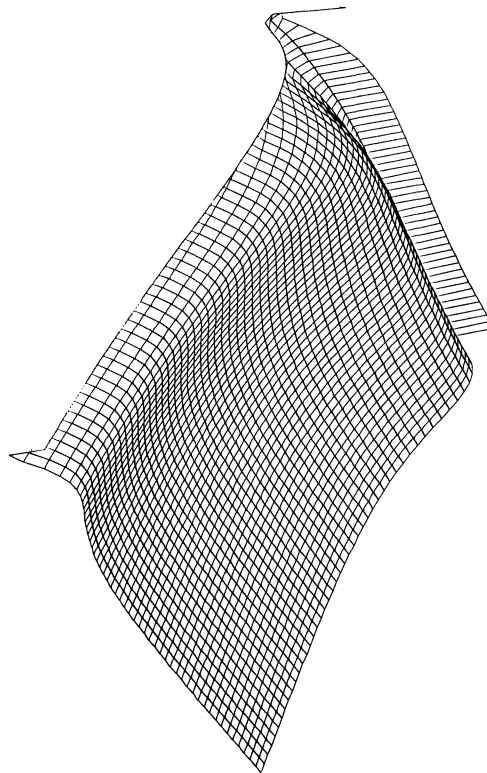


FIG. 6. Continuous least squares surface  $(I(b_1, b_2))^{1/2}$  for  $f(t) = \exp$  sum. Range:  $-0.1 \cong b_1, b_2 \cong -10.0$ , vertical scale:  $(0.0, 0.12)$ .

The surface is incredibly flat in this case; over the whole range  $-0.1 \cong b_1, b_2 \cong -10.0$ ,  $.078 \times 10^{-2} \cong I(b_1, b_2) \cong .146 \times 10^{-2}$ . Thus if the surface is scaled like the others (where  $\max I(b_1, b_2) \cong 1.0$ ), it would appear totally flat. Scaled up however, it is much more interesting: see Fig. 6. The three local minima from the confluent case  $b_1 = b_2$  appear to be the only minima; however the flatness indicates that the original fitting problem can be solved by using any values of  $b_1$  and  $b_2$  in this range. Notice that for this problem  $\alpha$  is very nearly the eigenvector corresponding to the smallest eigenvalue of  $A$  in (4.2).

**5. Conclusions.** Although we have emphasized the possibility of ill-conditioning in exponential fitting, we do not mean to imply that every such problem is ill-conditioned, or admits multiple solutions. If the best fit has powers  $\{b_j\}$  which are widely separated, then the Hessian matrix  $H$  of § 3 can be well-conditioned and the corresponding least squares surface convex and not flat.

However it is exceedingly difficult to judge this from the data: because of the nonlinearity of the problem, the condition will depend on the data and on the solution generated. Thus we recommend that when solving such problems, the eigenvalues of the Hessian near the solution be calculated, at the very least, and if possible, a plot of the least squares surface (projected into two-dimensional planes for more than two exponents) be computed as well, to give a more global picture of the possible ill-condition of the problem.

#### REFERENCES

- Y. BARD (1974), *Nonlinear Parameter Estimation*, Academic Press, New York.
- D. M. BATES AND D. G. WATTS (1980), *Relative curvature measures of nonlinearity*, J. Royal Stat. Soc., Ser. B, 42, pp. 1-25.
- J. W. EVANS, W. B. GRAGG, AND R. J. LE VEQUE (1980), *On least squares exponential sum approximation with positive coefficients*, Math. Comp., 34, pp. 203-211.
- G. H. GOLUB AND V. PEREYRA, *The differentiation of pseudoinverses and nonlinear least squares problems whose variables separate*, SIAM J. Numer. Anal., 10, pp. 413-432.
- D. W. KAMMLER (1979), *Least squares approximation of completely monotonic functions by sums of exponentials*, SIAM J. Numer. Anal., 16, pp. 801-818.
- L. KAUFMAN (1978), *A program for solving separable nonlinear least squares problems*, Bell Labs. Technical Memo 78-1274-7.
- C. LANCZOS (1956), *Applied Analysis*, Prentice-Hall, Englewood Cliffs, NJ, 1956.
- M. R. OSBORNE (1972), *Some aspects of nonlinear least squares calculations*, in Numerical Methods for Nonlinear Optimization, F. A. Lootsma, ed., Academic Press, New York.
- (1975), *Some special nonlinear least squares problems*, SIAM J. Numer. Anal., 12, pp. 571-592.
- H. RAMSIN AND P.-A. WEDIN (1977), *A comparison of some algorithms for the nonlinear least squares problem*, BIT, 17, pp. 72-90.
- A. RUHE (1980), *Fitting empirical data by positive sums of exponentials*, this Journal, 1, pp. 481-498.
- J. M. VARAH (1982), *A spline least squares method for numerical parameter estimation in differential equations*, this Journal, 2, pp. 28-46.

## SYMMETRIC VERSUS NONSYMMETRIC DIFFERENCING\*

WAYNE R. DYKSEN† AND JOHN R. RICE‡

**Abstract.** Consider the self-adjoint elliptic problem  $(pu_x)_x + (qu_y)_y + ru = f$  with Dirichlet boundary conditions on the unit square. This problem is symmetric in the sense that if the data is symmetric then so is the solution. The straightforward finite difference discretization has one expand the derivatives and apply differences to  $pu_{xx} + p_x u_x + \dots$ . Alternatively there are symmetric discretizations which are attractive intuitively and which are usually recommended. We have observed that symmetric discretizations are sometimes much less accurate; a simple analysis is made to compare the expected behavior of the two discretizations. Data from a simplified model problem confirms the expectations that nonsymmetric differences are more accurate than symmetric differences much more often than vice versa. We conclude for elliptic problems that unless it is known that  $u$  varies much more rapidly than  $p$  and  $q$ , one should use nonsymmetric differences.

**Key words.** finite difference, elliptic differential equations, symmetric discretizations, experimental study

**1. The difference approximations.** We have observed differences in the discretization errors of finite difference approximations for several elliptic problems. These were large enough to completely change the expected outcomes of substantial software performance evaluations. We believe the underlying phenomenon has nothing to do with the dimensionality, so we restrict ourselves to one-dimensional problems here.

Consider the finite difference discretizations of the terms

$$(pu_x)_x \quad \text{and} \quad pu_{xx} + p_x u_x.$$

One introduces a grid  $x_i = ih$  for  $0 \leq i \leq n+1 = 1/h$ , and uses the variables  $u_i$  to approximate  $u(x_i)$  and  $p_i = p(x_i)$ . The symmetric discretization of  $(pu_x)_x$  at  $x_i$  is

$$\frac{[p_{i-1/2}u_{i-1} - (p_{i-1/2} + p_{i+1/2})u_i + p_{i+1/2}u_{i+1}]}{h^2} + \frac{h^2[(pu''')'_i + (pu')''_i]}{24} + O(h^4),$$

where primes indicate differentiation with respect to  $x$ . The nonsymmetric discretization of  $pu_{xx} + p_x u_x$  at  $x_i$  is

$$\left[ \frac{p_i}{h^2} - \frac{p'_i}{2h} \right] u_{i-1} - \left[ \frac{2p_i}{h^2} \right] u_i + \left[ \frac{p_i}{h^2} + \frac{p'_i}{2h} \right] u_{i+1} + \frac{h^2}{3} \left[ \frac{p_i u_i^{(iv)}}{4} + \frac{p'_i u_i'''}{2} \right] + O(h^4).$$

Note that  $p'(x_i)$  is computed symbolically in the nonsymmetric discretization.

The error terms of these two approximations are substantially different and it is clear that one can construct problems (chose  $p(x)$  and  $u(x)$ ) so that either approximation is much more accurate than the other. Intuitively, one would expect the nonsymmetric difference to be more accurate when  $p(x)$  is rapidly varying because the derivative  $p_x$  is computed symbolically. This is indicated also by the presence of the third derivative of  $p(x)$  in the leading error term for the symmetric difference while the nonsymmetric difference has only the first derivative of  $p(x)$  in the leading error term. On the other hand, if the derivative  $p_x$  cannot be computed symbolically (say  $p$  is piecewise constant) and the product  $pu_x$  is smooth, then one would expect the symmetric difference to be more accurate.

---

\* Received by the editors December 7, 1982. This work was supported in part by Department of Energy contract DE-AC02-81ER10997.

† Division of Mathematical Sciences, Purdue University, West Lafayette, Indiana 47907.

**2. An experimental study.** To study experimentally the nature of the situation, we consider the one-dimensional model problem

$$-(p(x)u_x)_x = f \text{ in } [0, 1], \quad u(0), u(1) \text{ given.}$$

The function  $f(x)$  is chosen to make the model problem solution be as specified. We choose ten functions for  $u(x)$ :

$$x^2, x^4, e^x, \sin x, \frac{1}{1+x^2}, e^{10x}, \frac{1}{1+10x^2}, \sin 10x, \sin 100x, x^{10}$$

and ten functions for  $p(x)$ :

$$1, x, x^2, x^4, e^x, \sin x, \frac{1}{1+x^2}, e^{10x}, 1.1 + \sin 100x, \frac{1}{1+10x^2}.$$

Then all 100 combinations of elliptic problems are solved. We compute the maximum relative errors  $e_N$  and  $e_S$  of the nonsymmetric and symmetric differences, respectively. The results are tabulated in the following manner. A factor  $R$  is chosen, the two discretizations are said to tie if either

$$\frac{\max(e_N, e_S)}{\min(e_N, e_S)} \leq R \text{ or } \max(e_N, e_S) \leq \text{round-off.}$$

The computation is made on a VAX 11/780 (6 decimal digit arithmetic) and the round-off level is determined from those cases where the discretization is theoretically exact. Table 1 has four arrays with entry “-” if the methods tie, “N” if the nonsymmetric

TABLE 1

*Arrays showing the error performance of the two discretizations. The columns correspond to the ten  $p(x)$  functions, the rows to the ten  $u(x)$  functions. A dash means the discretizations tie, N and S mean that nonsymmetric and symmetric are better, respectively.*

-	-	N	N	N	N	N	N	N	N	-	-	N	N	N	N	N	N	N	N
-	-	-	N	-	-	-	N	N	S	-	-	-	-	-	-	-	-	N	-
-	-	N	N	N	S	-	N	N	N	-	-	-	N	-	-	-	-	N	N
-	-	N	N	-	-	N	N	N	N	-	-	-	N	-	-	-	-	N	N
-	-	-	N	-	-	-	N	N	-	-	-	-	-	-	-	-	-	-	N
-	-	-	-	-	-	-	N	N	-	-	-	-	-	-	-	-	-	-	N
-	-	S	N	-	-	-	-	N	-	-	-	-	-	-	-	-	-	-	N
-	-	S	-	-	-	-	-	N	-	-	-	-	-	-	-	-	-	-	N
-	-	-	-	-	-	-	-	S	-	-	-	-	-	-	-	-	-	-	-
-	-	-	-	-	-	-	N	N	-	-	-	-	-	-	-	-	-	-	N
33 N's 5 S's 62 ties $h = \frac{1}{20}$ $R = 1.4$										20 N's 0 S's 80 ties $h = \frac{1}{20}$ $R = 10.0$									
-	-	N	N	N	N	N	N	N	N	-	N	N	N	-	N	N	N	N	N
-	-	-	-	-	-	-	N	N	S	-	-	-	-	-	-	-	-	-	N
-	-	-	N	-	-	-	N	N	-	-	-	-	N	-	N	N	N	N	N
-	-	-	N	-	-	-	N	N	-	-	-	-	N	-	-	N	N	N	-
-	-	-	N	-	-	-	N	N	-	-	-	-	-	-	-	-	-	-	N
-	-	-	-	-	-	-	-	N	-	-	-	-	-	-	-	-	-	-	N
-	-	S	-	-	-	-	-	N	-	-	-	-	-	-	-	-	-	-	N
-	-	-	-	-	-	-	-	N	-	-	-	-	N	-	-	-	-	-	-
-	-	-	-	-	-	-	-	S	-	-	-	-	-	-	-	-	-	-	S
-	-	-	-	-	-	-	-	N	-	-	-	-	-	-	-	-	-	-	N
23 N's 3 S's 74 ties $h = \frac{1}{20}$ $R = 4.0$										24 N's 1 S 76 ties $h = \frac{1}{100}$ $R = 10.0$									

error  $e_N$  is smaller and there is no tie, “S” if the symmetric error  $e_S$  is smaller and there is no tie. Data are given for  $h = \frac{1}{20}$  with  $R = 1.4, 4.0$  and  $10.0$  and for  $h = \frac{1}{100}$  with  $R = 10.0$ .

The main observation to be made is that most of the time (at least  $\frac{2}{3}$ ) it does not make any difference which discretization is used. It is probably “fairer” to exclude the first row of the arrays where  $u = x^2$  and the nonsymmetric difference is exact. However, the general conclusion is unchanged if this is done. In those cases where it does make a difference which discretization is used, the nonsymmetric one is much more likely to be the best and often by a substantial amount. The eight largest differences in discretization errors are shown in Table 2 for  $h = \frac{1}{20}$  (excluding the case  $u = x^2$ ).

TABLE 2

$e_N$	$e_S$	$u$	$p$	$e_S/e_N$
0.0052	3.7	$x^4$	$1.1 + \sin 100x$	712
0.00021	0.69	$e^x$	$1.1 + \sin 100x$	3,286
0.072	5.8	$e^{10x}$	$1.1 + \sin 100x$	81
0.013	1.9	$1/(1 + 10x^2)$	$1.1 + \sin 100x$	146
0.076	16.0	$\sin 10x$	$1.1 + \sin 100x$	211
0.067	6.5	$x^{10}$	$1.1 + \sin 100x$	97
0.00039	0.011	$e^x$	$x^4$	28
0.00064	0.023	$\sin x$	$x^4$	36

We have made an additional study with  $p(x)$  functions that represent sharp “wave fronts”, a case where the symmetric differences are thought to be particularly good. We chose three functions for  $p(x)$  as follows

$$\frac{(x-.4)^2}{.001+(x-.4)^2}, \quad \frac{(x-.4)^2}{.02+(x-4)^2}, \quad \frac{(x-.4)^2}{.0001+(x-.4)^2}$$

and repeated the experiment using the same 10 true solutions  $u(x)$ . With the notation of Table 1 (with rows and columns interchanged), the results are as shown in Table 3,

TABLE 3

$N$	-	$N$	$N$	$N$	-	$N$	$N$	-	-
$N$	-	$N$	-	$N$	-	-	-	-	-
$N$	$N$	$N$	$N$	$N$	$N$	$N$	$N$	-	-
17 $N$ 's   0 $S$ 's   13 ties $h = \frac{1}{20}$ $R = 4.0$									

We observe that the nonsymmetric differences are more advantageous for this case than in the general experiment. In many cases the accuracy advantage of the nonsymmetric differences is quite large; for  $R = 10, 100$  and  $1,000$  the number of  $N$ 's in Table 3 is 15, 8 and 5, respectively.

These data give strong experimental support to the conclusion reached by the analysis: *Unless it is known that  $u$  varies much more rapidly than  $p$ , one should use the nonsymmetric differences in order to obtain the best accuracy from the discretization.*

**3. Preservation of side conditions.** It is sometimes the case that the solution of the physical problem and its mathematical model has properties which are not preserved in a numerical model derived from the mathematical model. In the present instance, the nonsymmetric differencing does not preserve symmetry or, in fluid flow applications,

the conservation of mass. The conservation of mass is important for numerical stability near shock waves, but shock waves are very rarely present in elliptic problems. In fact, the motivation for deriving the symmetric formulas in the first place was to preserve side conditions. Many people feel that forcing the numerical model to preserve these conditions will improve the accuracy of the results. We see that this is not the case here and we believe that, in general, one should expect satisfaction of constraints to be achieved at the price of reduced accuracy.

We note that sometimes side conditions can be used in a different way with considerable advantage. A simple example of this is symmetry; if the solution is known to be symmetric in some sense, then it is fairly easy to take the solution from the numerical method and enforce the symmetry. One would do this with a minimal perturbation and achieve two things: (a) a solution which satisfies the side condition and (b) a true lower bound on the numerical error. The lower bound is the amount of perturbation required to enforce the side conditions. For the simple model studied here we have experimentally examined the usefulness of this error estimator and the results are summarised in Table 4. As before, 100 problems were created and solved; their solutions are symmetric about  $x=0.5$ . We only consider results where the numerical error is above round-off;  $10^{-5}$  in this instance. Of the 100 problems considered, this round-off criterion gives 78 and 93 problems for nonsymmetric and symmetric differences, respectively. Thus we see again the superior accuracy of the nonsymmetric differences, though it is not well quantified here.

TABLE 4  
*Frequency of  $e$  being contained in intervals based on  $t$  for symmetric and nonsymmetric differences.*

interval	symmetric	nonsymmetric
$[t/2, 2t)$	50	39
$[2t, 4t)$	12	11
$[4t, 10t)$	12	11
$[10t, 100t)$	12	9
$[100t, \infty)$	7	8
totals	93	78

Let  $e$  denote the maximum relative error on the grid in solving a problem with  $h=0.01$  and  $t$  denote the maximum relative difference in symmetry on the grid. Then we have  $e \geq t/2$ . Table 4 shows the number of instances that  $e$  is contained in each of five intervals related to  $t$ . The first one is where  $t/2$  is an excellent estimate of  $e$  and the last is where  $t/2$  is a grossly misleading estimate. We see that for both difference formulas that  $t$  is very close to  $e$  about half the time and is a reasonable estimate of  $e$  about 80% of the time. We consider this to be remarkably reliable considering the simplicity and computational cost involved. Of course, other side conditions might well not be simple or cheap to use in this way.



## MULTI-GRID SOLUTION OF BIFURCATION PROBLEMS\*

H. D. MITTELMANN† AND H. WEBER‡

**Abstract.** A continuation strategy is developed for a method of inverse iteration type to compute solutions of nonlinear eigenvalue problems. The solution curves are parametrized by a norm of the solution. Thus continuation near turning points does not represent any difficulties. A multi-grid version of the algorithm is presented and successfully applied to solve problems on general two-dimensional domains. Numerical results including a comparison with a more standard approach are also given for the continuation strategy proposed for the basic algorithm.

**Key words.** nonlinear eigenvalue problem, bifurcation, turning point, continuation, inverse iteration, multi-grid method

**Introduction.** Nonlinear boundary value problems with several solutions and exhibiting bifurcation phenomena have only recently been treated with the numerical techniques which have been quite successful in the solution of boundary value problems; see, for example, [1]–[5], [7], [10]–[12], [16]. Preconditioned conjugate gradient (PCCG) methods as well as multi-grid (MG) algorithms have been used extensively for linear problems and both have been applied to nonlinear problems, too, either after a linearization by, for example, Newton's method or in a suitable nonlinear version.

For the treatment of parameter-dependent nonlinear boundary value problems quite a few approaches have been proposed and used successfully; see, for example, [6], [9], [14]. Since there is in general a sequence of linear or nonlinear problems which has to be solved it is quite natural to combine these methods with the efficient PCCG and MG algorithms. Here we present such a combination. The underlying method is not classical, but it is the generalized inverse iteration of [10] for which a first combination with multi-grid ideas has been proposed in [11]. This method is not as general as, for example, pseudo-arclength continuation [6] and in its original form is applicable to a certain class of nonlinear eigenvalue problems. For this class, however, it proves to be a very robust and efficient method and it may also be generalized.

In the following we shall first describe the classes of problems treated, define the generalized inverse iteration and state a local convergence result. We propose then a simple but efficient way to use this algorithm for continuation purposes, i.e. for following the solution curves of a discrete nonlinear eigenvalue problem. Then a MG version of the method is given which is a generalization of the two-level method of [11], [12]. Numerical results for the continuation strategy are compared to those obtained with the pseudo-arclength method as implemented in [1]. Finally, some experiments with the MG algorithm are reported.

The contents of the following sections are:

1. Nonlinear eigenvalue problems
2. The basic algorithm
3. The continuation method
4. A multi-grid algorithm
5. Extensions
6. Continuation results
7. Multi-grid results

---

\* Received by the editors April 11, 1983, and in revised form September 23, 1983.

† Abteilung Mathematik, Universität Dortmund, Postfach 500500, 4600 Dortmund 50, Federal Republic of Germany, and Department of Mathematics, Arizona State University, Tempe, Arizona 85287.

‡ Fachbereich Mathematik and Rechenzentrum, Johannes Gutenberg Universität, Postfach 3980, 6500 Mainz 1, Federal Republic of Germany.

**1. Nonlinear eigenvalue problems.** Instead of describing in detail the classes of continuous nonlinear eigenvalue problems which may be considered, we assume that by a suitable discretization with parameter  $h$  a finite-dimensional problem of the form

$$(1.1) \quad f(x) = \lambda Bx, \quad x \in R^n, \quad \lambda \in R, \quad n = n(h)$$

has been obtained, where  $B$  is a symmetric and positive definite  $n \times n$  matrix and  $f: R^n \rightarrow R^n$  a smooth mapping. This gives some preference to a discretization by finite element methods since these usually yield a symmetric and positive definite  $B$  as discretization of an elliptic operator. The assumption on  $B$  may, however, be relaxed (cf. § 5).

Problem (1.1) is more general than that considered in [10] since there it was assumed that  $f$  is the gradient of a functional or equivalently that the matrix  $F = f'$  is symmetric. Then solutions of (1.1) may be characterized as critical points of that functional and this was exploited in [10] to develop a global convergence theory for the generalized inverse iteration and in [12] to distinguish between relevant and irrelevant solutions of the discrete eigenvalue problem.

The symmetry of  $F$  together with that of  $B$  was also advantageous numerically in [10] since it allowed to use conjugate gradient methods particularly suited for the resulting linear systems. These methods were applied to the augmented system (cf. (2.1), (2.2)). That this technique in general is preferable over the combination of block-elimination [6] and conjugate gradient methods has also been confirmed for nonsymmetric systems by the numerical results in [3].

**2. The basic algorithm.** The generalized inverse iteration for the solution of (1.1) iteratively computes from a given pair  $(x_k, \lambda_k)$ ,  $\|x_k\|_B = \rho$ ,

$$(2.1a) \quad \begin{aligned} \tilde{x}_{k+1} &= x_k - H_k f(x_k), \\ x_{k+1} &= \rho \tilde{x}_{k+1} / \|\tilde{x}_{k+1}\|_B, \\ \lambda_{k+1} &= f(x_{k+1})^T x_{k+1} / \rho^2 \end{aligned}$$

where

$$(2.1b) \quad H_k = \begin{bmatrix} F_k - \lambda_k B & -Bx_k \\ -x_k^T B & 0 \end{bmatrix}_{n \times n}^{-1}.$$

Here  $\|\cdot\|_B$  denotes the norm introduced by  $B$ .  $H_k$  is the  $n \times n$  principal submatrix of the inverse of the matrix in brackets provided this is regular. We note a relationship to Newton's method for the augmented system

$$(2.2) \quad f(x) - \lambda Bx = 0, \quad -\frac{1}{2}\|x\|_B^2 + \frac{\rho^2}{2} = 0.$$

$\tilde{x}_{k+1}$  is the update obtained by one step of Newton's method for (2.2) starting from  $x_k$ . For ways to compute  $\tilde{x}_{k+1}$  for discretizations of pde problems see, for example, [10]. The solutions of (1.1) are parametrized by their  $B$ -norm and (2.1) in contrast to Newton's method for (2.2) actually generates only iterates with  $B$ -norm equal to  $\rho$ . The second difference is that the parameter  $\lambda$  is updated by a Rayleigh-quotient in (2.1). The starting values for (2.1) are thus assumed to be  $(x_1, \lambda_1)$  with  $\|x_1\|_B = \rho$  and  $\lambda_1 = f(x_1)^T x_1 / \rho^2$ .

We must expect in general the existence of multiple solutions with respect to  $\lambda$  and  $\rho$ . A globally convergent method as, for example, a damped Newton method for (2.2) or the global version of the generalized inverse iteration of [10] may converge

to any of these solutions for the given parameter value. So, in order to stay on a solution branch we rather try to exploit the local convergence properties of an algorithm. The following result was proved in [10], [12].

**THEOREM 2.1.** *Let  $f$  in (1.1) be twice continuously differentiable in a neighbourhood of a solution  $(x_0, \lambda_0)$  of (1.1) and assume that*

$$N(F_0 - \lambda_0 B) \subset \text{span} \{Bx_0\}.$$

*Here  $N(L)$  denotes the nullspace of the linear operator  $L$ . Then for  $x_1, \|x_1\|_B = \|x_0\|_B = \rho$  sufficiently close to  $x_0$  the sequence  $\{x_k\}$  generated by (2.1) converges quadratically to  $x_0$ .*

**3. The continuation method.** The parametrization by  $\rho$  in the generalized inverse iteration is, of course, not as general as one by, for example, the pseudo-arclength along the solution curve. In the following we shall see, however, that many problems of the form (1.1) may be solved very efficiently and reliably with a very simple  $\rho$ -continuation technique. This technique may be further refined and combined, for example, with methods to switch branches at bifurcation points or to compute singular points. In the following we describe the basic step of the  $\rho$ -continuation process.

First we derive some useful formulae.

**LEMMA 3.1.** *If the following expressions are well-defined it holds for solutions of problem (1.1) that*

$$(3.1) \quad \frac{d\rho}{d\lambda} = x^T B \frac{dx}{d\lambda} / \rho,$$

$$(3.2) \quad \frac{d(x/\|x\|_B)}{d\rho} = \left( \frac{dx}{d\lambda} \frac{d\lambda}{d\rho} - \frac{x}{\rho} \right) / \rho.$$

*Proof.* We have  $\rho^2 = f(x)^T x / \lambda$ , so

$$\frac{d\rho}{d\lambda} = \left[ \left( x^T F(x) \frac{dx}{d\lambda} + f(x)^T \frac{dx}{d\lambda} \right) \lambda - f(x)^T x \right] / (2\rho\lambda^2).$$

From (1.1) we have, except in turning points,

$$(3.3) \quad (F(x) - \lambda B) \frac{dx}{d\lambda} = Bx$$

from which (3.1) follows. But by differentiation and using (3.1) we derive (3.2).

If a solution  $(x_\rho, \lambda_\rho)$  for a given  $\rho$ -level is known and one for a different level  $\rho + \delta\rho$  is to be computed then algorithm (2.1) requires the starting guess to have that norm. A very simple predictor-step is thus to use  $x_\rho(\rho + \delta\rho)/\rho$  and the corresponding  $\lambda$  given by the Rayleigh-quotient. For the quality of this guess the derivative (3.2) is important. We therefore propose to choose  $|\delta\rho|$  inversely proportional to this derivative. With a suitable monotone function  $g$ , (see, for example, (6.1)), let

$$(3.4) \quad |\delta\rho| = g \left( \left\| \frac{d(x/\|x\|_B)}{d\rho} \right\|_B^{-1} \right)$$

where

$$(3.5) \quad \left\| \frac{d(x/\|x\|_B)}{d\rho} \right\|_B = \left( \left\| \frac{dx}{d\lambda} \right\|_B^2 \left( \frac{d\lambda}{d\rho} \right)^2 - 1 \right)^{1/2} / \rho.$$

It is necessary, however, to bound the step-length proposed by (3.4) in cases when the change of the solution is rather small but also the slope  $d\rho/d\lambda$  is small. By

extrapolation of  $d\rho/d\lambda$  as a function of  $\rho$  to a zero we obtain from the recent values  $\rho_{k-1}, \rho_k$

$$(3.6) \quad (\delta\rho_k)_{\max} := -t \frac{d\rho}{d\lambda}(\rho_k) \frac{\rho_k - \rho_{k-1}}{(d\rho/d\lambda)(\rho_k) - (d\rho/d\lambda)(\rho_{k-1})}$$

where  $t=2$  is usually chosen corresponding to an expected double zero, although, of course a zero of  $d\rho/d\lambda$  need not be forthcoming along the branch.

Utilizing the above ideas the following strategy was used for continuation with the generalized inverse iteration to follow a branch from level  $\rho_0$  to the target level  $\rho_t$ . In step M1 the maximum allowable steplength  $|p|$  is determined.

*The continuation strategy.* Let  $x_0$  be a solution to  $\rho = \rho_0$  and let

$$\text{sg} := \text{signum}(\rho_t - \rho_0); k := 0.$$

Let  $RL_k, DX_k, DR_k, k=0, 1, \dots$  and  $DM_k, k=1, 2, \dots$  be the expressions given by (3.1), (3.2), (3.4) and (3.6), respectively.

M1:  $p :=$  if  $DX_k \neq 0$  then  $\text{sg} \cdot DR_k$  else  $\rho_t - \rho_0$ ;  
 if  $k = 0$  then goto M2;  
 if  $\text{itold} \leq 2$  then  $p := p + p$ ;  
 $q :=$  if  $RL_k \cdot RL_{k-1} < 0$  then 0 else  $DM_k$ ;  
 if  $\text{sg} > 0$  and  $q > 0$  then  $p := \min(p, q)$ ;  
 if  $\text{sg} < 0$  and  $q < 0$  then  $p := \max(p, q)$ ;  
 M2:  $\delta\rho := \text{sg} \cdot \min(|p|, |\rho_t - \rho_0|)$ ;  $\text{it} := 0$ ;  $k := k + 1$ ;  
 M3:  $\rho_k := \rho_{k-1} + \delta\rho$ ;  
 M4: perform one iteration of (2.1);  $\text{it} := \text{it} + 1$ ;  
 if stopping criterion satisfied then  
 if  $\rho_k = \rho_t$  then stop else begin  $\text{itold} := \text{it}$ ; goto M1 end  
 else if  $\text{it} < \text{itmax}$  then goto M4 else  
 begin  $\text{itold} := \text{it}$ ;  $\text{it} := 0$ ;  $x := x_{k-1}$ ;  $\delta\rho := \delta\rho/2$  end;  
 goto M3;

**4. A multi-grid algorithm.** In the following we present a MG-version of the generalized inverse iteration of § 2 which generalizes the two-level method for which convergence was shown in [11]. We use a sequence of grids  $G^{(0)}, G^{(1)}, \dots, G^{(l)}$  with grid-constants  $h^{(0)} > h^{(1)} > \dots > h^{(l)} > 0$  and assume that the coarsest grid  $G^{(0)}$  is fine enough to qualitatively approximate the solutions on the finest grid well enough. The interpolation (restriction) operators from  $G^{(j)}$  to  $G^{(j+1)}$  ( $G^{(j+1)}$  to  $G^{(j)}$ ) are denoted by  $I_j^{j+1}$  ( $I_{j+1}^j$ ).

The following algorithm may be applied in connection with the  $\rho$ -continuation of § 3. It is assumed that a solution on the coarsest grid has been computed and in a full multi-grid or nested iteration way the algorithm then computes a corresponding point on the solution curve for the finest grid. On grid  $G^{(i)}$  we denote  $x = x^{(i)}$ ,  $B = B^{(i)}$ ,  $f = f^{(i)}$  and  $\|\cdot\|_i = \|\cdot\|_{B^{(i)}}$ . An eigenvalue parameter  $\lambda$  is always assumed to be related to the given  $x$  by the generalized Rayleigh-quotient (cf. (2.1a)). A total of  $l_{\max}$  grids is used. The function  $R$  relates the norms on the different levels, see (7.2) for an example.  $\tilde{I}_{i-1}^i$  denotes an interpolation which may differ from  $I_{i-1}^i$ .

*The MG algorithm.* Let  $x^{(0)}, \lambda^{(0)}$  with  $\|x^{(0)}\|_0 = \rho^{(0)}$  be a given solution. Set  $\rho^{(i)} := R(\rho^{(0)}, i)$ ,  $i=1, \dots, l_{\max}$  and  $l := 1$ . We present the algorithm in the usual quasi-ALGOL form.  $H^{(0)}$  formally denotes the matrix of (2.1b) on level 0 evaluated at  $x^{(0)}, \lambda^{(0)}$ .

```

for  $l := 1$  step 1 until  $l_{\max}$  do
  begin  $x := \tilde{I}_{l-1}^l x^{(l-1)}$ ;  $\lambda := \lambda^{(l-1)}$ ;
    for  $j := 1$  step 1 until  $k^{(l)}$  do MG ( $l, x, \lambda, d$ );
     $x^{(l)} := \rho^{(l)} x / \|x\|_l$ ;  $\lambda^{(l)} := f(x^{(l)})^T x^{(l)} / \rho^{(l)2}$ 
  end;
  procedure MG ( $l, u, \lambda, d$ );
  integer  $l$ ; real  $\lambda$ ; array  $u, d$ ;
  if  $l = 0$  then  $u := H^{(0)} I_1^0 d$  else
  begin integer  $j, \nu_1, \nu_2$ ; array  $v$ ;
    for  $j := 1$  step 1 until  $\nu_1$  do  $u := \mathcal{S}_1(u, d)$ ;
     $d := I_l^{l-1} (-f(u) + \lambda Bu)$ ;  $v := 0$ ;
    MG ( $l-1, v, \lambda, d$ );
     $u := u + I_{l-1}^l v$ ;
    for  $j := 1$  step 1 until  $\nu_2$  do  $u := \mathcal{S}_2(u, d)$ 
  end;

```

$\mathcal{S}_i, i=1, 2$ , denotes smoothing w.r.t. the linear system  $Bu = f(u)/\lambda$  if called on the highest level and subsequent Rayleigh-quotient update of  $\lambda$  if  $i = 1$  while it denotes smoothing w.r.t.  $(F(x^{(l)}) - \lambda^{(l)} B)u = d$  on lower levels.

We note that the above algorithm is completely defined if the function  $R$  is specified and a smoothing method and a stopping criterion have been chosen. This algorithm should have advantages over methods that make use of the parametrization by  $\lambda$  as, for example, is the case for the pseudo-arclength method. There  $s$  is introduced as auxiliary parameter but  $\lambda = \lambda(s)$  is kept and hence difficulties have to be expected for a multi-grid version at least in regions where for a given  $\lambda$  not on all grids solutions exist. This will, for example, in general be the case near singular points.

In [1] (cf. also [5]) it was proposed to look for solutions on the finer grid curves on a line orthogonal to the coarse-grid curve and to use additional diagonal shifts of the Jacobians in order to assure that these matrices all have the same number of negative eigenvalues but sacrificing quadratic convergence. While the first strategy is not needed here, once a suitable function  $R$  has been chosen, the second was not necessary for the computations reported in § 7. For another technique overcoming the first problem in the neighbourhood of simple primary bifurcation points see [16].

We do not analyse here the above MG algorithm theoretically but we present some numerical results which show the efficiency of this approach.

**5. Extensions.** As mentioned earlier several extensions of the generalized inverse iteration, the continuation strategy and the MG version are possible. We discuss here only a few ideas most of which have been tested numerically.

In order to have a general purpose continuation procedure several features would have to be added to the basic ideas outlined in the preceding sections. In order, for example, to follow branches bifurcating from the trivial solution at eigenvalues of the linearization it is not possible to follow the trivial solution and branch off since only solutions with nonvanishing norm may be computed. The eigenvalues and -vectors of the linear eigenvalue problems may, however, be computed by the algorithm and be used as starting guesses.

If then along such a primary branch a secondary bifurcation point is detected one may switch to a bifurcating branch by methods as proposed in [6] and implemented in [1]. We have usually preferred to use a simple perturbation.

If it is desired to accurately determine singular points the present approach has the advantage of allowing very crude initial guesses because of its robustness if an

appropriate method is used for computation of the singular points. For the determination of (simple) turning points we have, for example, used polynomial interpolation in  $\rho$  in order to find extrema of  $\lambda(\rho)$  (cf. [13]). The condition  $d\lambda/d\rho = 0$  may, of course, be exploited explicitly. Since in the MG version continuation is done on the coarsest grid a direct method will usually be used for solving the linear systems. Hence the determinant is available for detection and computation of singular points. A point on the curve with  $d\rho/d\lambda = 0$  may, of course not be overcome in general by  $\rho$ -continuation. In this case a step of  $\lambda$ -continuation should be used as long as  $|d\rho/d\lambda|$  is below a suitable threshold.

We conclude with a remark on the case of unsymmetric but regular  $B$ . In this case we define the  $B$ -norm as  $\|x\|_B = \|Bx\|$  and formally multiply (1.1) from the left by  $B^T$ . The modifications necessary in (2.1) are, however, minimal. The Rayleigh-quotient is replaced by  $\lambda = f(x)^T Bx / \rho^2$  and the last row of the matrix on the right of (2.1b) becomes  $[-x_k^T B^T B, 0]$ . So one additional matrix-vector product has to be computed. An analogue of the convergence theorem holds and the method has been used successfully, for example, to treat finite-difference discretizations of problems with mixed boundary conditions.

We point out that this method is not equivalent to using the normal equations but assures only that the matrix  $B$  is symmetric and positive definite. The convergence speed, however, in general deteriorates somewhat and no MG experience has been gathered yet.

**6. Continuation results.** The generalized inverse iteration for  $\rho$ -continuation along solution curves of various nonlinear eigenvalue problems turned out to be very robust and efficient. By robust we mean that rather large stepsizes were possible without leading to divergence or to jumping to another branch. The efficiency was measured by looking at iteration counts in case the work per iteration was similar or otherwise by comparing computing times. A few results were reported in [12]. The methods that were used for comparisons are the  $\lambda$ -continuation with predictor-step, the pseudo-arclength method of [6], continuation along a suitable component of the solution as proposed, for example, in [14] and the method of [9].  $\rho$ -continuation required frequently half or less of the iterations needed by the other methods if relatively small steps were chosen. It still converged in slightly more iterations for larger stepsizes while the other methods then often exhibited convergence to another branch or divergence. In the following we restrict the representation to a few examples and a comparison with the pseudo-arclength method as implemented in [1], i.e., using an adaptive strategy for choosing the parameter  $\theta$  (cf. [6]) and a step picker similar to Rheinboldt's [14].

Developing an automated continuation algorithm with any underlying method usually makes it necessary to choose smaller stepsizes than the method would allow. This is in particular true for  $\rho$ -continuation with the generalized inverse iteration for which rather large steps may be taken (cf. [12]). A balance between the goals of providing a detailed impression of the solution curve and not spending too much computational work seems desirable.

The results in this section were obtained with the  $\rho$ -continuation strategy of § 3 for the basic algorithm of § 2. In all cases the nonlinear eigenvalue problems were posed on the unit square  $Q = (0, 1) \times (0, 1)$  with homogeneous Dirichlet boundary conditions. The Laplacian was discretized by using the usual five-point difference star yielding the matrix  $B$  except for (6.3a). For the finite element method in [1] this was accomplished by choosing the standard triangulation of a square mesh. The right-hand

side was discretized pointwise yielding the function  $f(x)$  for the  $\rho$ -continuation while a suitable quadrature formula is used in [1]. The discretization parameter was  $h = \frac{1}{4}$  since that is a reasonable coarsest grid for a MG algorithm at least for bifurcation from the lowest eigenvalues. Finally, the function  $g$  in (3.4) was chosen as

$$(6.1) \quad g(s) = \frac{1}{4h} \sqrt{s}.$$

The first example is the well-known Bratu problem

$$(6.2) \quad -\Delta u = \mu \exp\left(\frac{u}{1 + \varepsilon u}\right), \quad \varepsilon \cong 0.$$

The branch of positive solutions emanating from the origin has one, two or no (simple) turning points.  $\rho$ -continuation was started with the constant solution and the branch was followed from  $\rho_0 = 1$  to  $\rho_t = 100$ . Table 6.1 shows the steps taken and the accumulated iteration counts denoted by iter.

TABLE 6.1  
 $\rho$ -continuation for Bratu's problem,  $\varepsilon = 0$ .

$\rho$	$\mu$	iter
1	1.363	2
12.45	6.670	5
19.88	5.076	9
22.23	4.205	13
26.91	2.622	17
33.03	1.309	21
42.31	.4168	24
58.80	.04711	26
100	.001466	28

Between  $\rho = 20$  and  $\rho = 40$  the continuation algorithm chooses relatively small steps increasing the total iteration count. We did not try to modify the strategy since it allows to solve efficiently problems with completely different solution curves.

For PLTMGC the accumulated work depended strongly on the  $\mu$ -steps chosen. Table 6.2 represents the best results we have achieved in a series of runs. The intermediate steps taken by the algorithm are not given.

TABLE 6.2  
PLTMGC—results for Bratu's problem,  $\varepsilon = 0$ .

$\mu_t$	$\mu$	iter
7	7	14
8	7	43
2	2	61
.1	.1	87
.001	.001	102

Here  $\mu_t$  denotes the target values used. The turning point is at  $\bar{\mu} \approx 7.3$ . The algorithm starts in the origin and if  $\mu_t > \bar{\mu}$  the algorithm tries to continue to the previous target value but beyond the turning point. Similar results for both methods were obtained for the case  $\varepsilon = .2$  when two turning points are present.

We turn now to the simple bifurcation problems

$$(6.3a) \quad -\Delta u = \mu u - u^3,$$

$$(6.3b) \quad -\Delta v = \mu(v - v^3).$$

These problems are equivalent for positive eigenvalue parameters via the transformation  $u = \mu^{1/2}v$ . The branch bifurcating from the first eigenvalue was computed which for problem (6.3b) does not extend beyond a certain norm-level. Hence this is a test case with small values for  $d\rho/d\lambda$  for the  $\rho$ -continuation. Starting solution was the first eigenfunction. The results for both problems are in fact quite similar so we present only those for (6.3a), which was rewritten for the generalized inverse iteration as

$$-\Delta u + u^3 = \mu u.$$

The left-hand side was discretized to yield the vector  $f$  in (1.1) while  $B$  was the identity matrix.

TABLE 6.3  
 $\rho$ -continuation for (6.3a),  $\rho_0 = 1$ ,  $\rho_t = 60$ .

$\rho$	$\mu$	iter
1	18.89	1
19.56	63.53	5
22.09	75.33	8
37.34	176.2	11
60	421.3	14

TABLE 6.4  
PLTMGC results for (6.3a).

$\mu_t$	iter
50	31
200	49

Many other examples confirmed these results. We make a final remark on the problem

$$(6.4) \quad -\Delta u = \mu u^p, \quad p > 1$$

which has a solution branch of the form in Fig. 6.1.

No starting guess is easily available and it was suggested in [4] to perturb (6.4) by adding  $\delta > 0$  to the right-hand side yielding the dotted curve in order to be able to “jump” on the branch.

For  $\rho$ -continuation a constant initial guess allowed computation of any point on the branch in, for example, 7 steps for  $p = 5$ . Since, however, all solutions on the branch are proportional, the derivative (3.2) is computed as zero and any other solution is then obtained without a single iteration by simple normalization. There is no similar advantage in following this curve with the pseudo-arclength method.

We have seen that already the strategy proposed in § 3 makes  $\rho$ -continuation a very competitive method for following branches of finite-dimensional nonlinear eigenvalue problems of the form (1.1).



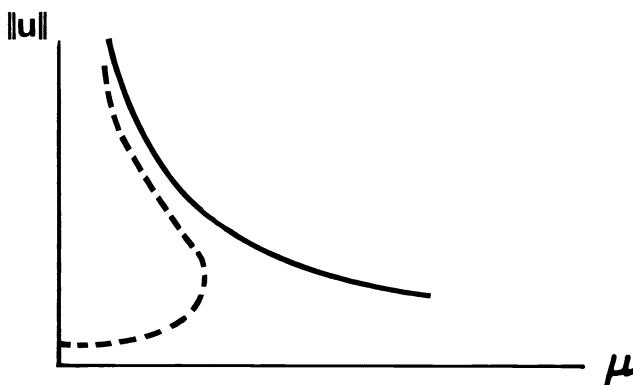


FIG. 6.1. Solution branch for (6.3).

The numerical results of this section were obtained using single-precision FORTRAN on the IBM 3081 at Arizona State University.

**7. Multi-grid results.** In this section some experience will be reported with the MG version of the generalized inverse iteration as given in § 4. The program was written for problems of the form

$$(7.1) \quad \begin{aligned} -\Delta u &= \mu f(u) \quad \text{in } \Omega, \\ u &= g \quad \text{on } \partial\Omega, \end{aligned}$$

where  $\Omega \subset R^2$  was a rather arbitrary domain as allowed in [15] from which auxiliary routines were taken and appropriately modified.

In all computations  $\nu_1 = \nu_2 = 2$  was chosen in the MG algorithm. The function  $R$  was

$$(7.2) \quad R(s, i) = s \cdot h^{(0)} / h^{(i)}, \quad i = 1, \dots, l_{\max}$$

and the iteration was stopped when

$$|\lambda_{k+1}^{(l)} - \lambda_k^{(l)}| < \text{eps} |\lambda_k^{(l)}|.$$

The smoother used was checked Gauss-Seidel relaxation, the interpolation  $\tilde{I}_{l-1}^l$  was of higher order,  $I_{j-1}^j$  was linear interpolation while the restriction  $I_{j+1}^j$  was injection. The grids had mesh-widths  $h^{(i)} = 2^{-i}h^{(0)}$ ,  $i = 1, \dots, l_{\max}$ .

As alternative smoother SSOR-MINRES was used, i.e., an iterate was updated by adding  $\alpha_k p_k$  where  $p_k$  is the direction given by a step of the standard SSOR method (and eventually orthogonalized w.r.t. the previous direction) starting from the current iterate. The stepsize  $\alpha_k > 0$  was chosen to minimize the residual of the corresponding linear systems. For more indefinite cases this smoothing should be superior. In the computations reported here, where never divergence or no convergence occurred, it only increased the computing time. It was also not necessary to use the normal equations for the smoothing.

Since the computing time is of interest for the MG version we include it in the tables (in seconds). The computations were performed in FORTRAN on the HB 66/80 at the Computing Center of the University of Mainz. We present the results for the MG refinement process at a few selected points on the solution curves of each of the following examples.  $Q$  denotes the unit square while  $K(0, r)$  is the circle around the

origin with radius  $r$ . Homogeneous Dirichlet conditions were prescribed in all examples. The starting solution was constant in Examples 1 and 3 and the restriction of the eigenfunctions in the other cases.

*Example 1.*

$$-\Delta u = \mu e^u \text{ in } Q, h^{(0)} = \frac{1}{4}, l_{\max} = 4, \text{eps} = 10^{-5}.$$

*Example 2.*

$$-\Delta u = \mu \sin u \text{ in } Q, \text{ branch from eigenvalue } \mu = \lambda_{22} = 8\pi^2, h^{(0)} = \frac{1}{4}, l_{\max} = 5, \text{eps} = 10^{-6}.$$

*Example 3.*

$$-\Delta u = \mu(1 + u + u^2/2)/(1 + u^2/100) \text{ in } Q, h^{(0)} = \frac{1}{4}, l_{\max} = 5, \text{eps} = 10^{-6}.$$

*Example 4.*

$$-\Delta u = \mu(u - u^3) \text{ in } K(0, \frac{1}{2}), \text{ branch from eigenvalue } \mu = \lambda_0 = k^2, \text{ where } k \text{ is twice the smallest zero of the Bessel function } J_0, h^{(0)} = \frac{1}{5}, l_{\max} = 4, \text{eps} = 10^{-5}.$$

Following the bifurcating branches in Examples 2 and 4 for large values of  $\lambda$  required an increasing number of iterations. The SSOR-MINRES smoother brought considerable improvement but the function  $R$  in (7.2) is not appropriate here for large values of  $\lambda$  and would have to be modified suitably if it is desired to compute such solutions.

TABLE 7.1  
Multi-grid results for Example 1.

$\rho^{(0)}$	$\mu^{(3)}$	$k^{(3)}$	time
8	6.142	3	3.731
10	6.637	4	4.281
12	6.806	4	4.288
14	6.653	4	4.471
16	6.179	4	4.607

TABLE 7.2  
Multi-grid results for Example 2.

$\rho^{(0)}$	$\mu^{(4)}$	$k^{(4)}$	$\ u\ _{\infty}$	time
5	79.48066	2	.280538	9.476
15	84.33039	2	.834066	9.363
25	94.83650	3	1.35935	12.381
40	125.3734	3	2.05389	12.826

TABLE 7.3  
Multi-grid results for Example 3.

$\rho^{(0)}$	$\mu^{(4)}$	$k^{(4)}$	$u(.5, .5)$	time
19	8.019515	4	2.10733	13.752
20	8.031423	4	2.22549	13.612
21	8.032892	4	2.34387	13.597
22	8.025606	4	2.46241	13.619

TABLE 7.4  
Multi-grid results for Example 4.

$\rho^{(0)}$	$\mu^{(3)}$	$k^{(3)}$	$\ u\ _\infty$	time
1	23.21995	2	.0903495	2.857
4	24.93392	2	.356758	2.735
7	29.50112	3	.603734	3.847
9	35.21668	4	.745784	4.904
11	44.54570	2	.858101	3.792

**Acknowledgment.** The authors would like to thank the referees for valuable comments and corrections.

*Note added in proof.* The final version of [1] does not use the pseudo-arclength method for continuation but a method which may be viewed as another way to generalize the algorithm considered here. The following features of our algorithm are utilized in [1]: Parametrization by  $\rho$  and  $\lambda$ , use of the Rayleigh-quotient in the predictor step, the  $n$ -vector in the last row of the Jacobian is proportional to  $x_k$  (corresponding to using the Euclidean instead of the  $B$ -norm in (2.1)), MG refinement for a fixed value of  $\rho$  (or  $\lambda$ ) on all grids. This led to a drastic reduction of the number of continuation steps compared to those in Tables 6.2 and 6.4.

#### REFERENCES

- [1] R. E. BANK AND T. F. CHAN, PLTMGC: A multi-grid continuation package for solving parametrized nonlinear elliptic systems, Report # 261, Dept. Computer Science, Yale Univ., New Haven, CT, 1983.
- [2] T. F. CHAN AND H. B. KELLER, Arclength continuation and multi-grid techniques for nonlinear eigenvalue problems, this Journal, 3 (1982), pp. 173–194.
- [3] T. F. CHAN AND Y. SAAD, Iterative methods for solving bordered systems with applications to continuation methods, Report # 235, Dept. Computer Science, Yale Univ., New Haven, CT, 1982.
- [4] R. GLOWINSKI, H. B. KELLER AND L. REINHART, Continuation-conjugate gradient methods for the least square solution of nonlinear boundary value problems, INRIA-Report # 141, LeChesnay, France, 1982.
- [5] W. HACKBUSCH, Multi-grid solution of continuation problems, in Iterative Solution of Nonlinear Systems, R. Ansoerge, T. Meis and W. Törnig eds., Lecture Notes in Mathematics 953, Springer, New York, 1982.
- [6] H. B. KELLER, Numerical solution of bifurcation and nonlinear eigenvalue problems, in Applications of Bifurcation Theory, P. Rabinowitz, ed., Academic Press, New York, 1977.
- [7] T. MEIS, H. LEHMANN AND H. MICHAEL, Application of the multigrid method to a nonlinear indefinite problem, in Multi-Grid Methods, W. Hackbusch and U. Trottenberg, eds., Lecture Notes in Mathematics 960, Springer, New York, 1982.
- [8] R. G. MELHEM AND W. C. RHEINBOLDT, A comparison of methods for determining turning points of nonlinear equations, Computing, 29 (1982), pp. 201–226.
- [9] R. MENZEL AND H. SCHWETLICK, Zur Lösung parameterabhängiger nichtlinearer Gleichungen mit singulären Jacobi-Matrizen, Numer. Math., 30 (1978), pp. 65–79.
- [10] H. D. MITTELMANN, An efficient algorithm for bifurcation problems of variational inequalities, Math. of Comp., 41 (1983), pp. 473–485.
- [11] ———, Multi-grid methods for simple bifurcation problems, in Multi-Grid Methods, W. Hackbusch and U. Trottenberg, eds., Lecture Notes in Mathematics 960, Springer, New York, 1982.
- [12] ———, A fast solver for nonlinear eigenvalue problems, in Iterative Solution of Nonlinear Systems, R. Ansoerge, T. Meis and W. Törnig, eds., Lecture Notes in Mathematics 953, Springer, New York, 1982.
- [13] G. PÖNISCH AND H. SCHWETLICK, Ein lokal überlinear konvergentes Verfahren zur Bestimmung von Rückkehrpunkten implizit definierter Raumkurven, Numer. Math., 38 (1982), pp. 455–566.

- [14] W. C. RHEINBOLDT, *Solution fields of nonlinear equations and continuation methods*, SIAM J. Numer. Anal., 17 (1980), pp. 222–237.
- [15] K. STÜBEN, MGØ1: *A multi-grid program to solve  $\Delta U - c(x, y)U = f(x, y)$  (on  $\Omega$ ),  $U = g(x, y)$  (on  $\partial\Omega$ ), on nonrectangular bounded domains  $\Omega$* , IMA-Report # 82.02.02, Gesellschaft für Mathematik und Datenverarbeitung, St. Augustin, 1982.
- [16] H. WEBER, *An efficient technique for the computation of stable bifurcation branches*, this Journal, 5 (1984), pp. 332–348.

## SOME COSINE SCHEMES FOR SECOND-ORDER SYSTEMS OF ODE'S WITH TIME-VARYING COEFFICIENTS\*

STEVEN M. SERBIN†

**Abstract.** We derive a family of two-step fourth-order approximation schemes for the second-order linear system  $u_{tt} + A(t)u = f(t)$  with initial conditions  $u(0) = u_0, u_t(0) = v_0$ . These are extensions to the case that  $A$  is time-dependent of some previously developed methods resulting from rational approximation of the cosine operator for time-independent  $A$ , and are unconditionally stable for choice of family parameter  $\beta \cong \frac{1}{4} + (\frac{1}{24})^{1/2}$ .

We first employ a three-point Hermitian expansion to develop a base scheme and then refine it so that fourth order is maintained but only values of  $A(t)$  and  $f(t)$ , not their derivatives, are required. We similarly modify a locally fifth-order Taylor expansion to provide an easily computable starting procedure to approximate the solution at the first time step. Our scheme has the advantage that the operator to be "inverted" at each step is the square of a linear operator in  $A(t)$ . We also identify and discuss a fourth-order unconditionally stable method due to Hairer as another cosine scheme resulting from a different form of rational approximation.

We present numerical evidence for both a scalar and a system problem of the fourth-order accuracy of the base scheme through to the final form with approximate starting procedure. We finally discuss the applicability of our schemes to implicit systems of the form  $Gu_{tt} + S(t)u = f(t)$  which arise, for example, in finite element semidiscretization of second-order hyperbolic problems with time-varying coefficients.

**Key words.** cosine schemes, second-order systems, time-varying coefficients

**1. Introduction.** In several previous papers, we have introduced and analyzed a class of "cosine schemes" for the second-order systems

$$(1.1) \quad \begin{aligned} u_{tt} + Au &= f(t), & t \in (0, T], \\ u(0) &= u_0, & u_t(0) = v_0, \end{aligned}$$

in various settings, from the concrete case where  $u \in \mathbb{R}^N$  (or  $\mathbb{C}^N$ ) and  $A$  is an  $N \times N$  constant matrix in [8], to the more abstract Hilbert space setting considered in [1] and [4]. Applications to time-stepping methods for second-order hyperbolic partial differential equations are found in [2] and [5]. These methods are called cosine schemes because they are developed so as to approximate the exact relation (for the homogeneous problem)

$$(1.2) \quad u(t+k) - 2 \cos(kA^{1/2})u(t) + u(t-k) = 0, \quad k \leq t \leq T-k,$$

by replacing the cosine operator in (1.2) by an appropriate rational approximation. We have concentrated on a specific family of rational approximations (to be reviewed below), which yield for the homogeneous case of (1.1) schemes of arbitrarily high order which, depending upon choice of a parameter, can always be selected to be unconditionally stable and which are efficient to implement.

In this paper, we advance to the case in which  $A = A(t)$ , i.e.  $A$  depends upon time. We shall restrict our attention in this case to the problem (1.1) where  $u \in \mathbb{R}^N$  and we shall assume that  $A(t)$  is a uniformly positive definite, real  $N \times N$  matrix for  $t \in [0, T]$  which is sufficiently smooth to admit all required Taylor expansions ( $A(t) \in C^4[0, T]$  will suffice in the case we discuss herein). Similarly, we assume  $f(t) \in C^6[0, T]$ .

---

\* Received by the editors October 26, 1982, and in revised form August 1, 1983. This research was supported by the U.S. Army Research Office under grant DAAG29-80-K-0056.

† Department of Mathematics, University of Tennessee, Knoxville, Tennessee 37996-1300.

The idea of the cosine schemes for this problem is as follows. Whereas (1.2) no longer is satisfied, it suggests a certain Hermitian (i.e. multipoint Taylor) expansion which we shall then approximate for the solution of (1.1) by replacing derivatives using the differential equation. This idea has been used in the context of parabolic partial differential equations by Nassif and Descloux [7] and Bramble and Sammon [3]. In each of these schemes, greater than second-order can be achieved only at the expense of introducing derivatives of  $A(t)$ . We shall show that due to the symmetry (i.e. use of points  $t+k$ ,  $t$ ,  $t-k$ ) present in our scheme, we can achieve fourth order without requiring any derivatives of  $A(t)$ ; this will follow by suitable replacement of derivatives from our original scheme by appropriate linear combinations of undifferentiated terms.

Whereas we approach this problem as generalization of a class of schemes for  $A$  time-independent, it is certainly possible to specify schemes which have been derived for the generally nonlinear problem  $u_t = g(t, u)$  to our situation. Desiring high-order and unconditional stability, this necessitates some form of implicit scheme. We shall identify certain of our cosine schemes in this context, but the class of fourth-order schemes we present seems to be more efficient.

Since our schemes emulate (1.2), they will be two-step methods. Hence the problem of efficiently starting the computation (specifically, obtaining an approximation to  $u(k)$ ) will be addressed. Of course, it is always possible to convert (1.1) into an equivalent first-order system and apply single-step methods (thus avoiding the starting problem), but unless approximations to  $u_t$  are also required, our approach continues to be the treatment of (1.1) directly in second-order form, which we have shown for the time-independent case to be more efficient than a similarly constructed approximation for the first order formulation.

Our paper will conclude with some numerical examples and some short remarks on the modifications for the implicit equation

$$(1.3) \quad Gu_t + S(t)u = f(t)$$

which arises, for example, in semidiscretization of second-order hyperbolic equations with time-dependent coefficients. A detailed analysis of our schemes for this problem will follow in a subsequent paper.

**2. Derivation of the schemes.** One way to obtain the recursion (1.2) is to consider the *formal* Hermitian expansion

$$(2.1) \quad u(t+k) + u(t-k) = 2 \cosh(k\mathcal{D})u(t)$$

where  $\mathcal{D}$  indicates the operation of differentiation with respect to  $t$ . Then, since in the time-independent case we have

$$(2.2) \quad u^{(2k)} = (-1)^k A^k u$$

(parenthesized superscripts indicating derivatives), we have  $\cosh(k\mathcal{D})u(t) = \cos(KA^{1/2})u(t)$  and the result (1.2) follows. In the present case  $A = A(t)$ , of course (2.2) no longer holds, but (2.1) still suggests a way to generate a class of two-step implicit schemes. Namely, let us formally approximate  $\cosh(k\mathcal{D})$  by an appropriately chosen rational operator  $R(k\mathcal{D}) \equiv P(k\mathcal{D})/Q(k\mathcal{D})$ . Then (2.1) can be approximated by

$$(2.3) \quad Q(k\mathcal{D})[u(t+k) + u(t-k)] \equiv 2P(k\mathcal{D})u(t).$$

Since  $\cosh \tau$  is an even function of  $\tau$ ,  $P(\tau)$  and  $Q(\tau)$  will themselves be even and so only even powers of the differentiation operator occur in (2.3). Thus, for specific choice

of rational approximation to  $\cosh \tau$  (chosen for accuracy and stability considerations) a numerical approximation to the time-dependent  $A(t)$  version of (1.1) can be obtained by operating through (2.3) with the appropriate derivatives, then substituting from the equation, e.g.,

$$(2.4) \quad u^{(2)}(t) = -A(t)u(t) + f(t),$$

$$(2.5) \quad u^{(3)}(t) = -A(t)u^{(1)}(t) - A^{(1)}(t)u(t) + f^{(1)}(t),$$

$$(2.6) \quad u^{(4)}(t) = [A^2(t) - A^{(2)}(t)]u(t) - 2A^{(1)}(t)u^{(1)}(t) + [f^{(2)}(t) - A(t)f(t)],$$

and so on. We observe from (2.5) and (2.6) that these replacements involve not only  $u(t)$  and  $A(t)$ , but also  $u^{(1)}(t)$  and derivatives of  $A(t)$  and  $f(t)$ . Since we wish to emulate the time-independent cosine schemes, we shall endeavor to avoid using these latter quantities, replacing them instead with linear combinations of undifferentiated terms.

Let us now make specific choices of the rational function  $R(\tau)$  which approximates  $\cosh \tau$ . Note first that to each rational approximation  $r(\tau)$  to  $\cos \tau$  there corresponds naturally the approximation  $R(\tau) = r(i\tau)$  to  $\cosh \tau$ . Thus, our previous consideration [1], [8] of rational approximations to the cosine motivates specific families of approximations. We could approximate  $\cos \tau$  by, say, the Padé criterion or by the real part of a Padé approximant to  $e^{i\tau}$  (as it turns out below we identify a method due to Hairer [6]), but we shall concentrate on families of approximations to  $\cos \tau$ , analyzed in [1], which have the form (for parameter  $\beta > 0$ )

$$(2.7) \quad r_\alpha(\beta, \tau) = \left( \sum_{j=0}^{\alpha} \phi_j^{(\alpha)}(\beta) \tau^{2j} \right) / (1 + \beta \tau^2)^\alpha.$$

The functions  $\phi_j^{(\alpha)}(\beta)$  are polynomials of degree  $j$  in  $\beta$  given by

$$\phi_j^{(\alpha)}(\beta) = \sum_{m=0}^j \frac{(-1)^m}{(2m)!} \binom{\alpha}{j-m} \beta^{[j-m]}.$$

These rational functions have the accuracy property

$$(2.8) \quad |r_\alpha(\beta; \tau) - \cos \tau| \leq C\tau^{2\alpha+2}, \quad 0 \leq \tau < \beta^{-1/2},$$

where  $C$  is a constant depending on  $\alpha$  and  $\beta$ , and the stability property that there exists a constant  $\beta^{[\alpha]}$  such that for  $\beta \geq \beta^{[\alpha]}$

$$(2.9) \quad |r_\alpha(\beta; \tau)| \leq 1 \quad \text{for all real } \tau \geq 0.$$

(For conditional stability, if  $0 < \beta < \beta^{[\alpha]}$ , a result like (2.9) holds for  $0 \leq \tau \leq \eta$ .)

The scheme obtained for  $\alpha = 1$  is, in fact, a well-known family of two-step linear methods, so we shall give only the result. Denoting the approximation to  $u(nk)$  by  $u_n$ ,  $n = 0, 1, \dots, [T/k]$  and letting  $A^{(j)}(nk) \equiv A_n^{(j)}$ ,  $f^{(j)}(nk) \equiv f_n^{(j)}$ , we obtain

$$(2.10) \quad \begin{aligned} [I + \beta k^2 A_{n+1}]u_{n+1} &= 2[I + (\beta - \frac{1}{2})k^2 A_n]u_n - [I + \beta k^2 A_{n-1}]u_{n-1} \\ &\quad + k^2[\beta f_{n+1} + (1 - 2\beta)f_n + \beta f_{n-1}], \end{aligned}$$

which is, in general, second-order accurate globally and from (2.1) is unconditionally stable for  $\beta \geq \frac{1}{4}$ . The (exceptional) fourth-order conditionally stable Störmer–Numerov method arises from  $\beta = \frac{1}{12}$ . Observe that approximations to  $u^{(1)}(t)$  do not appear.

While there is nothing new for  $\alpha = 1$ , our approach does yield new methods for  $\alpha \geq 2$ . We could attempt to describe methods for general  $\alpha$ , but it is our belief that the most useful schemes and all of the relevant ideas are included in this case  $\alpha = 2$ , to which we now proceed. Going directly to the rational approximation to  $\cosh \tau$ , from [1] we deduce that

$$(2.11) \quad R(\tau) = \frac{1 + (\frac{1}{2} - 2\beta)\tau^2 + (\beta^2 - \beta + \frac{1}{24})\tau^4}{(1 - \beta\tau^2)^2} = 1 + \frac{\frac{1}{2}[\tau^2 + (\frac{1}{12} - 2\beta)\tau^4]}{(1 - \beta\tau^2)^2}$$

satisfies (2.8) with  $\alpha = 2$  and (2.9) with  $\beta^{(2)} = \frac{1}{4} + (\frac{1}{24})^{1/2}$ . Analogous to (2.10), using (2.4)–(2.6), we obtain first

$$(2.12) \quad \begin{aligned} & (u_{n+1} - 2u_n + u_{n-1}) - 2\beta k^2(-A_{n+1}u_{n+1} + f_{n+1} + 2A_n u_n - 2f_n - A_{n-1}u_{n-1} + f_{n-1}) \\ & + \beta^2 k^4[(A_{n+1}^2 - A_{n+1}^{(2)})u_{n+1} - 2(A_n^2 - A_n^{(2)})u_n + (A_{n-1}^2 - A_{n-1}^{(2)})u_{n-1} \\ & + (f_{n+1}^{(2)} - A_{n+1}f_{n+1}) - 2(f_n^{(2)} - A_n f_n) + (f_{n-1}^{(2)} - A_{n-1}f_{n-1})] \\ & - 2\beta^2 k^4[A_{n+1}^{(1)}u_{n+1}^{(1)} - 2A_n^{(1)}u_n^{(1)} + A_{n-1}^{(1)}u_{n-1}^{(1)}] \\ & = k^2[-A_n, u_n + f_n] + (\frac{1}{12} - 2\beta)k^4[(A_n^2 - A_n^{(2)})u_n - 2A_n^{(1)}u_n^{(1)} + (f_n^{(2)} - A_n f_n)]. \end{aligned}$$

Now, (2.12), at first glance, appears to involve substantial differentiation of  $A(t)$  and  $f(t)$ , and the matrix to be “inverted” is  $(I + \beta k^2 A_{n+1})^2 - \beta^2 k^4 A_{n+1}^{(2)}$ . The local accuracy from (2.8) is  $O(\tau^6)$ , so (2.12) yields a globally fourth-order scheme, which, via the usual linear stability analysis, is unconditionally stable for  $\beta \geq \frac{1}{4} + (\frac{1}{24})^{1/2}$ . However, even if we are willing to compute all required derivatives, we are still required to obtain approximations to  $u^{(1)}(nk)$ , which are not provided by the scheme nor are they in the spirit of the cosine method.

It can be seen that certain combinations of  $O(k^2)$  terms of the form  $W(t+k) - 2W(t) + W(t-k)$  are multiplied in (2.12) by  $k^4$  and may be neglected without disturbing the local  $O(k^6)$  accuracy or the linear stability. In particular, though, the term  $A_{n+1}^2 u_{n+1} - 2A_n^2 u_n + A_{n-1}^2 u_{n-1}$  may not be omitted. We further replace

$$(2.13) \quad ku_n^{(1)} = u_n - u_{n-1} + \frac{k^2}{3}(-A_n u_n + f_n) + \frac{k^2}{6}(-A_{n-1} u_{n-1} + f_{n-1}) + O(k^4),$$

$$(2.14) \quad k^2 A_n^{(2)} = A_{n+1} - 2A_n + A_{n-1} + O(k^4)$$

and

$$(2.15) \quad 2kA_n^{(1)} = A_{n+1} - A_{n-1} + O(k^3),$$

and, after appropriate substitutions, use

$$(2.16) \quad \begin{aligned} & k^2 f_n + \left(\frac{1}{12} - 2\beta\right) \left[ -\frac{k^4}{3}(A_{n+1} - A_{n-1})f_n - \frac{k^4}{6}(A_{n+1} - A_{n-1})f_{n-1} + k^4(f_n^{(2)} - A_n f_n) \right] \\ & = \frac{k^2}{12}(f_{n+1} + 10f_n + f_{n-1}) \\ & + \left(2\beta - \frac{1}{12}\right)k^4 \left[ A_n f_n + \frac{1}{6}(A_{n+1} - A_{n-1})(2f_n + f_{n-1}) \right] + O(k^6), \end{aligned}$$



so that we finally arrive after rearrangement at the class of schemes

$$\begin{aligned}
 & (I + \beta k^2 A_{n+1})^2 u_{n+1} \\
 &= 2(I + \beta k^2 A_n)^2 u_n - (I + \beta k^2 A_{n-1})^2 u_{n-1} \\
 & \quad + k^2 \left[ \frac{1}{12} (f_{n+1} + 10f_n + f_{n-1}) - A_n u_n \right] \\
 (2.17) \quad & + \left( \frac{1}{12} - 2\beta \right) k^2 \left[ -2(A_{n+1} - A_n) u_n + k^2 A_n (A_n u_n - f_n) \right. \\
 & \quad \left. + (A_{n+1} - A_{n-1}) \left\{ \left( I + \frac{k^2}{6} A_{n-1} \right) u_{n-1} + \frac{k^2}{3} A_n u_n - \frac{k^2}{6} (2f_n + f_{n-1}) \right\} \right]
 \end{aligned}$$

which are fourth order and unconditionally stable for  $\beta \geq \frac{1}{4} + (\frac{1}{24})^{1/2}$  and reduce directly to the cosine scheme with  $\alpha = 2$  of [1] when  $A$  is constant. Note that the operator  $I + \beta k^2 A_{n+1}$  is always invertible ( $\beta > 0$ ) under the hypotheses on  $A(t)$  and, as in the constant coefficient case, we can decompose this operator and back solve twice. In practice, we will probably not decompose at every time step, but rather solve (2.17) by some preconditioned iterative method with preconditioner  $(I + \beta k^2 A_n)$ , holding  $m$  fixed for many time steps (starting with  $m = 0$ , of course). One final aspect of (2.17) is noteworthy. If we denote  $z_n = (I + \beta k^2 A_n)^2 u_n$ , then we can first form  $z_{n+1} = 2z_n - z_{n-1} + \dots$  (remaining terms on the right side of (2.17)), and then solve  $(I + \beta k^2 A_{n+1})^2 u_{n+1} = z_{n+1}$ .

The only other fourth-order unconditionally stable scheme of which we are aware results from the specification to our problem of an unconditionally stable method for  $u'' = g(t, u)$  proposed by Hairer [6]. His scheme can be written

$$(2.18) \quad \tilde{u}_n = u_n - \frac{k^2}{12B} (g_{n+1} - 2g_n + g_{n-1}),$$

$$(2.19) \quad u_{n+1} - 2u_n + u_{n-1} = \frac{k^2}{12} (g_{n+1} + (10 - B)g_n + g_{n-1} + Bg(t_n, \tilde{u}_n)).$$

Letting  $g(t, u) = -A(t)u + f(t)$ , it is possible to substitute (2.18) directly into (2.19), and, after considerable simplification, the free parameter  $B$  vanishes and there results

$$\begin{aligned}
 & \left( I + \frac{k^2}{12} A_{n+1} + \frac{k^4}{144} A_n A_{n+1} \right) u_{n+1} \\
 (2.20) \quad &= 2 \left( I - \frac{5k^2}{12} A_n + \frac{k^4}{144} A_n^2 \right) u_n - \left( I + \frac{k^2}{12} A_{n-1} + \frac{k^4}{144} A_n A_{n-1} \right) u_{n-1} \\
 & \quad + \frac{k^2}{12} [f_{n+1} + 10f_n + f_{n-1}] + \frac{k^4}{144} A_n [f_{n+1} - 2f_n + f_{n-1}].
 \end{aligned}$$

This is, indeed, another cosine method, resulting from the rational approximation

$$(2.21) \quad r(\tau) = \frac{1 - 5\tau^2/12 + \tau^4/144}{1 + \tau^2/12 + \tau^4/144}$$

which we considered in [8] as the real part of the (2, 2) Padé approximation to  $e^{i\tau}$ , and is, indeed, unconditionally stable. Whereas the right side of (2.20) seems simpler than that of (2.17) and the error constant will be smaller (as follows from Hairer's

analysis), it seems that the formation and factorization (which requires complex arithmetic) of the matrix  $I + (k^2/12)A_{n+1} + (k^4/144)A_n A_{n+1}$  makes this scheme possibly less inviting, particularly in the case where  $A(t) \equiv G^{-1}S(t)$  for  $G$  and  $S$  sparse (which we consider below for our scheme). Certainly this is true in the time-independent case.

Regardless of which cosine approximation scheme we use, we have to come up with an initial approximation  $u_1$  for  $u(k)$ . We shall concentrate on starting the fourth-order schemes. This requires a fifth-order accurate local expression, but as stability is not a consideration, the scheme being used for only one step, it appears that the most effective starting scheme is simply a Taylor expansion with, again, certain derivatives replaced by differences. Namely, recalling (2.4)–(2.6), it follows by standard expansions that to fifth order, we may approximate  $u(k)$  by

$$(2.22) \quad u_1 = \left[ I + \frac{k^2}{24}(-7A_0 + 6A_1 + A_2) + \frac{k^4}{24}A_0^2 \right] u_0 + k \left[ I - \frac{k^2}{12}(A_0 + A_1) \right] v_0 \\ + \frac{k^2}{24}[7f_0 + 6f_1 - f_2] - \frac{k^4}{24}A_0 f_0.$$

**3. Numerical example and some concluding remarks.** We have implemented the fourth-order cosine scheme in several versions and with several choices of parameter and starting procedure, first on scalar problems and then on a model problem for a system. Our results for the scalar problems indicated clearly the fourth-order convergence with both the exact starting and also (2.22) and with several choices of parameter  $\beta$  for both the base scheme (2.12) and the final scheme (2.17). To settle on the final version of the scheme, we remark that we have shown in [5], and it also follows from Proposition 2 of Hairer [6], that the error constant of the schemes (2.12) or (2.17), i.e. the leading coefficient in  $|\cos \tau - r(\tau)|$  with  $r(\tau)$  given by (2.7) with  $\alpha = 2$ , is  $C = -\frac{1}{720}(360\beta^2 - 60\beta + 1)$ . From this, it follows that the minimum value of  $|C|$  among the unconditionally stable schemes is obtained at the lower bound, namely  $\beta = \frac{1}{4} + (\frac{1}{24})^{1/2}$ .

We thus will examine the performance of the algorithm, now specified to be (2.17) with  $\beta = \frac{1}{4} + (\frac{1}{24})^{1/2}$ , on a model problem for a system of equations. Suppose we consider the partial differential equation

$$(3.1) \quad U_{tt} = [a(x, t)U_x]_x + F(x, t), \quad 0 < x < 1, \quad 0 < t \leq T$$

with some specified initial conditions  $U(x, 0)$ ,  $U_t(x, 0)$  and homogeneous boundary conditions  $U(0, t) = U(1, t) = 0$ . The method of lines can be employed as follows. Let  $x_j = jh$ , where  $(N+1)h = 1$ . Then, we discretize the spatial variable, replacing (3.1) by

$$(3.2) \quad U_{tt}(x_j, t) \approx h^{-2} \{ a(x_{j+1/2}, t)[U(x_{j+1}, t) - U(x_j, t)] \\ - a(x_{j-1/2}, t)[U(x_j, t) - U(x_{j-1}, t)] \} + F(x_j, t), \quad 1 \leq j \leq N.$$

If we let  $u(t)$  be the  $N$ -vector whose  $j$ th component approximates  $U(x_j, t)$ , and we denote  $f_j(t) = F(x_j, t)$  and  $a_j(t) = a(x_j, t)$ , then the semidiscrete equation implied by (3.2) is of the form (1.1) with  $A(t)$  the symmetric tridiagonal matrix with

$$(3.3) \quad A_{jj}(t) = \{ a_{j-1/2}(t) + a_{j+1/2}(t) \} \cdot h^{-2}, \\ A_{j,j+1}(t) = \{ -a_{j+1/2}(t) \} \cdot h^{-2}$$

(where the boundary conditions  $U(x_0, t) = U(x_{n+1}, t) = 0$  are used in the first and last equations). In our numerical experiment, we select  $a(x, t) = 1 + x^2 + t^2$ . Since we are interested in the performance of the ODE solver, we construct a known solution  $u(t)$  with  $j$ th component  $(\cos \pi t + \sin \pi t) \sin \pi jh$  and use the  $j$ th equation to construct  $f_j(t)$  (rather than selecting a solution to a PDE problem (3.1) with which to compare the numerical solution). In Table 1, we exhibit the results of our study of the scheme (2.17) with  $\beta = \frac{1}{4} + (\frac{1}{24})^{1/2}$  for systems of size  $N = 10$  and  $N = 50$ . We report, for both exact starting value for  $u(k)$  and the method (2.22), two measures of the error at  $T = 1$ ,  $E_2 \equiv N^{-1/2} \|u_M - u(1)\|_2$  where  $Mk = 1$  and  $\|\cdot\|_2$  is the usual Euclidean vector norm, and  $E_\infty = \|u_M - u(1)\|_\infty$  the usual maximum norm. We note that for  $N = 50$ , the spectral radius of  $k^2A(0)$ , estimated by the power method, is approximately  $18,530k^2$ , so that for  $k = \frac{1}{10}$ , the problem is moderately stiff, and an unconditionally stable method is appropriate. By way of comparison, with  $\beta = \frac{1}{4}$ , conditionally stable, the scheme blows up for  $k \geq \frac{1}{40}$ .

TABLE 1  
System problem error at  $T = 1$  and rates of error reduction, scheme (2.17),  $\beta = \frac{1}{4} + (\frac{1}{24})^{1/2}$ .

$N = 10$	Exact starting		(2.22) Starting	
	$E_2$	$E_\infty$	$E_2$	$E_\infty$
$\frac{1}{10}$	$.440 \times 10^{-2}$	$.599 \times 10^{-2}$	$.446 \times 10^{-2}$	$.607 \times 10^{-2}$
$\frac{1}{20}$	$.299 \times 10^{-3}$ 3.88	$.425 \times 10^{-3}$ 3.82	$.304 \times 10^{-3}$ 3.87	$.431 \times 10^{-3}$ 3.82
$\frac{1}{40}$	$.191 \times 10^{-4}$ 3.97	$.268 \times 10^{-4}$ 3.99	$.194 \times 10^{-4}$ 3.97	$.272 \times 10^{-4}$ 3.99
$N = 50$				
$\frac{1}{10}$	$.424 \times 10^{-2}$	$.613 \times 10^{-2}$	$.430 \times 10^{-2}$	$.622 \times 10^{-2}$
$\frac{1}{20}$	$.289 \times 10^{-3}$ 3.87	$.435 \times 10^{-3}$ 3.82	$.294 \times 10^{-3}$ 3.87	$.441 \times 10^{-3}$ 3.82
$\frac{1}{40}$	$.184 \times 10^{-4}$ 3.97	$.279 \times 10^{-4}$ 3.96	$.187 \times 10^{-4}$ 3.97	$.283 \times 10^{-4}$ 3.96

We observe that both the exact starting and the Taylor scheme (2.22) yield essentially the same results, which are clearly fourth order in both measures of the error.

Although we used a finite difference approximation in the spatial variables to derive our model problem, we might also have employed a Galerkin finite element procedure, which would give rise to an implicit equation of the form

$$(3.4) \quad Gu^{(2)}(t) + S(t)u(t) = f(t)$$

where  $G$  is the time-independent mass matrix and  $S(t)$  is time-dependent stiffness matrix. Both of these matrices are often large and sparse.

This leads us to the final point of the paper. One obviously can return (3.4) to the form (1.1) by multiplying through by  $G^{-1}$ , but of course one does not do this since symmetry and sparsity would be lost. Suppose, however, that we would consider the scheme (2.17), with  $A(t) = G^{-1}S(t)$ . First, multiply through (2.17) by  $G$ , and let  $\tilde{Z}_j = GZ_j = (G + \beta k^2 S_j)G^{-1}(G + \beta k^2 S_j)u_j$  be the vector stored for  $j = n, n - 1$  for

advancement to step  $n+1$ . Then, we obtain

$$\begin{aligned}
 \check{Z}_{n+1} = & 2\check{Z}_n - \check{Z}_{n-1} + k^2 \left[ \frac{1}{12} G(f_{n+1} + 10f_n + f_{n-1}) - S_n u_n \right] \\
 & + \left( \frac{1}{12} - 2\beta \right) k^2 \left[ -2(S_{n+1} - S_n) u_n + k^2 S_n (G^{-1} S_n u_n - f_n) \right. \\
 (3.5) \quad & \left. + (S_{n+1} - S_{n-1}) G^{-1} \left\{ \left( G + \frac{k^2}{6} S_{n-1} \right) u_{n-1} \right. \right. \\
 & \left. \left. + \frac{k^2}{3} S_n u_n - \frac{k^2}{6} G(2f_n + f_{n-1}) \right\} \right].
 \end{aligned}$$

Clearly, the right-hand side of (3.5) involves no difficulty. Since  $G$  is sparse, positive definite, and time-independent, its sparse Cholesky decomposition need be performed only once, and then the indicated multiplications by  $G^{-1}$  are in fact performed by solving linear systems with the factored matrix  $G$ . All of the other indicated multiplications involve sparse matrices.

Finally, to determine  $u_{n+1}$ , we have

$$(3.6) \quad (G + \beta k^2 S_{n+1}) G^{-1} (G + \beta k^2 S_{n+1}) u_{n+1} = \check{Z}_{n+1},$$

which can obviously be solved by

$$(3.7) \quad (G + \beta k^2 S_{n+1}) W = \check{Z}_{n+1}, \quad (G + \beta k^2 S_{n+1}) u_{n+1} = GW.$$

We see that the same sparse matrix is involved in both steps of (3.7). Clearly, this suggests an obvious preconditioner for any preconditioned iterative method (hold  $S$  fixed over several time steps). While the treatment of the right-hand side will be similar for other schemes, e.g. (2.20), it is not so obvious how one should proceed to solve for  $u_{n+1}$  in the case where the (quadratic) operator to be inverted cannot be factored into the product of linear factors in  $A$  with real coefficients. Thus the applicability of (2.20) in the present setting would require further study, which is not the intention of this work. It is our intention to pursue the study of all of these schemes in the context of time-stepping for second-order hyperbolic equations in a future paper.

#### REFERENCES

- [1] G. A. BAKER, V. A. DOUGALIS AND S. M. SERBIN, *An approximation theorem for second order evolution equations*, Numer. Math., 35, (1980), pp. 127-142.
- [2] ———, *High-order accurate two-step approximations for hyperbolic equations*. RAIRO Anal. Numer., 13 (1979), pp. 201-226.
- [3] J. H. BRAMBLE AND P. H. SAMMON, *Efficient higher order single step methods for parabolic problems, part 1*, Math. Comp., 35 (1980), pp. 655-677.
- [4] V. A. DOUGALIS AND S. M. SERBIN, *Two-step high-order accurate full discretization of second-order hyperbolic equations.*, Proc. 3rd IMACS Symposium, Advances in Computer Methods for Partial Differential Equations, R. Vichnevetsky and R. S. Stepleman, eds., IMACS, Rutgers Univ., New Brunswick, NJ, 1979, pp. 214-220.
- [5] ———, *On the efficiency of some fully discrete methods for second-order hyperbolic equations.*, Comp. Maths. Appl., 7 (1981), pp. 261-279.
- [6] E. HAIRER, *Unconditionally stable methods for second-order differential equations*, Numer. Math., 32 (1979), pp. 373-379.
- [7] N. NASSIF AND J. DESCLOUX, *Stability study for time-dependent linear parabolic equations and its application to Hermitian methods* in Topics in Numerical Analysis, III, J. Miller, ed, Academic Press, New York, 1977.
- [8] S. M. SERBIN, *Rational approximations of trigonometric matrices with applications to second-order systems of differential equations*, Appl. Math. Comput., 5 (1979), pp. 75-92.

## THE SOLUTION OF SINGULAR-VALUE AND SYMMETRIC EIGENVALUE PROBLEMS ON MULTIPROCESSOR ARRAYS\*

RICHARD P. BRENT† AND FRANKLIN T. LUK‡

**Abstract.** Parallel Jacobi-like algorithms are presented for computing a singular-value decomposition of an  $m \times n$  matrix ( $m \geq n$ ) and an eigenvalue decomposition of an  $n \times n$  symmetric matrix. A linear array of  $O(n)$  processors is proposed for the singular-value problem; the associated algorithm requires time  $O(mnS)$ , where  $S$  is the number of sweeps (typically  $S \leq 10$ ). A square array of  $O(n^2)$  processors with nearest-neighbor communication is proposed for the eigenvalue problem; the associated algorithm requires time  $O(nS)$ .

**Key words.** multiprocessor arrays, systolic arrays, singular-value decomposition, eigenvalue decomposition, real symmetric matrices, Hestenes method, Jacobi method, VLSI, real-time computation, parallel algorithms

**1. Introduction.** A singular-value decomposition (SVD) of a real  $m \times n$  ( $m \geq n$ ) matrix  $A$  is its factorization into the product of three matrices:

$$(1.1) \quad A = U \Sigma V^T,$$

where  $U$  is an  $m \times n$  matrix with orthonormal columns,  $\Sigma$  is an  $n \times n$  nonnegative diagonal matrix and the  $n \times n$  matrix  $V$  is orthogonal. This decomposition has many important scientific and engineering applications (cf. [1], [11], [26], [27]). If the matrix  $A$  is square (i.e.,  $m = n$ ) and symmetric, we may adjust the sign of the elements of  $\Sigma$  so that  $U = V$ . We then obtain an eigenvalue decomposition:

$$(1.2) \quad A = U D U^T,$$

where  $U$  is orthogonal and  $D$  diagonal. The advent of massively parallel computer architectures has aroused much interest in parallel singular-value and eigenvalue procedures, e.g. [2], [4], [5], [6], [7], [9], [13], [14], [16], [19], [20], [22], [23], [24], [25]. Such architectures may turn out to be indispensable in settings where real-time computation of the decompositions is required [26], [27]. Speiser and Whitehouse [26] survey parallel processing architectures and conclude that systolic arrays offer the best combination of characteristics for utilizing VLSI/VHSIC technology to do real-time signal processing. (See also [17], [27].)

In this paper we present an array of  $O(n)$  linearly-connected processors which computes an SVD in time  $O(mnS)$ . Here  $S$  is a slowly growing function of  $n$  which is conjectured to be  $O(\log n)$ ; for practical purposes  $S$  may be regarded as a constant (see [21] and the Appendix). Our array implements a one-sided orthogonalization method due to Hestenes [15]. His method is essentially the serial Jacobi procedure for finding an eigenvalue decomposition of the matrix  $A^T A$ , and has been used by Luk [20] on the ILLIAC IV computer. We also describe how one may implement a Jacobi method on a two-dimensional array of processors to compute an eigenvalue

---

\* Received by the editors November 12, 1982, and in revised form August 9, 1983.

† Centre for Mathematical Analysis, Australian National University, GPO Box 4, Canberra, ACT 2601, Australia.

‡ Department of Computer Science, Cornell University, Ithaca, New York 14853. The research of this author was supported in part by the U.S. Army Research Office under grant DAAG 29-79-C0124 and the National Science Foundation under grant MCS-8213718, and in part by the Mathematical Sciences Research Centre and the Centre for Mathematical Analysis, Australian National University.

decomposition of a symmetric matrix. Our array requires  $O(n^2)$  processors and  $O(nS)$  units of time. Assuming that  $S = O(\log n)$ , this time requirement is within a factor  $O(\log n)$  of that necessary for the solution of  $n$  linear equations in  $n$  unknowns on a systolic array [2], [3], [17], [18].

Results similar to ours have been reported in the literature. For computing the SVD, Sameh [23] describes an implementation of Hestenes' method on a ring of  $O(n)$  processors. Suppose  $n$  is even (the result is similar for an odd  $n$ ). At each orthogonalization step  $n/2$  column rotations are performed. Sameh's permutation scheme requires  $3n - 2$  steps to ensure the execution of every possible pairwise rotation at least once; our permutation scheme (presented in § 3) requires only  $n - 1$  steps.

Parallel Jacobi methods for computing eigenvalues are given in [7], [16], [22]. However, the procedure of Sameh [22] may be unsuitable for multiprocessor arrays. For simplicity, assume again that  $n$  is even, so  $n/2$  off-diagonal elements can be set to zero at each elimination step. Let us compare the number of permutations necessary for the annihilation of each off-diagonal element at least once. Our procedure (see §§ 3 and 6) requires  $n - 1$  permutations, which is optimal; that of Chen and Irani [7] requires  $n$  permutations. The scheme of Kuck and Sameh [16] is worse. Their basic scheme appears to cycle every  $2n - 2$  steps and to miss some off-diagonal elements. A modification ("the second row and column are shifted to the  $n$ th position after every  $(n - 1)$  orthogonal transformations") can be made to overcome this problem, but the modified scheme requires  $(n - 1)^2$  permutations [7].

Let us generalize the notion of a "sweep" and use it to denote a minimum-length sequence of rotations that eliminates each off-diagonal element at least once [7]. It is probably fair to assume that the Jacobi procedures in [7], [16] and in this paper require an equal number (say  $S$ ) of sweeps for convergence. For the algorithms presented in this paper a sweep always consists of  $n(n - 1)/2$  rotations (the minimal number possible), but this is not the case for the Chen and Irani or Kuck and Sameh algorithms mentioned above. The architecture proposed in [7] is a linear array of  $O(n)$  processors; the associated Jacobi method requires time  $O(n^2S)$ . The architecture described in [16] is a square array of  $O(n)$  processors, with boundary wraparounds and a broadcast unit. The associated algorithm requires time  $O(n^3S)$ . In comparison, our procedure requires  $O(n^2)$  processors and  $O(nS)$  units of time.

The principal results of this paper were first reported in [4], [5]. A related (generalized) SVD algorithm is presented by the authors and Van Loan in [6]. It requires  $O(n^2)$  processors and  $O(nS)$  time to compute the (generalized) singular values of  $n \times n$  matrices.

This paper is organized as follows. Sections 2–4 are devoted to the singular-value problem and §§ 5–8 to the eigenvalue problem. Hestenes' method is reviewed in § 2. The new ordering is described in § 3 and the corresponding SVD algorithm in § 4. The serial Jacobi method is outlined in § 5. Details are filled in and some variations and extensions of the basic algorithm are given in §§ 7 and 8. The results of some numerical simulations are presented in the Appendix.

The SVD algorithm described in §§ 3–4 below is being implemented on an experimental 64-processor systolic array by Speiser at the Naval Ocean Systems Center (San Diego).

**2. Hestenes' method.** We wish to compute an SVD of an  $m \times n$  matrix  $A$ , where  $m \geq n$ . An idea is to generate an orthogonal matrix  $V$  such that the transformed matrix  $AV = W$  has orthogonal columns. Normalizing the Euclidean length of each nonnull column to unity, we get the relation

$$(2.1) \quad W = \tilde{U}\Sigma,$$

where  $\tilde{U}$  is a matrix whose nonnull columns form an orthonormal set of vectors and  $\Sigma$  is a nonnegative diagonal matrix. An SVD of  $A$  is given by

$$(1.1') \quad A = \tilde{U}\Sigma V^T.$$

As a null column of  $\tilde{U}$  is always associated with a zero diagonal element of  $\Sigma$ , there is no essential difference between (1.1) and (1.1').

Hestenes [15] uses plane rotations to construct  $V$ . He generates a sequence of matrices  $\{A_k\}$  using the relation

$$A_{k+1} = A_k Q_k,$$

where  $A_1 = A$  and  $Q_k$  is a plane rotation. Let  $A_k \equiv (\mathbf{a}_1^{(k)}, \dots, \mathbf{a}_n^{(k)})$  and  $Q_k \equiv (q_{rs}^{(k)})$ , and suppose  $Q_k$  represents a rotation in the  $(i, j)$  plane, with  $i < j$ , i.e.

$$(2.2) \quad \begin{aligned} q_{ii}^{(k)} &= \cos \theta, & q_{ij}^{(k)} &= \sin \theta, \\ q_{ji}^{(k)} &= -\sin \theta, & q_{jj}^{(k)} &= \cos \theta. \end{aligned}$$

We note that postmultiplication by  $Q_k$  affects only two columns:

$$(2.3) \quad (\mathbf{a}_i^{(k+1)}, \mathbf{a}_j^{(k+1)}) = (\mathbf{a}_i^{(k)}, \mathbf{a}_j^{(k)}) \begin{pmatrix} \cos \theta & \sin \theta \\ -\sin \theta & \cos \theta \end{pmatrix}.$$

The rotation angle  $\theta$  is chosen so that the two new columns are orthogonal. Adopting the formulas of Rutishauser [21], we let

$$(2.4) \quad \alpha \equiv \|\mathbf{a}_i^{(k)}\|_2^2, \quad \beta \equiv \|\mathbf{a}_j^{(k)}\|_2^2, \quad \gamma \equiv \mathbf{a}_i^{(k)T} \mathbf{a}_j^{(k)}.$$

We set  $\theta = 0$  if  $\gamma = 0$ ; otherwise we compute

$$(2.5) \quad \xi = \frac{\beta - \alpha}{2\gamma}, \quad t = \frac{\text{sign}(\xi)}{|\xi| + \sqrt{1 + \xi^2}}, \quad \cos \theta = \frac{1}{\sqrt{1 + t^2}},$$

and

$$\sin \theta = t \cdot \cos \theta.$$

The rotation angle always satisfies

$$(2.6) \quad |\theta| \leq \frac{\pi}{4}.$$

However, there remains the problem of choosing  $(i, j)$ , which is usually done according to some fixed cycle. An objective is to go through all column pairs exactly once in any sequence (a sweep) of  $n(n-1)/2$  rotations. A simple sweep consists of a cyclic-by-rows ordering:

$$(2.7) \quad (1, 2), (1, 3), \dots, (1, n), (2, 3), \dots, (2, n), (3, 4), \dots, (n-1, n).$$

Forsythe and Henrici [10] prove that, subject to (2.6), the cyclic-by-rows Jacobi method always converges. Convergence of the cyclic-by-rows Hestenes method thus follows.

Unfortunately, the cyclic-by-rows scheme is apparently not amenable to parallel processing. In § 3 we present an ordering that enables us to do  $\lfloor n/2 \rfloor$  rotations simultaneously. The (theoretical) price we pay is the loss of guaranteed convergence. Hansen [12] discusses the convergence properties associated with various orderings for the serial Jacobi method. He defines a certain "preference factor" for comparing different ordering schemes. Our new ordering is in fact quite desirable, for it asymptotically optimizes the preference factor as  $n \rightarrow \infty$ . Thus, although the convergence proof

of [10] does not apply, we expect convergence in practice to be faster than for the cyclic-by-rows ordering. Simulation results (presented in the Appendix) support this conclusion.

To enforce convergence, we may choose a threshold approach [29, pp. 277–278]. That is, we associate with each sweep a threshold value, and when making the transformations of that sweep, we omit any rotation based on a normalized inner product

$$\frac{\mathbf{a}_i^{(k)T} \mathbf{a}_j^{(k)}}{\|\mathbf{a}_i^{(k)}\|_2 \|\mathbf{a}_j^{(k)}\|_2}$$

which is below the threshold value. Although such a strategy guarantees convergence, we do not know any example for which our new ordering fails to give convergence even without using thresholds. Our method, like the cyclic-by-rows method, is ultimately quadratically convergent [28].

The plane rotations are accumulated if the matrix  $V$  is desired. We compute

$$V_{k+1} = V_k Q_k,$$

with  $V_1 = I$ . Denoting the  $r$ th column of  $V_k$  (respectively  $V_{k+1}$ ) by  $\mathbf{v}_r^{(k)}$  (respectively  $\mathbf{v}_r^{(k+1)}$ ), we may update both  $A_k$  and  $V_k$  simultaneously:

$$(2.8) \quad \begin{pmatrix} \mathbf{a}_i^{(k+1)} & \mathbf{a}_j^{(k+1)} \\ \mathbf{v}_i^{(k+1)} & \mathbf{v}_j^{(k+1)} \end{pmatrix} = \begin{pmatrix} \mathbf{a}_i^{(k)} & \mathbf{a}_j^{(k)} \\ \mathbf{v}_i^{(k)} & \mathbf{v}_j^{(k)} \end{pmatrix} \begin{pmatrix} \cos \theta & \sin \theta \\ -\sin \theta & \cos \theta \end{pmatrix}.$$

**3. Generation of all pairs  $(i, j)$ .** In this section we show how  $O(n)$  linearly-connected processors can generate all pairs  $(i, j)$ ,  $1 \leq i < j \leq n$ , in  $O(n)$  steps. The application to the computation of the SVD and of the symmetric eigenvalue decomposition is described in § 4 and in §§ 6–8, respectively.

First, suppose  $n$  is even. We use  $n/2$  processors  $P_1, \dots, P_{n/2}$ , where  $P_k$  and  $P_{k+1}$  communicate ( $k = 1, 2, \dots, n/2 - 1$ ). Each processor  $P_k$  has registers  $L_k$  and  $R_k$ , output lines out  $L_k$  and out  $R_k$ , and input lines in  $L_k$  and in  $R_k$ , except that out  $L_1$ , in  $L_1$ , out  $R_{n/2}$  and in  $R_{n/2}$  are omitted. The output out  $R_k$  is connected to the input in  $L_{k+1}$  as shown in Fig. 1.

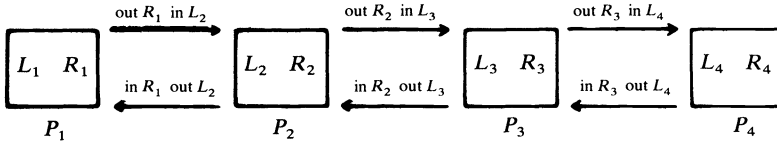


FIG. 1. Inter-processor connections for  $n = 8$ .

Initially  $L_k = 2k - 1$  and  $R_k = 2k$ . At each time step processor  $P_k$  executes the following program:

```

if  $L_k < R_k$  then process  $(L_k, R_k)$  else process  $(R_k, L_k)$ ;
if  $k = 1$  then out  $R_k \leftarrow R_k$ 
  else if  $k < n/2$  then out  $R_k \leftarrow L_k$ ;
if  $k > 1$  then out  $L_k \leftarrow R_k$ ;
{wait for outputs to propagate to inputs of adjacent processors}
if  $k < n/2$  then  $R_k \leftarrow$  in  $R_k$  else  $R_k \leftarrow L_k$ ;
if  $k > 1$  then  $L_k \leftarrow$  in  $L_k$ ;

```



Here “process  $(i, j)$ ” means perform whatever operations are required on the pair  $(i, j)$ ,  $1 \leq i < j \leq n$ . The operation of the systolic array is illustrated in Fig. 2.

We see that the index 1 stays in the register  $L_1$  of processor  $P_1$ . Indices  $2, \dots, n$  travel through a cycle of length  $n-1$  consisting of the registers  $L_2, L_3, \dots, L_{n/2}, R_{n/2}, R_{n/2-1}, \dots, R_1$ . During any  $n-1$  consecutive steps a pair  $(i, j)$  or  $(j, i)$  can appear in a register pair  $(L_k, R_k)$  at most once. A parity argument shows that  $(i, j)$  and  $(j, i)$  cannot both occur (see Fig. 2). Since there are  $n/2$  register pairs at each of  $n-1$  time steps, each possible pair  $(i, j)$ ,  $1 \leq i < j \leq n$ , is processed exactly once during a cycle of  $n-1$  consecutive steps.

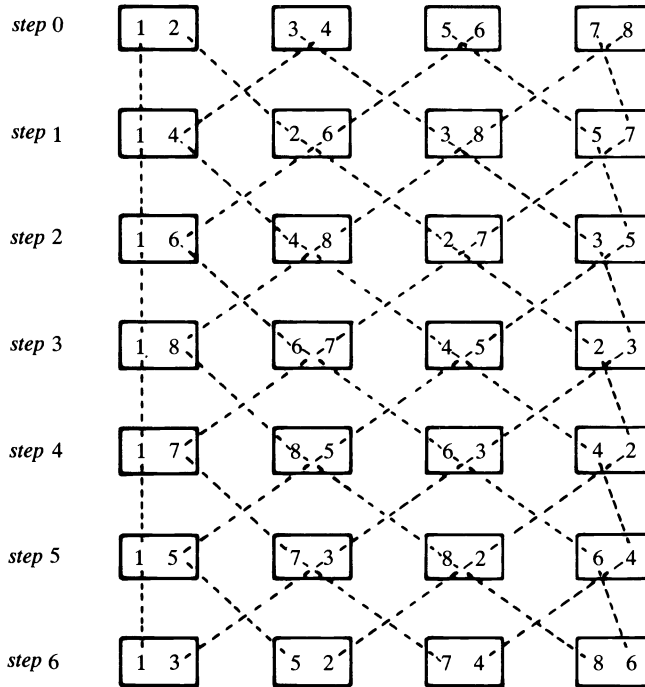


FIG. 2. Full cycle of the systolic array for  $n=8$ .

If  $n$  is odd, we use  $\lceil n/2 \rceil$  processors but initialize  $L_k = 2k-2$ ,  $R_k = 2k-1$  for  $k = 1, \dots, \lceil n/2 \rceil$  and omit any “process” calls from processor  $P_1$ .

It is interesting to note that similar permutations are “well known” for use in chess and bridge tournaments, but have apparently not been applied to parallel computation.

**4. A one-dimensional systolic array for SVD computation.** Assume that  $n$  is even (else we can add a zero column to  $A$  or modify the algorithm as described at the end of § 3). We use  $n/2$  processors  $P_1, \dots, P_{n/2}$ , as described in § 3, except that  $L_k$  and  $R_k$  are now local memories large enough to store a column of  $A$  (i.e.,  $L_k$  and  $R_k$  each has at least  $m$  floating-point words). Shift registers or other sequential access memories are sufficient as we do not need random access to the elements of each column.

Suppose processor  $P_k$  contains column  $\mathbf{a}_i^c$  in  $L_k$  and column  $\mathbf{a}_j^c$  in  $R_k$ . It is clear that  $P_k$  can implement the column orthogonalization scheme in time  $O(m)$  by making one pass through  $\mathbf{a}_i^c$  and  $\mathbf{a}_j^c$  to compute the inner products (2.4), and another pass to

perform the transformations (2.3) or (2.8). Adjacent processors can then exchange columns in the same way that the processors of § 3 exchange indices. This takes time  $O(m)$  if the bandwidth between adjacent processors is one floating-point word. (Alternatively, exchanges can be combined with the transformations (2.3) or (2.8).)

Consequently, we see that  $n/2$  processors can perform a full sweep of the Hestenes method in  $n-1$  steps of time  $O(m)$  each, i.e., in total time  $O(mn)$ . Initialization requires that the  $(2k-1)$ th and  $2k$ th columns of  $A$  be stored in the local memory of processor  $P_k$  for  $k=1, \dots, n/2$ ; clearly this can also be performed in time  $O(mn)$ .

The process is iterative. Suppose  $S$  sweeps are required to orthogonalize the columns to full machine accuracy. We then have a systolic array of  $n/2$  processors which computes the SVD in time  $O(mnS)$ . By comparison, the serial Hestenes algorithm takes time  $O(mn^2S)$ . Our simulation results suggest that  $S$  is  $O(\log n)$ , although for practical purposes we can regard  $S$  as a constant in the range six to ten [21].

After an integral number of sweeps the columns of the matrix  $W \equiv AV$  (see (2.1)) will be stored in the systolic array (two per processor). If  $V$  is required, it can be accumulated at the same time that  $W$  is accumulated, at the expense of increasing each processor's local memory (but the computation time remains  $O(mnS)$ ); see (2.8).

**5. Serial Jacobi method.** We now consider the related problem of diagonalizing a given  $n \times n$  symmetric  $A = A_1$ . The serial Jacobi method generates a sequence of symmetric matrices  $\{A_k\}$  via the relation

$$A_{k+1} = Q_k^T A_k Q_k,$$

where  $Q_k$  is a plane rotation. Let  $A_k \equiv (a_{rs}^{(k)})$  and suppose  $Q_k$  represents a rotation through angle  $\theta$  in the  $(i, j)$  plane, with  $i < j$  (see (2.2)). We choose the rotation angle to annihilate the  $(i, j)$  element of  $A_k$ . If  $a_{ij}^{(k)} = 0$ , we do not rotate, i.e.,  $\theta = 0$ . Otherwise we use the formulas in [21] to compute  $\sin \theta$  and  $\cos \theta$ :

$$(5.1) \quad \begin{aligned} \xi &= \frac{a_{jj}^{(k)} - a_{ii}^{(k)}}{2a_{ij}^{(k)}}, & \cos \theta &= \frac{1}{\sqrt{1+t^2}}, \\ t &= \frac{\text{sign}(\xi)}{|\xi| + \sqrt{1+\xi^2}} = \tan \theta, & \sin \theta &= t \cdot \cos \theta. \end{aligned}$$

Note that the rotation angle  $\theta$  may be chosen to satisfy

$$|\theta| \leq \frac{\pi}{4}.$$

The new matrix  $A_{k+1}$  differs from  $A_k$  only in rows and columns  $i$  and  $j$ . The modified values are defined by

$$(5.2) \quad \begin{aligned} a_{ii}^{(k+1)} &= a_{ii}^{(k)} - t \cdot a_{ij}^{(k)}, \\ a_{jj}^{(k+1)} &= a_{jj}^{(k)} + t \cdot a_{ij}^{(k)}, \\ a_{ij}^{(k+1)} &= a_{ji}^{(k+1)} = 0, \\ \left. \begin{aligned} a_{iq}^{(k+1)} &= a_{qi}^{(k+1)} = \cos \theta \cdot a_{iq}^{(k)} - \sin \theta \cdot a_{jq}^{(k)} \\ a_{jq}^{(k+1)} &= a_{qj}^{(k+1)} = \sin \theta \cdot a_{iq}^{(k)} + \cos \theta \cdot a_{jq}^{(k)} \end{aligned} \right\} (q \neq i, j). \end{aligned}$$

Again we choose  $(i, j)$  in accordance to the new ordering introduced in § 3. The comments that were made in § 2 concerning various aspects (convergence proof,

convergence rate, threshold approach, etc.) of the Hestenes method apply equally well here to the Jacobi procedure.

**6. An idealized systolic architecture.** In this section we describe an idealized systolic architecture for implementing the Jacobi method to compute an eigenvalue decomposition of  $A$ . The architecture is idealized in that it assumes the ability to broadcast row and column rotation parameters in constant time. In § 8 we show how to avoid this assumption, after showing in § 7 how to take advantage of symmetry, compute eigenvectors, etc.

Assume that the order  $n$  is even and that we have a square array of  $n/2$  by  $n/2$  processors, each processor containing a  $2 \times 2$  submatrix of  $A \equiv (a_{ij})$ . Initially processor  $P_{ij}$  contains

$$\begin{pmatrix} a_{2i-1,2j-1} & a_{2i-1,2j} \\ a_{2i,2j-1} & a_{2i,2j} \end{pmatrix} \text{ for } i, j = 1, \dots, n/2,$$

and  $P_{ij}$  is connected to its nearest neighbors  $P_{i\pm 1,j}$  and  $P_{i,j\pm 1}$  (see Fig. 3). In general  $P_{ij}$  contains four real numbers

$$\begin{pmatrix} \alpha_{ij} & \beta_{ij} \\ \gamma_{ij} & \delta_{ij} \end{pmatrix},$$

where  $\alpha_{ij} = \alpha_{ji}$ ,  $\delta_{ij} = \delta_{ji}$  and  $\beta_{ij} = \gamma_{ji}$  by symmetry.

The diagonal processors  $P_{ii}$  ( $i = 1, \dots, n/2$ ) act differently from the off-diagonal processors  $P_{ij}$  ( $i \neq j, 1 \leq i, j \leq n/2$ ). Each time step the diagonal processors  $P_{ii}$  compute rotations

$$\begin{pmatrix} c_i & s_i \\ -s_i & c_i \end{pmatrix}$$

to annihilate their off-diagonal elements  $\beta_{ii}$  and  $\gamma_{ii}$ , (actually  $\beta_{ii} = \gamma_{ii}$ ), i.e., so that  $c_i^2 + s_i^2 = 1$  and

$$\begin{pmatrix} c_i & -s_i \\ s_i & c_i \end{pmatrix} \begin{pmatrix} \alpha_{ii} & \beta_{ii} \\ \gamma_{ii} & \delta_{ii} \end{pmatrix} \begin{pmatrix} c_i & s_i \\ -s_i & c_i \end{pmatrix} = \begin{pmatrix} \alpha'_{ii} & 0 \\ 0 & \delta'_{ii} \end{pmatrix}$$

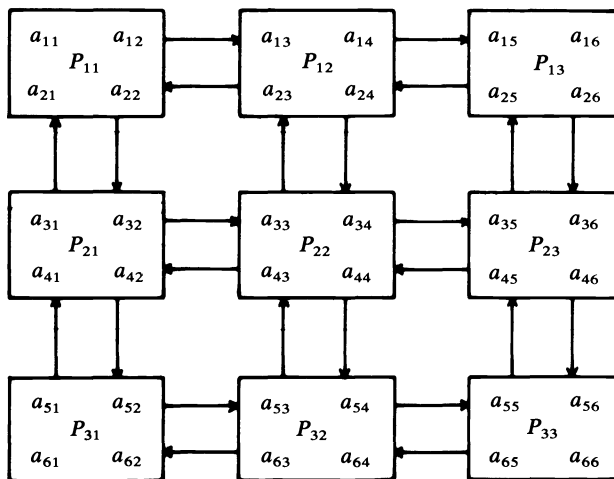


FIG. 3. Initial configuration (idealized,  $n = 6$ ).

is diagonal. From (5.1) and (5.2) with a change of notation we find that

$$(6.1) \quad \begin{pmatrix} c_i \\ s_i \end{pmatrix} = \frac{1}{\sqrt{1+t_i^2}} \begin{pmatrix} 1 \\ t_i \end{pmatrix}$$

and

$$\begin{pmatrix} \alpha'_{ii} \\ \delta'_{ii} \end{pmatrix} = \begin{pmatrix} \alpha_{ii} \\ \delta_{ii} \end{pmatrix} + t_i \beta_{ii} \begin{pmatrix} -1 \\ 1 \end{pmatrix},$$

where

$$(6.2) \quad t_i = \begin{cases} 0 & \text{if } \beta_{ii} = 0, \\ \frac{\text{sign}(\xi_i)}{|\xi_i| + \sqrt{1 + \xi_i^2}} & \text{if } \beta_{ii} \neq 0, \end{cases}$$

and

$$\xi_i = \frac{\delta_{ii} - \alpha_{ii}}{2\beta_{ii}}.$$

To complete the rotations which annihilate  $\beta_{ii}$  and  $\gamma_{ii}$ ,  $i = 1, \dots, n/2$ , the off-diagonal processors  $P_{ij}$  ( $i \neq j$ ) must perform the transformations

$$\begin{pmatrix} \alpha_{ij} & \beta_{ij} \\ \gamma_{ij} & \delta_{ij} \end{pmatrix} \leftarrow \begin{pmatrix} \alpha'_{ij} & \beta'_{ij} \\ \gamma'_{ij} & \delta'_{ij} \end{pmatrix},$$

where

$$\begin{pmatrix} \alpha'_{ij} & \beta'_{ij} \\ \gamma'_{ij} & \delta'_{ij} \end{pmatrix} = \begin{pmatrix} c_i & -s_i \\ s_i & c_i \end{pmatrix} \begin{pmatrix} \alpha_{ij} & \beta_{ij} \\ \gamma_{ij} & \delta_{ij} \end{pmatrix} \begin{pmatrix} c_j & s_j \\ -s_j & c_j \end{pmatrix}.$$

We assume that the diagonal processor  $P_{ii}$  broadcasts the rotation parameters  $c_i$  and  $s_i$  to processors  $P_{ij}$  and  $P_{ji}$  ( $j = 1, \dots, n/2$ ) in constant time, so that the off-diagonal processor  $P_{ij}$  has access to the parameters  $c_i, s_i, c_j$  and  $s_j$  when required. (This assumption is removed in § 8.)

To complete a step, columns (and corresponding rows) are interchanged between adjacent processors so that a new set of  $n$  off-diagonal elements is ready to be annihilated by the diagonal processors during the next time step. This is done in two sub-steps. First, adjacent columns are exchanged as in the SVD algorithm described in §§ 3–4 and as illustrated in Fig. 2. Next, the same permutation is applied to rows, so as to maintain symmetry. Formally, we can specify the operations performed by a processor  $P_{ij}$  with outputs out  $h\alpha_{ij}, \dots$ , out  $h\delta_{ij}$ , out  $v\alpha_{ij}, \dots$ , out  $v\delta_{ij}$ , and inputs in  $h\alpha_{ij}, \dots$ , in  $v\delta_{ij}$  by Program 1. Note that outputs of one processor are connected to inputs of adjacent processors in the obvious way, e.g. out  $h\beta_{ij}$  is connected to in  $h\alpha_{i,j+1}$  ( $1 \leq i \leq n/2, 1 \leq j < n/2$ ): see Fig. 4. In Fig. 4 and elsewhere, we have omitted subscripts  $(i, j)$  if no ambiguity arises, e.g. in  $v\alpha$  is used instead of in  $v\alpha_{ij}$ .

{subscripts  $(i, j)$  omitted if no ambiguity results}  
 {column interchanges}  
 if  $i = 1$  then [out  $h\beta \leftarrow \beta$ ; out  $h\delta \leftarrow \delta$ ]  
 else if  $i < n/2$  then [out  $h\beta \leftarrow \alpha$ ; out  $h\delta \leftarrow \gamma$ ];  
 if  $i > 1$  then [out  $h\alpha \leftarrow \beta$ ; out  $h\gamma \leftarrow \delta$ ];  
 {wait for outputs to propagate to inputs of adjacent processors}  
 if  $i < n/2$  then [ $\beta \leftarrow$  in  $h\beta$ ;  $\delta \leftarrow$  in  $h\delta$ ]  
 else [ $\beta \leftarrow \alpha$ ;  $\delta \leftarrow \gamma$ ];  
 if  $i > 1$  then [ $\alpha \leftarrow$  in  $h\alpha$ ;  $\gamma \leftarrow$  in  $h\gamma$ ];  
 {row interchanges}  
 if  $j = 1$  then [out  $v\gamma \leftarrow \gamma$ ; out  $v\delta \leftarrow \delta$ ]  
 else if  $j < n/2$  then [out  $v\gamma \leftarrow \alpha$ ; out  $v\delta \leftarrow \beta$ ];  
 if  $j > 1$  then [out  $v\alpha \leftarrow \gamma$ ; out  $v\beta \leftarrow \delta$ ];  
 {wait for outputs to propagate to inputs of adjacent processors}  
 if  $j < n/2$  then [ $\gamma \leftarrow$  in  $v\gamma$ ;  $\delta \leftarrow$  in  $v\delta$ ]  
 else [ $\gamma \leftarrow \alpha$ ;  $\delta \leftarrow \beta$ ];  
 if  $j > 1$  then [ $\alpha \leftarrow$  in  $v\alpha$ ;  $\beta \leftarrow$  in  $v\beta$ ];

PROGRAM 1. Column and row interchanges for idealized processor  $P_{ij}$ .

The only difference between the data flow here and that in § 4 is that here rows are permuted as well as columns in order to maintain the symmetry of  $A$  and move the elements to be annihilated during the next time step into the diagonal processors. Hence, from § 3 it is clear that a complete sweep is performed every  $n-1$  steps, because each off-diagonal element of  $A$  is moved into one of the diagonal processors in exactly one of the steps. Each sweep takes time  $O(n)$  so, assuming that  $O(\log n)$  sweeps are required for convergence, the total time required to diagonalize  $A$  is  $O(n \log n)$ .

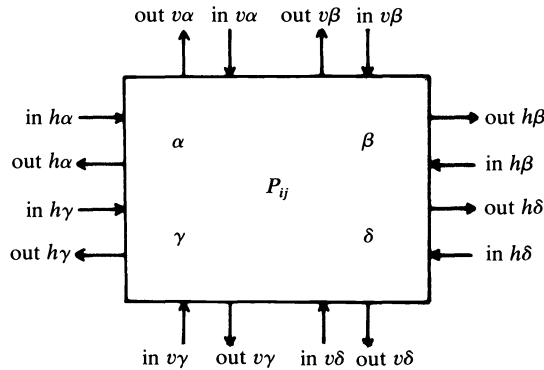


FIG. 4. Input and output lines for idealized processor  $P_{ij}$  with nearest-neighbor connections.

**7. Further details.** Several assumptions were made in § 6 to simplify the exposition. In this section we show how to remove these assumptions (except for the broadcast of rotation parameters, discussed in § 8) and we also suggest some practical optimizations.

**7.1. Threshold strategy.** It is clear that a diagonal processor  $P_{ii}$  might omit rotations if its off-diagonal elements  $\beta_{ii} = \gamma_{ii}$  were sufficiently small. All that is required is to broadcast

$$\begin{pmatrix} c_i \\ s_i \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$$

along processor row and column  $i$ . As discussed in § 2, a suitable threshold strategy guarantees convergence, although we do not know any example for which our ordering fails to give convergence even without a threshold strategy.

**7.2. Computation of eigenvectors.** If eigenvectors are required, the matrix  $U$  of eigenvectors can be accumulated at the same time as  $A$  is being diagonalized. Each systolic processor  $P_{ij}$  ( $1 \leq i, j \leq n/2$ ) needs four additional memory cells

$$\begin{pmatrix} \mu_{ij} & \nu_{ij} \\ \sigma_{ij} & \tau_{ij} \end{pmatrix},$$

and during each step sets

$$\begin{pmatrix} \mu_{ij} & \nu_{ij} \\ \sigma_{ij} & \tau_{ij} \end{pmatrix} \leftarrow \begin{pmatrix} \mu_{ij} & \nu_{ij} \\ \sigma_{ij} & \tau_{ij} \end{pmatrix} \begin{pmatrix} c_j & s_j \\ -s_j & c_j \end{pmatrix}.$$

Each processor transmits its

$$\begin{pmatrix} \mu & \nu \\ \sigma & \tau \end{pmatrix}$$

values to adjacent processors in the same way as its

$$\begin{pmatrix} \alpha & \beta \\ \gamma & \delta \end{pmatrix}$$

values (see Program 1). Initially  $\mu_{ij} = \nu_{ij} = \sigma_{ij} = \tau_{ij} = 0$  if  $i \neq j$ , and  $\mu_{ii} = \tau_{ii} = 1$ ,  $\sigma_{ii} = \nu_{ii} = 0$ . After a sufficiently large (integral) number of sweeps, we have  $U$  defined to working accuracy by

$$\begin{pmatrix} u_{2i-1,2j-1} & u_{2i-1,2j} \\ u_{2i,2j-1} & u_{2i,2j} \end{pmatrix} = \begin{pmatrix} \mu_{ij} & \nu_{ij} \\ \sigma_{ij} & \tau_{ij} \end{pmatrix}.$$

**7.3. Diagonal connections.** In Program 1 we assumed that only horizontal and vertical nearest-neighbor connections were available. Except at the boundaries, diagonal connections are more convenient. This is illustrated in Figs. 5 and 6 (with subscripts  $(i, j)$  omitted).

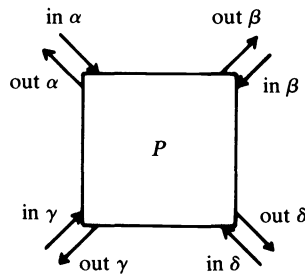


FIG. 5. Diagonal input and output lines for processor  $P_{ij}$ .

Diagonal outputs and inputs are connected in the obvious way, as shown in Fig. 6,

$$\text{e.g. out } \beta_{ij} \text{ is connected to } \begin{cases} \text{in } \gamma_{i-1, j+1} & \text{if } i > 1, j < n/2, \\ \text{in } \alpha_{i, j+1} & \text{if } i = 1, j < n/2, \\ \text{in } \delta_{i-1, j} & \text{if } i > 1, j = n/2, \\ \text{in } \beta_{i, j} & \text{if } i = 1, j = n/2. \end{cases}$$

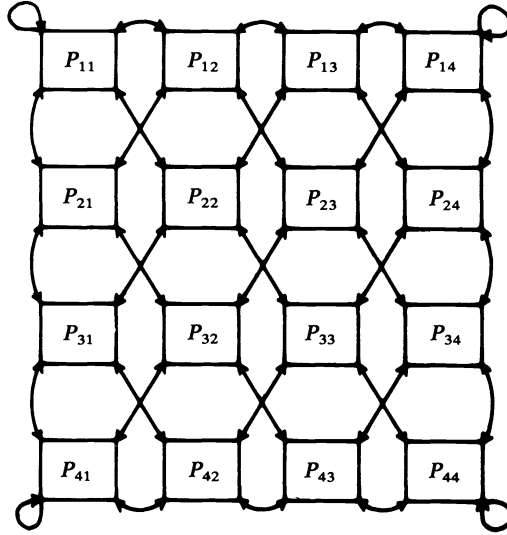


FIG. 6. "Diagonal" connections,  $n = 8$  (here and below  $\leftrightarrow$  stands for  $\rightleftarrows$ ).

Program 2 is equivalent to Program 1 but assumes a diagonal connection pattern as illustrated in Figs. 5 and 6. Subsequently we assume the diagonal connection pattern for convenience, although it can easily be simulated if only horizontal and vertical connections are available.

{subscripts  $(i, j)$  omitted for clarity}

if  $(i = 1)$  and  $(j = 1)$  then  $\left[ \begin{array}{l} \text{out } \alpha \leftarrow \alpha; \text{ out } \beta \leftarrow \beta; \\ \text{out } \gamma \leftarrow \gamma; \text{ out } \delta \leftarrow \delta; \end{array} \right.$

else if  $i = 1$  then  $\left[ \begin{array}{l} \text{out } \alpha \leftarrow \beta; \text{ out } \beta \leftarrow \alpha; \\ \text{out } \gamma \leftarrow \delta; \text{ out } \delta \leftarrow \gamma; \end{array} \right.$

else if  $j = 1$  then  $\left[ \begin{array}{l} \text{out } \alpha \leftarrow \gamma; \text{ out } \beta \leftarrow \delta; \\ \text{out } \gamma \leftarrow \alpha; \text{ out } \delta \leftarrow \beta; \end{array} \right.$

else  $\left[ \begin{array}{l} \text{out } \alpha \leftarrow \delta; \text{ out } \beta \leftarrow \gamma; \\ \text{out } \gamma \leftarrow \beta; \text{ out } \delta \leftarrow \alpha; \end{array} \right.$

{wait for outputs to propagate to inputs of adjacent processors};

$\alpha \leftarrow \text{in } \alpha; \beta \leftarrow \text{in } \beta;$

$\gamma \leftarrow \text{in } \gamma; \delta \leftarrow \text{in } \delta.$

PROGRAM 2. Diagonal interchanges for processor  $P_{ij}$ .

**7.4. Taking full advantage of symmetry.** Because  $A$  is symmetric and our transformations preserve symmetry, only a triangular array of  $(1/2)(n/2)(n/2 + 1) = n(n + 2)/8$  systolic processors is necessary for the eigenvalue computation. In the description above, simply replace any reference to a below-diagonal element  $a_{ij}$  (or processor  $P_{ij}$ ) with  $i > j$  by a reference to the corresponding above-diagonal element  $a_{ji}$  (or processor  $P_{ji}$ ). Note, however, that this idea complicates the programs, and cannot be used if eigenvectors as well as eigenvalues are to be computed. Hence, for clarity of exposition we do not take advantage of symmetry in what follows, although only straightforward modifications would be required to do so.

**7.5. Odd  $n$ .** So far we assumed  $n$  to be even. For odd  $n$  we can modify the program for processors  $P_{1i}$  and  $P_{i1}$  ( $i = 1, \dots, \lceil n/2 \rceil$ ) in a manner analogous to that used in § 3, or simply border  $A$  by a zero row and column. For simplicity we continue to assume that  $n$  is even.

**7.6. Rotation parameters.** In § 6 we assumed that the diagonal processor  $P_{ii}$  would compute  $c_i$  and  $s_i$  according to (6.1), and then broadcast both  $c_i$  and  $s_i$  along processor row and column  $i$ . It may be preferable to broadcast only  $t_i$  (given by (6.2)) and let each off-diagonal processor  $P_{ij}$  compute  $c_b, s_b, c_j$  and  $s_j$  from  $t_i$  and  $t_b$ . Thus communication costs are reduced at the expense of requiring off-diagonal processors to compute two square roots per time step (but this may not be significant since the diagonal processors must compute one or two square roots per step in any case). In what follows a “rotation parameter” may mean either  $t_i$  or the pair  $(c_b, s_b)$ .

**8. Avoiding broadcast of rotation parameters.** The most serious assumption of § 6 is that rotation parameters computed by diagonal processors can be broadcast along rows and columns in constant time. We now show how to avoid this assumption, and merely transmit rotation parameters at constant speed between adjacent processors, while retaining total time  $O(n)$  for the algorithm. We use a special case of a general procedure (due to Leiserson and Saxe) for the elimination of broadcasting.

Let  $\Delta_{ij} = |i - j|$  denote the distance of processor  $P_{ij}$  from the diagonal. The operation of processor  $P_{ij}$  will be delayed by  $\Delta_{ij}$  time units relative to the operation of the diagonal processors, in order to allow time for rotation parameters to be propagated at unit speed along each row and column of the processor array.

A processor cannot commence a rotation until data from earlier rotations is available on all its diagonal input lines. Thus, processor  $P_{ij}$  needs data from processors  $P_{i-1, j-1}, P_{i-1, j+1}, P_{i+1, j-1}$  and  $P_{i+1, j+1}$  if  $1 < i < n/2, 1 < j < n/2$  (for the other cases see § 7.3). Since

$$|\Delta_{ij} - \Delta_{i\pm 1, j\pm 1}| \leq 2$$

it is sufficient for processor  $P_{ij}$  to be idle for two time steps while waiting for the processors  $P_{i\pm 1, j\pm 1}$  to complete their (possibly delayed) steps. Thus, the price paid to avoid broadcasting rotation parameters is that each processor is active for only one third of the total computation time. A similar inefficiency occurs with many other systolic algorithms, [2], [3], [17], [18]. (The fraction one-third can be increased almost to unity if rotation parameters are propagated at greater than unit speed.)

A typical processor  $P_{ij}$  ( $1 < j \leq i < n/2$ ) has input and output lines as shown in Fig. 7 (with subscripts  $(i, j)$  or  $(i, i)$  omitted). Figure 7 differs from Fig. 5 in that it

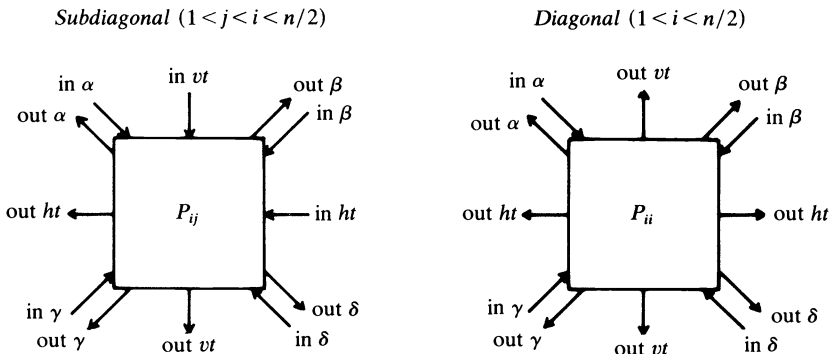


FIG. 7. Input and output lines for typical subdiagonal and diagonal processors.



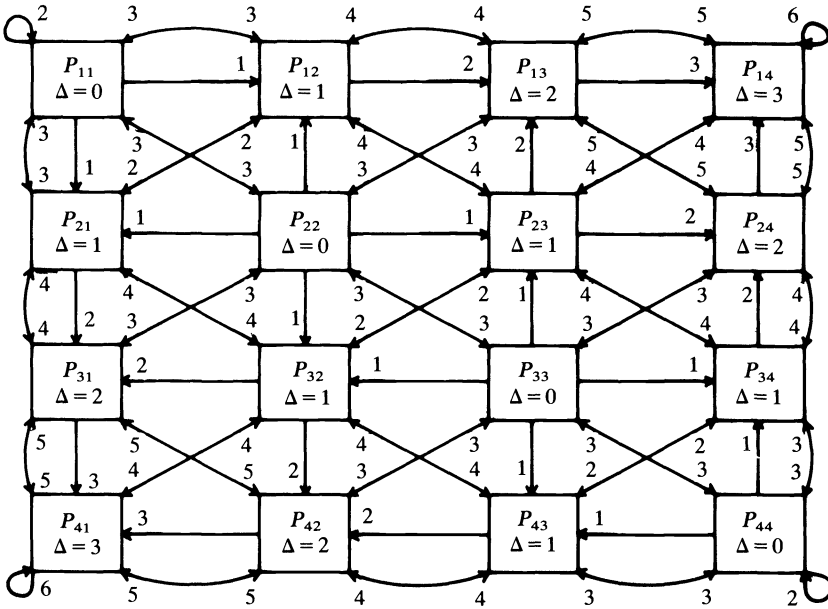


FIG. 8. Interprocessor connections ( $n = 8$ ). (The first times at which inputs are available are indicated.)

shows the horizontal and vertical lines in  $ht$ , out  $ht$ , in  $vt$ , out  $vt$  for transmission of rotation parameters. Processors interconnect as shown in Fig. 8.

Assuming that the array  $(a_{ij})_{1 \leq i, j \leq n}$  is available in the systolic array at time  $T = 0$ , the operation of processor  $P_{ij}$  proceeds as described by Program 3. We assume that each time step has nonoverlapping read and write phases; the result of a write at step  $T$  should be available at the read phase of steps  $T + 1$ ,  $T + 2$  and  $T + 3$  in a neighbouring processor, but should not interfere with a read at step  $T$  in a neighbouring processor. The first time steps at which data are available on various processors' input lines are indicated in Fig. 8.

Program 3 does not compute eigenvectors, but may easily be modified to do so (as outlined in § 7). We have also omitted a termination criterion. The simplest is to perform a fixed number  $S$  (say conservatively 10) sweeps; then processor  $P_{ij}$  halts when  $T = 3S(n - 1) + \Delta_{ij} + 3$ , since a sweep takes  $3(n - 1)$  time steps. A more sophisticated criterion is to stop if no nontrivial rotations were performed during the previous sweep. This requires communication along the diagonal, which can be done in  $n/2$  time steps.

if  $(T \geq \Delta)$  and  $(T - \Delta \equiv 0 \pmod{3})$  then

begin

$$\text{if } T \neq \Delta \text{ then } \begin{bmatrix} \alpha & \beta \\ \gamma & \delta \end{bmatrix} \leftarrow \begin{bmatrix} \text{in } \alpha & \text{in } \beta \\ \text{in } \gamma & \text{in } \delta \end{bmatrix};$$

if  $\Delta = 0$  then {diagonal processor}

begin

$$\text{if } \beta = 0 \text{ then } \xi \leftarrow 0 \text{ else } \xi \leftarrow (\delta - \alpha) / (2 * \beta);$$

$$\text{if } \xi = 0 \text{ then } t \leftarrow 0 \text{ else } t \leftarrow \frac{\text{sign}(\xi)}{|\xi| + \sqrt{1 + \xi^2}};$$

$$t' \leftarrow t;$$

```

 $\alpha \leftarrow \alpha - t * \beta; \delta \leftarrow \delta + t * \beta;$ 
 $\beta \leftarrow 0; \gamma \leftarrow 0$ 
end
else {off-diagonal processor}
begin
 $t \leftarrow \text{in } ht; t' \leftarrow \text{in } vt;$ 
 $c \leftarrow 1/\sqrt{1+t^2}; c' \leftarrow 1/\sqrt{1+t'^2};$ 
 $s \leftarrow t * c; s' \leftarrow t' * c';$ 

$$\begin{pmatrix} \alpha & \beta \\ \gamma & \delta \end{pmatrix} \leftarrow \begin{pmatrix} c & -s \\ s & c \end{pmatrix} \begin{pmatrix} \alpha & \beta \\ \gamma & \delta \end{pmatrix} \begin{pmatrix} c' & s' \\ -s' & c' \end{pmatrix}$$

end;
out  $ht \leftarrow t; \text{out } vt \leftarrow t';$ 
if  $i > j$  then set out  $\beta$  as in Program 2;
if  $i < j$  then set out  $\gamma$  as in Program 2
end
else if  $(T \cong \Delta)$  and  $(T - \Delta \equiv 1 \pmod{3})$  then
begin
if  $(i = 1)$  or  $(j = 1)$  then set out  $\alpha$  as in Program 2;
if  $(i = n/2)$  or  $(j = n/2)$  then set out  $\delta$  as in Program 2
end
else if  $(T \cong \Delta)$  and  $(T - \Delta \equiv 2 \pmod{3})$  then
begin
if  $(i > 1)$  and  $(j > 1)$  then set out  $\alpha$  as in Program 2;
if  $i \leq j$  then set out  $\beta$  as in Program 2;
if  $i \geq j$  then set out  $\gamma$  as in Program 2;
if  $(i < n/2)$  and  $(j < n/2)$  then set out  $\delta$  as in Program 2
end
else {do nothing this time step}.

```

PROGRAM 3. Program for one time step of processor  $P_{ij}$ .

**9. Conclusion.** We have presented a linear array of  $\lceil n/2 \rceil$  processors, each able to perform floating-point operations (including square roots) and having  $O(m)$  local storage, for computing the SVD of a real  $m \times n$  matrix in time  $O(mn \log n)$ . We have also described how a square array of  $\lceil n/2 \rceil$  by  $\lceil n/2 \rceil$  processors, each with similar arithmetical capabilities but with only  $O(1)$  local storage, and having connections to nearest horizontal and vertical (and preferably also diagonal) neighbors, can compute the eigenvalues and eigenvectors of a real symmetric matrix in time  $O(n \log n)$ . The constant is sufficiently small that the method is competitive with the usual  $O(n^3)$  serial algorithms, e.g., tridiagonalization followed by the QR iteration, for quite small  $n$ . The speedup should be significant for real-time computations with moderate or large  $n$ .

The problem of computing eigenvalues and eigenvectors of an unsymmetric real matrix on a systolic array is currently being investigated; unfortunately, the ideas used for symmetric matrices do not all appear to carry over to Eberlein's methods [8] in an obvious way. However, everything that we have said concerning real symmetric matrices goes over with the obvious changes to complex Hermitian matrices.

**Appendix. Simulation results.** We have compared the ordering described in § 3 with the cyclic-by-rows ordering (2.7) by applying the Jacobi method with each ordering to random  $n \times n$  symmetric matrices  $(a_{ij})$ , where the elements  $a_{ij}$  for  $1 \leq i \leq j \leq n$  were

uniformly and independently distributed in  $[-1, 1]$ . (Other distributions were also tried, and similar results were obtained.) The stopping criterion was that the sum  $\sum_{i \neq j} a_{ij}^2$  of squares of off-diagonal elements was reduced to  $10^{-12}$  times its initial value. Table 1 gives the mean number of sweeps  $S_{\text{row}}$  and  $S_{\text{new}}$  for the cyclic-by-rows ordering and the ordering of § 3, respectively, where a "sweep" is  $n(n-1)/2$  rotations. The maximum number of sweeps required for each ordering is given in parentheses in the Table.

TABLE 1  
*Simulation results for row and new orderings.*

$n$	trials	$S_{\text{row}}$	$S_{\text{new}}$
4	5,000	2.96 (4.17)	2.64 (4.00)
6	5,000	3.63 (4.87)	3.37 (4.40)
8	2,000	4.07 (5.04)	3.79 (4.75)
10	2,000	4.39 (5.56)	4.09 (5.47)
20	1,000	5.23 (5.93)	4.94 (5.81)
30	1,000	5.67 (6.62)	5.41 (6.49)
40	1,000	5.92 (6.76)	5.74 (6.54)
50	1,000	6.17 (7.13)	5.99 (6.78)
100	500	6.81 (7.42)	6.78 (7.32)

From Table 1 we see that our new ordering is better than the cyclic-by-rows ordering, perhaps for the reason suggested in § 2, although the difference between the two orderings becomes less marked as  $n$  increases. For both ordering, the number of sweeps  $S$  grows slowly with  $n$ . Empirically we find that  $S = O(\log n)$ , and there are theoretical reasons for believing this, although it has not been proved rigorously. In practice  $S$  can be regarded as a constant (say 10) for all realistic values of  $n$  (say  $n \leq 1,000$ ); see [21]. More extensive simulation results for six different classes of orderings will be reported elsewhere.

**Acknowledgment.** We thank the referees and the editor for their comments, which helped to improve the presentation and make the list of references more complete.

#### REFERENCES

- [1] H. C. ANDREWS AND C. L. PATTERSON, *Singular value decomposition and digital image processing*, IEEE Trans. Acoustics, Speech and Signal Processing ASSP-24 (1976), pp. 26–53.
- [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. Also available as Tech. Report. TR-CS-81-01, Dept. Computer Science, Australian National Univ., 1981.
- [3] R. P. BRENT AND F. T. LUK, *Computing the Cholesky factorization using a systolic architecture*, Proc. 6th Australian Computer Science Conference (1983), pp. 295–302.
- [4] ———, *A systolic architecture for the singular value decomposition*, Tech. Report TR-CS-82-09, Dept. Computer Science, Australian National Univ., August, 1982.
- [5] ———, *A systolic architecture for almost linear-time solution of the symmetric eigenvalue problem*, Tech. Report TR-CS-82-10, Dept. Computer Science, Australian National Univ., 1982.
- [6] R. P. BRENT, F. T. LUK AND C. VAN LOAN, *Computation of the generalized singular value-decomposition using mesh-connected processors*, Proc. SPIE Vol. 431, Real Time Signal Processing VI, SPIE, Bellingham, WA, 1983, pp. 66–71.
- [7] K-W. CHEN AND K. B. IRANI, *A Jacobi algorithm and its implementation on parallel computers*, Proc. 18th Annual Allerton Conference on Communication, Control and Computing, 1980, pp. 564–573.
- [8] P. J. EBERLEIN AND J. BOOTHROYD, *Solution to the eigenproblem by a norm reducing Jacobi type method*, in [30], pp. 327–338.

- [9] A. M. FINN, F. T. LUK AND C. POTTLE, *Systolic array computation of the singular value decomposition*, Proc. SPIE Symp. East 1982, Vol. 341, Real Time Signal Processing V, 1982, pp. 35–43.
- [10] 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.
- [11] G. H. GOLUB AND F. T. LUK, *Singular value decomposition: applications and computations*, ARO Report 77-1, Trans. 22nd Conference of Army Mathematicians (1977), pp. 577–605.
- [12] E. R. HANSEN, *On cyclic Jacobi methods*, J. Soc. Indust. Appl. Math., 11, 1963, pp. 448–459.
- [13] D. E. HELLER AND I. C. F. IPSEN, *Systolic networks for orthogonal equivalence transformations and their applications*, Proc. 1982 Conf. on Advanced Research on VLSI, Massachusetts Institute of Technology, 1982, pp. 113–122.
- [14] ———, *Systolic networks for orthogonal decompositions*, this Journal, 4 (1983), pp. 261–269.
- [15] M. R. HESTENES, *Inversion of matrices by biorthogonalization and related results*, J. Soc. Indust. Appl. Math., 6 (1958), pp. 51–90.
- [16] D. J. KUCK AND A. H. SAMEH, *Parallel computation of eigenvalues of real matrices*, Information Processing 1971, North-Holland, Amsterdam, 1972, pp. 1266–1272.
- [17] H. T. KUNG, *Why systolic architectures*, IEEE J. Comput., (1982), pp. 37–46.
- [18] H. T. KUNG AND C. E. LEISERSON, *Algorithms for VLSI processor arrays*, in Introduction to VLSI Systems, C. Mead and L. Conway, eds. Addison-Wesley, Reading, MA, 1980, pp. 271–292.
- [19] S. Y. KUNG AND R. J. GAL-EZER, *Linear or square array for eigenvalue and singular value decompositions?*, Proc. USC Workshop on VLSI and Modern Signal Processing, Los Angeles, California (Nov. 1982), pp. 89–98.
- [20] F. T. LUK, *Computing the singular-value decomposition on the ILLIAC IV*, ACM Trans. Math. Software, 6 (1980), pp. 524–539.
- [21] H. RUTISHAUSER, *The Jacobi method for real symmetric matrices*, in [30], pp. 202–211.
- [22] A. H. SAMEH, *On Jacobi and Jacobi-like algorithms for a parallel computer*, Math. Comput., 25 (1971), pp. 579–590.
- [23] ———, *Solving the linear least squares problem on a linear array of processors*, Proc. Purdue Workshop on Algorithmically-specialized Computer Organizations, 1982.
- [24] R. SCHREIBER, *Systolic arrays for eigenvalue computation*, Proc. SPIE Symp. East 1982, Vol. 341, Real-Time Signal Processing V, 1982.
- [25] ———, *A systolic architecture for singular value decomposition*, Proc. 1st International Colloquium on Vector and Parallel Computing in Scientific Applications, Paris, Mar., 1983.
- [26] J. M. SPEISER AND H. J. WHITEHOUSE, *Architecture for real-time matrix operations*, Proc. 1980 Government Microcircuits Applications Conference, Houston, TX, Nov., 1980.
- [27] H. J. WHITEHOUSE, J. M. SPEISER AND K. BROMLEY, *Signal processing applications of systolic array technology*, Proc. USC Workshop on VLSI and Modern Signal Processing, Los Angeles, CA, Nov. 1982, pp. 5–10.
- [28] J. H. WILKINSON, *Note on the quadratic convergence of the cyclic Jacobi process*, Numer. Math., 4 (1962), pp. 296–300.
- [29] ———, *The Algebraic Eigenvalue Problem*, Clarendon Press, Oxford, 1965.
- [30] J. H. WILKINSON AND C. REINSCH, eds., *Handbook for Automatic Computation, Vol. 2 (Linear Algebra)*, Springer-Verlag, Berlin, 1971.

## AN EFFICIENT PROGRAM FOR MANY-BODY SIMULATION\*

ANDREW W. APPEL†

**Abstract.** The simulation of  $N$  particles interacting in a gravitational force field is useful in astrophysics, but such simulations become costly for large  $N$ . Representing the universe as a tree structure with the particles at the leaves and internal nodes labeled with the centers of mass of their descendants allows several simultaneous attacks on the computation time required by the problem. These approaches range from algorithmic changes (replacing an  $O(N^2)$  algorithm with an algorithm whose time-complexity is believed to be  $O(N \log N)$ ) to data structure modifications, code-tuning, and hardware modifications. The changes reduced the running time of a large problem ( $N = 10,000$ ) by a factor of four hundred. This paper describes both the particular program and the methodology underlying such speedups.

**1. Introduction.** Isaac Newton calculated the behavior of two particles interacting through the force of gravity, but he was unable to solve the equations for three particles. In this he was not alone [7, p. 634], and systems of three or more particles can be solved only numerically. Iterative methods are usually used, computing at each discrete time interval the force on each particle, and then computing the new velocities and positions for each particle.

A naive implementation of an iterative many-body simulator is computationally very expensive for large numbers of particles, where "expensive" means days of Cray-1 time or a year of VAX time. This paper describes the development of an efficient program in which several aspects of the computation were made faster. The initial step was the use of a new algorithm with lower asymptotic time complexity; the use of a better algorithm is often the way to achieve the greatest gains in speed [2].

Since every particle attracts each of the others by the force of gravity, there are  $O(N^2)$  interactions to compute for every iteration. Furthermore, for the same reasons that the closed form integral diverges for small distances (since the force is proportional to the inverse square of the distance between two bodies), the discrete time interval must be made extremely small in the case that two particles pass very close to each other. These are the two problems on which the algorithmic attack concentrated. By the use of an appropriate data structure, each iteration can be done in time believed to be  $O(N \log N)$ , and the time intervals may be made much larger, thus reducing the number of iterations required. The algorithm is applicable to  $N$ -body problems in any force field with no dipole moments; it is particularly useful when there is a severe nonuniformity in the particle distribution or when a large dynamic range is required (that is, when several distance scales in the simulation are of interest).

The use of an algorithm with a better asymptotic time complexity yielded a significant improvement in running time. Four additional attacks on the problem were also undertaken, each of which yielded at least a factor of two improvement in speed. These attacks ranged from insights into the physics down to hand-coding a routine in assembly language. By finding savings at many design levels, the execution time of a large simulation was reduced from (an estimated) 8,000 hours to 20 (actual) hours. The program was used to investigate open problems in cosmology, giving evidence to support a model of the universe with random initial mass distribution and high mass density.

---

\* Received by the editors March 24, 1983, and in revised form October 1, 1983.

† Computer Science Department, Carnegie-Mellon University, Pittsburgh, Pennsylvania 15213. This research was supported by a National Science Foundation Graduate Student Fellowship and by the office of Naval Research under grant N00014-76-C-0370.

This paper describes the problem and its solution, considered from the point of view of a computer scientist approaching a software engineering problem. Thus, only a brief overview of the physics is given; the emphasis is on techniques of writing efficient software. Section 2 explains the nature of the cosmological questions that can be answered by many-body simulations. Section 3 describes some old algorithms for such simulations, § 4 introduces the data structure and the algorithm to reduce the time per iteration, and § 5 shows how to use the data structure to reduce the number of iterations. Section 6 shows how to create the structure and how to keep it from becoming distorted. Section 7 describes an implementation of the algorithm. The techniques used to attain speedups at various design levels are described. These speedups are summarized and the design methodology leading to them is discussed in § 8.

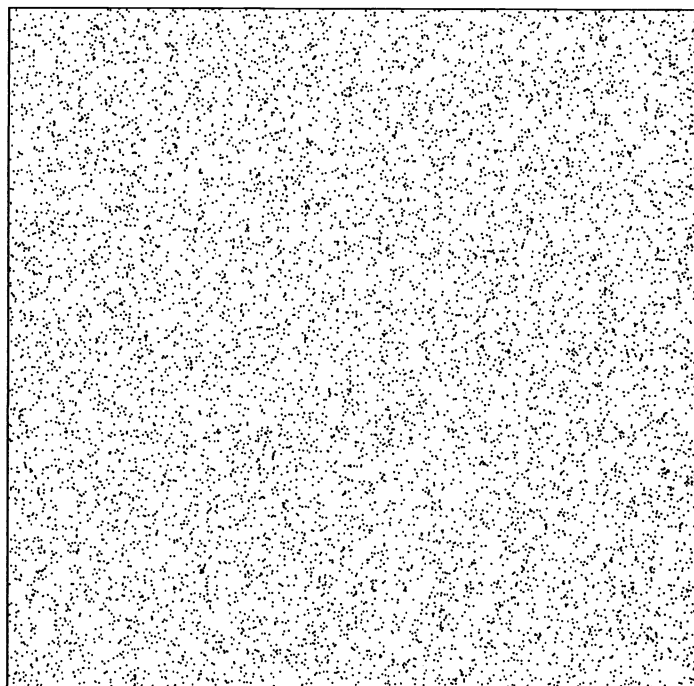
**2. Applications in astrophysics.** The search for a faster algorithm to compute many-body interactions in a gravitational force field was motivated by two important questions in cosmology that can be investigated by simulating gravitational interactions of tens of thousands of galaxies. An efficient computer program has made it feasible to do such simulations. This section describes the cosmological applications, and the remaining sections describe the program.

**2.1. How did galaxies form?** It is generally believed that the early universe was radiation-dominated, that is, that most of the energy of the universe was in the form of photons, and the forces on a typical particle were primarily electromagnetic. The present universe, however, is mass-dominated, with most of the energy condensed into massive bodies (such as stars), and the primary interaction between these bodies being gravitational (the gravitational force between the Earth and Sun, for example, completely dominates the “solar wind” of photons pushing the Earth away from the Sun).

The transition between a radiation-dominated and mass-dominated universe probably took place relatively suddenly; after that, massive bodies such as galaxies began to form (they would have been torn apart in a radiation-dominated universe). Two of the competing theories describing the formation of galaxies [21] may be characterized as “top-down” and “bottom-up,” respectively.

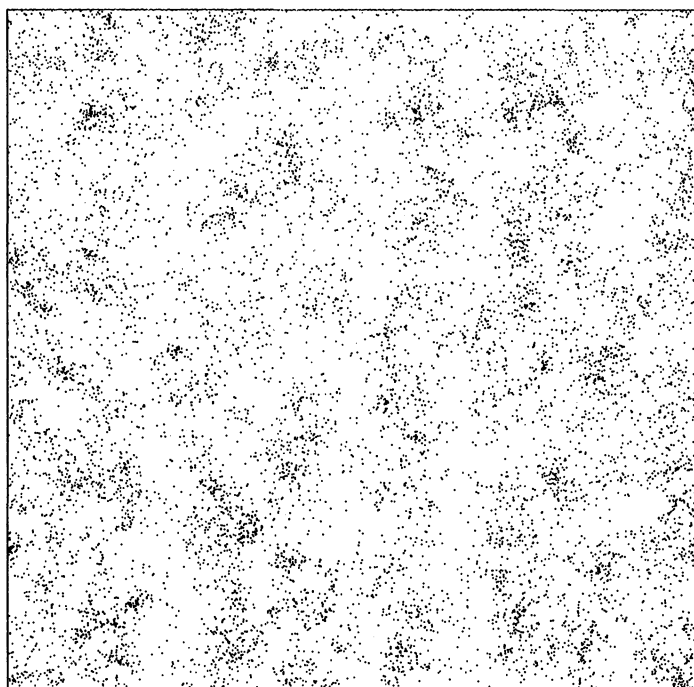
In the “top-down” theory [22], galaxy clusters formed as a result of long-range pressure waves left over from the radiation-dominated universe. A pressure wave contains alternating regions of high and low density. When the universe “condensed” and the radiation disappeared, there would be no medium to support the waves, but the regions of high and low mass-density would remain. It is proposed that the regions of high density became super-clusters of galaxies; that galaxies formed within these super-clusters; and that stars formed within the galaxies. Two-dimensional simulations under these assumptions have shown a cell-like structuring of the clusters [9]; it is not clear whether the dimensionality of the simulation is responsible. It may be that these cells exist in the present universe [14], but the observations at large distances are not conclusive.

In the “bottom-up” theory [17], there were no pressure waves, and the universe immediately after condensation consisted of randomly distributed hydrogen molecules. In a random distribution, there will be local fluctuations in mass density, and as the universe expands, the denser regions will tend to cohere, while the regions of lower density will expand. This will tend to increase the size of the fluctuations, forming stars. More expansion will increase the size of the fluctuations to that of galaxies and eventually of clusters and super-clusters of galaxies. The clusters will have a more random structure than in the “top-down” model.



T = 1.000000

(a)



T = 19.012801

(b)

FIG. 2.1. Result of a simulation. An initial randomly generated configuration of 10,000 galaxies, and the result of simulating the gravitational interactions of this configuration as the universe expands by a factor of 7.12, with mass density  $\rho = \rho_{\text{crit}}$  as a parameter of the simulation. The particles are in a three-dimensional space which has been projected into two dimensions for this picture. A periodic coordinate system is used in which the two extreme points in each dimension are identified. The pictures are scaled to the expansion factor of the simulated universe.

In both theories, the only significant interactions between galaxies after the condensation are gravitational. A simulation of the motion of many particles with gravitational interactions can therefore test these theories. A ten-thousand-galaxy, three-dimensional simulation testing the “bottom-up” theory (that is, starting with a uniform random distribution of particle positions) has been done using the techniques described in the remainder of this paper. The result of the simulation is clustering consistent with that observed by astronomers (see Fig. 2.1 for a picture of the simulation’s output). A similar test of the “top-down” theory has not yet been done, but since this theory differs from the “bottom-up” theory primarily in its specification of the distribution of the initial placement of the particles, it could be simulated easily using the same algorithm.

The large-scale simulations done using the program described in §§ 3 through 6 of this paper seem to imply that the bottom-up model can explain the present mass distribution of the universe quite well, without the complicated assumptions inherent in the top-down model.

**2.2. Is the universe open or closed?** One of the fundamental questions in cosmology is whether the universe will continue expanding forever, or whether it will eventually collapse in a gigantic reversal of the “big bang.” One way to answer this question is to look at the mass density of the universe. If the density of the universe— $\rho$ —is below a certain “critical density”— $\rho_{\text{crit}}$ —then expansion will continue forever; otherwise it will contract. Unfortunately, it is difficult to measure the mass density of the universe. Astrophysicists have been able to make estimates; most observational estimates put the mass density at about a tenth of the critical density. There are reasons for believing that the  $\rho/\rho_{\text{crit}}$  should be exactly equal to 1. For example, any deviation of  $\rho$  from  $\rho_{\text{crit}}$  in the early universe ( $t = 1$  sec) would be enormously amplified as the universe expanded, making the current measured value of  $\rho = 0.1\rho_{\text{crit}}$  far too close to have happened by chance [8].

The astronomical search for the “missing mass”—to determine whether  $\rho/\rho_{\text{crit}}$  is closer to 0.1 or to 1.0—is complicated by the fact that many forms of mass (such as black holes) are difficult to observe directly. This problem can be avoided by approaches that do not involve direct observation of the mass density. One such approach is through simulation of the gravitational interactions of galaxies under different assumptions about the mass density. Groth et al. [12] observed in small simulations that low mass densities will not lead to the amount of clustering actually observed, and that the critical density would lead to such clustering. The ten-thousand-body, three-dimensional simulation using the program described later in this paper was for the higher-density case; large-scale clustering was observed, lending support to this theory. The lower-density case can be examined by the same techniques.

**3. Previous algorithms.** Because the  $N$ -body problem cannot be done in closed form, the calculation must be done numerically. That is, at each time  $t$ , the gravitational forces of each mass on each of the others may be computed by Newton’s laws. (For an appropriate range of distances—say, between one and a few hundred million light-years—Newton’s laws are a good approximation to general relativity [17].) Using the inverse-square force law, an approximation to the true acceleration and velocity of each particle over a time  $dt$  can be computed. By many iterations of this method, the position of each particle after an arbitrary length of time may be found.

**3.1. A simple algorithm.** Newton’s law of gravity states that the force between any pair of particles is proportional to the product of their masses divided by the



square of the distance between them. Stated as a vector equation,

$$m_i \mathbf{r}_i'' = \frac{G m_i m_j (\mathbf{r}_j - \mathbf{r}_i)}{|\mathbf{r}_j - \mathbf{r}_i|^3},$$

where  $\mathbf{r}_i$  is the position vector of particle  $i$ ,  $\mathbf{r}_i''$  is the acceleration vector of particle  $i$ , and  $G$  is the universal gravitational constant.

When there are many particles, the acceleration of each particle is given by the sum of the accelerations (as computed by Newton's law) induced by all the other particles. This is simply a large set of differential equations. For two bodies, it is solvable in closed form; however, for more than two bodies no closed form solution exists.

The differential equation can be integrated numerically using a "naive" algorithm. At each iteration, compute the acceleration acting upon each particle; from this, compute a modified velocity over the next time increment, and then compute the position of each particle at the end of the time increment by calculating

$$\mathbf{r}_{\text{new}} = \mathbf{r}_{\text{old}} + \mathbf{v} \cdot dt.$$

The time increment  $dt$  must be made small enough that the accelerations do not greatly change between  $t$  and  $t + dt$ .

There are two problems with this algorithm. The first is that the number of interactions is large as a function of the number of particles. In particular, the gravitational action of each particle on every other particle must be computed every iteration, requiring a total of  $N^2 - N$  operations. When  $N$  is large (physicists would like to simulate tens of thousands of particles, although they are rarely able to do so), an  $O(N^2)$  algorithm is extremely costly to execute.

The second problem in many-body simulations is that it usually happens that some pairs of particles in such a system will pass very close to each other. Nearby particles in a gravitational field usually move at high speed with respect to each other; the combination of high velocities and small distances necessitates an extremely small time increment between iterations.

One approach to these problems is to use an extremely fast computer. The Cray-1 computer is very fast at algorithms that have a "vectorizable" formulation: that is, problems which can be expressed in terms of element-by-element arithmetic operations on long arrays of numbers. The acceleration computation can be formulated in terms of such large vectors. If the vector instructions of the Cray-1 are used to advantage (either by hand-coding, or by using the Cray Fortran compiler with a good understanding of what sorts of programs the compiler can generate efficient code for), the time required to calculate the acceleration between two bodies can be estimated at 100 clock cycles (40 of which are needed for a calculation of a periodic distance function peculiar to the many-galaxy problem [3]). The time for one clock cycle is 12 nanoseconds [19], and the number of pairs of bodies is  $N^2/2$ , so the time for one iteration can be estimated at  $.6N^2$  microseconds. Using scalar instructions, or using vector instructions with inefficient pipeline behavior, would more than double the time taken per iteration.

Using a similar program to simulate ten thousand bodies over one thousand iterations requires approximately 8,000 hours of VAX time (this was extrapolated from observations of 100-particle simulations). Table 3.1 gives the times required for various implementations of a straightforward simulator. Even on a fast vector processor like the Cray-1, this simulation takes several hours. The disadvantage to running the

simulation on the Cray computer is that the Cray-1 is enormously expensive: at a cost of eight to ten million dollars it is about 40 times as expensive as a large minicomputer such as a VAX. A solution whereby the problem can be solved in tens of hours on the VAX would obviously be preferable to any of the points in the solution space described in the table below.

TABLE 3.1  
*Running times, in hours, of an  $O(N^2)$  program for 10,000 bodies over 1,000 iterations.*

	VAX-11/780	Cray-1 (estimated)
Optimizing compiler	8000	30
Hand-optimized	5000	16

**3.2. Other algorithms in the literature.** Two approaches have been taken to reduce the cost of the acceleration calculation in the N-body problem. One approach is to represent the problem in a position-velocity phase space, and transform the force field using a fast Fourier transform into a form where it can be applied in linear time [15, 16]. This takes  $O(N \log N)$  time (dominated by the Fourier transform) per iteration. However, the phase space must be discrete. This means that all positions must be multiples of some lattice size  $a$ , and that all velocities must be less than some maximum  $f$ . Thus, the (physically interesting) effects of tight clusters cannot be modeled.

Another approach is to keep track, for each particle, of the sets of “nearby” particles and “faraway” particles [1]. The “faraway” particles may be integrated with larger time-steps than the “nearby” particles. When the particles are uniformly distributed, this has an asymptotic complexity of  $O(N^{1.5})$ . Unfortunately, when clustering occurs, the number of “nearby” particles is in the same order of magnitude as the total number of particles, and the asymptotic complexity is again  $O(N^2)$ . The problem of small time-steps is attacked by using a special-case technique for close two-body interactions, but this technique cannot be applied for tight clusters of three or more particles.

Another similar approach is to divide the universe into cells, computing the particle-particle interactions within the cell, and then the cell-cell interactions [13]. This has complexity  $O(N^{4/3})$  for a uniform distribution. A variant of this method is to compute the cell-cell interactions by a fast Fourier transform, reducing the complexity to  $O(N \log N)$ . Both variants degrade to a quadratic time-complexity when severe clustering occurs.

These algorithms are great improvements on the “naive” algorithm, especially for those problems with a relatively uniform mass distribution. (Problems in plasma physics are often of this nature, as are some problems in astrophysics.) However, when a greatly nonuniform mass density is to be simulated, their asymptotic complexity approaches that of the “naive” algorithm. With none of these algorithms is the problem of the vanishingly small discrete time-step solved; in the discrete phase-space approach, the time steps cannot be made smaller and thus information is lost, while in the second and third approaches, the problem is essentially the same as with the “naive” algorithm.

**4. Reducing the complexity of each iteration.** To compute the force of gravity on an apple exerted by the Earth, it suffices to treat the Earth as a point mass; it is not necessary to sum the forces exerted by each atom of the Earth. This is a consequence

of the spherical symmetry of the Earth; Newton invented the integral calculus to prove this fact.

When an attracting body is not spherically symmetric, the result obtained by treating it as a point mass is no longer exact, but it is a good approximation. This approximation—in which one attraction between a pair of point masses is calculated, rather than all the attractions between all their constituent particles—is the key to reducing the asymptotic complexity of computing the accelerations from  $O(N^2)$  to  $O(N \log N)$ .

**4.1. The monopole approximation.** A divide-and-conquer algorithm can solve the many-body problem in  $O(N \log N)$  time per iteration, and requires significantly fewer iterations. The computational complexity has not been proved, but a reasonable argument is given; furthermore, experience with an implementation of the algorithm has shown that it runs as quickly as expected.

The algorithm relies upon the following approximation: suppose there are two particles,  $m_1$  and  $m_2$ , each no more than  $dr$  from their center of mass (see Fig. 4.1). The gravitational attraction they exert upon an observer situated a distance  $r$  from the center of mass will be

$$\mathbf{g} = \frac{Gm_1(\mathbf{r} + \mathbf{dr}_1)}{|\mathbf{r} + \mathbf{dr}_1|^3} + \frac{Gm_2(\mathbf{r} + \mathbf{dr}_2)}{|\mathbf{r} + \mathbf{dr}_2|^3} = \frac{G(m_1 + m_2)\mathbf{r}}{|\mathbf{r}|^3} + O(dr^2).$$

Because there is no term in  $dr$  in this equation, the approximation is good to first order.

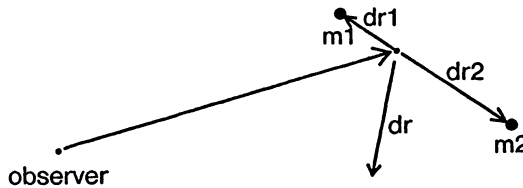


FIG. 4.1. *The monopole approximation.*

Now consider the arrangement of masses shown in Fig. 4.2, which we will suppose to be a subset of the particles in a many-body simulation. To compute the acceleration of each particle on every other, we may break the computation into three parts: those interactions of two particles which are in the left-hand clump, those interactions of which both particles are in the right-hand clump, and the interactions of a particle from each clump. The latter interactions may be approximated to order  $(dr/r)^2$  by using the approximation described in the previous paragraph: by computing one interaction, as if each of the two clumps were one large mass. The number of computations required to calculate the intra-clump interaction has thus been reduced from  $n_1 \cdot n_2$  to 1; the intra-clump calculation remains unchanged.

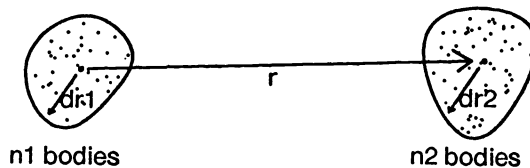


FIG. 4.2. *Two clumps to which the approximation can be applied.*

Had the two clumps been closer together, then the approximation would no longer have been as good, since it depends on the value of  $dr/r$ . In that case, more calculations would have had to be done.

**4.2. A data structure.** A method is needed for finding subsets of the particles for which the approximation can be made. This is made easier by the introduction of an appropriate data structure—a binary tree whose leaves are particles and whose internal nodes represent clumps of particles. Every node will have an associated mass and position. The leaves will have the mass and position of the particles they represent; each internal node will have a mass equal to the sum of the masses of its two child nodes, and a position equal to the center of mass of its child nodes. Also associated with each clump (internal node) will be the approximate radius of the clump.

It is now a simple matter to compute all of the gravitational interactions between two clumps that are small relative to their separation, that is,

$$dr_1/r < \delta \quad \text{and} \quad dr_2/r < \delta$$

for some fixed criterion of accuracy  $\delta$ . The parameters  $dr_1$  and  $dr_2$  are stored in the tree; the positions need only be subtracted and multiplied by the total masses of each clump (also stored in the tree).

If the accuracy criterion is not satisfied (that is, if the clumps are large and close together), then the calculation of the interaction of each of the two subclumps of one clump with each of the two subclumps of the other clump must be made. It is not always necessary to “break up” both clumps for this calculation; see Fig. 4.3 for an example in which one clump satisfies the criterion and need not be split, while the other clump is split into two pieces.

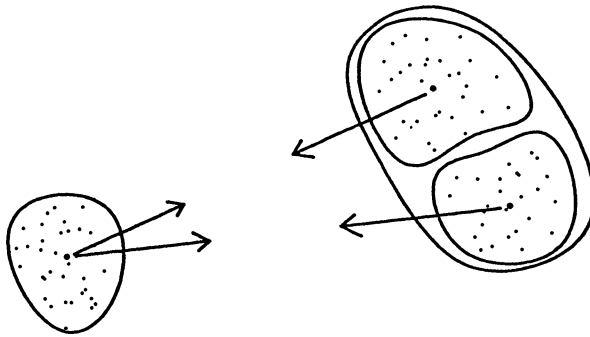


FIG. 4.3. An example of the calculation of clump interaction.

**4.3. The algorithm.** This algorithm can be coded as the following pair of pseudo-Pascal recursive procedures—procedure `ComputeAccel` computes all of the accelerations internal to one clump, and procedure `TwoNode` computes the interactions between two clumps.

```

procedure ComputeAccel (B)
  begin if B is a nontrivial clump
    then begin ComputeAccel(Bleft-child)
              ComputeAccel(Bright-child)
              TwoNode(Bleft-child, Bright-child)
    end
  end

```

```

procedure TwoNode( $A, B$ )
  begin  $\mathbf{d} \leftarrow \mathbf{r}_B - \mathbf{r}_A$ 
    if ( $dr_A/d > \delta$ ) and ( $dr_A > dr_B$ )
      then begin TwoNode( $A_{\text{left-child}}, B$ )
                TwoNode( $A_{\text{right-child}}, B$ )
      end
    else if  $dr_B/d > \delta$ 
      then begin TwoNode( $A, B_{\text{left-child}}$ )
                TwoNode( $A, B_{\text{right-child}}$ )
      end
    else begin  $\text{Acc}_A \leftarrow \text{Acc}_A + Gm_B \mathbf{d} / d^3$ 
               $\text{Acc}_B \leftarrow \text{Acc}_B - Gm_A \mathbf{d} / d^3$ 
    end
  end

```

One detail that for clarity has so far been omitted from the description of the algorithm pertains to the representation of position, velocity, and acceleration vectors. Rather than storing at each node the absolute position of the clump associated with that node, the position vector from the node's parent to the node is stored. (The same applies to velocities and accelerations.) This is done in order to minimize round-off errors in subtractions, which will be discussed in § 7. The absolute position of a particle or clump may be computed by taking the sum of the position offsets of all its ancestors up to the root, though it is rarely necessary to compute absolute positions. Note that the algorithm assigns accelerations throughout the data structure, taking advantage of the relativization of acceleration vectors.

**4.4. Analysis of time complexity.** If the parameter  $\delta$  is set to zero, then the TwoNode procedure will always recur down to the level of individual particles, and the accelerations assigned to the internal nodes will be zero. If  $\delta$  is not equal to zero, then the absolute acceleration of a single particle will be an approximation to the true acceleration. For values of  $\delta$  between 0 and 1, the time complexity of ComputeAccel is estimated (and observed) to be  $O(N \log N)$ .

To see this, consider the number of times a particle  $X$  is compared with other clumps for the purposes of adding to an acceleration vector. Suppose there is a spherical shell around  $X$  of radius  $r$  and thickness  $\delta \cdot r$ . If this shell is filled with clumps of diameter  $\delta \cdot r$ , then there will be  $4/\delta^2$  clumps in the shell. The smallest sphere will have a size such that the expected number of galaxies contained within it is 1; the largest will enclose a volume such that the expected number of galaxies within it is  $N$ . The quotient of the radii of the largest and smallest spheres will therefore be  $N^{1/3}$ . This will be equal to  $(1+\delta)^k$ , where  $k$  is the number of shells. Then  $k = \log(N)/3 \log(1+\delta)$ , and the number of clumps for which there must be calculation of accelerations with respect to particle  $X$  is approximately

$$\frac{4 \log N}{3\delta^2 \log(1+\delta)}$$

Note that this number overestimates the number of calculations done, in that some of the calculation will involve not the comparison of  $X$  with another clump, but the comparison of an enclosing clump of  $X$  with another clump. That calculation would also be counted in this analysis as a calculation for  $X$ 's sibling clump, and all other subclumps of the encompassing clump. However, this will do no more than change

the constant of proportionality: for each of the  $N$  galaxies,  $O(\log N)$  calculations must be done, giving a total execution time—for fixed  $\delta$ —of  $O(N \log N)$ .

**4.5. Accuracy of the algorithm.** The parameter  $\delta$  is a measure of the accuracy of the calculation. When one clump is compared with another, and the ratio of diameter to separation is less than  $\delta$ , then the computed acceleration will have a fractional error less than  $\delta^2$ . When all the accelerations that clump  $X$  feels from other clumps are summed, the error in acceleration should be proportional to  $\delta^2$  divided by the square root of the number of clumps compared with (assuming random directions of the error vector). A more intuitive explanation of this statistical argument is that larger clumps will tend to approach some sort of spherically symmetric distribution, simply because of the large number of randomly positioned particles. In a perfectly spherical distribution, the error made in assuming that all the mass is positioned at the center is exactly zero. Thus the error in acceleration, on the average, should be significantly less than  $\delta^2$ .

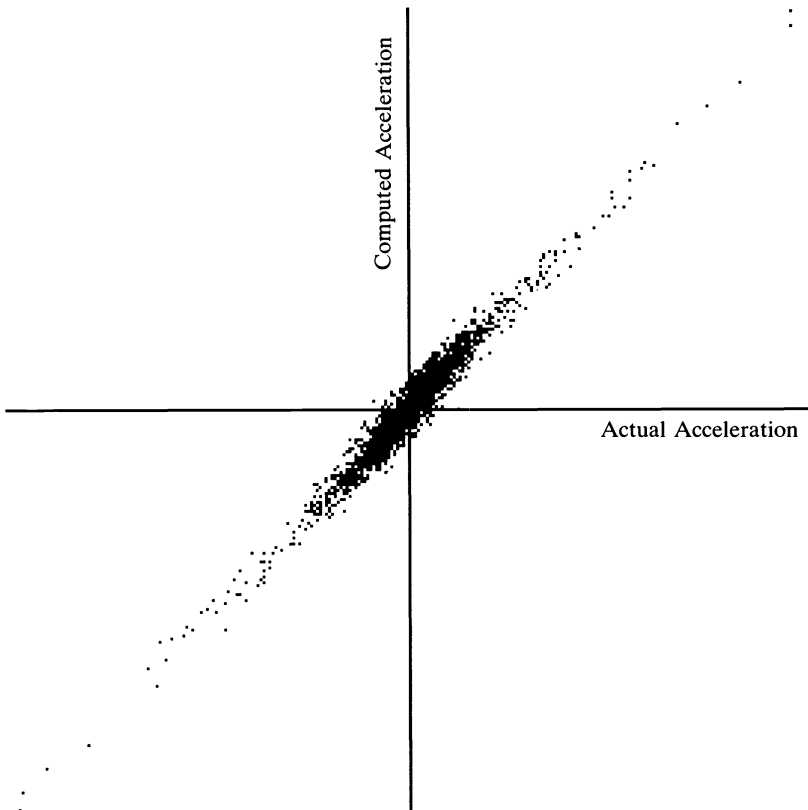


FIG. 4.4. Scatterplot of components of actual vs. computed accelerations for  $\delta = 0.3$ .

In fact, in the distribution of errors (shown in Fig. 4.4) there is a maximum absolute error range, such that for most particles the error is quite small on an absolute scale. For particles with large accelerations, the proportional error is practically zero. Figure 4.4 was computed by taking a random distribution of particles and using the (exact) results computed by running the algorithm with  $\delta = 0$  as the “actual” acceleration components, and using the results computed with  $\delta = 0.3$  as the “Computed” acceler-

ation components. The absolute errors are the deviations from the line  $y = x$ ; the scatterplot shows a good bound on the absolute error.

In those calculations where the exact final positions of the particles is not as important as statistics about their configurations, a relatively large value of  $\delta$  can be used (such as  $\frac{1}{2}$ ), greatly reducing the constant factor in the running time of the  $O(N \log N)$  program.

It is useful to note that although the  $O(N^2)$  algorithm has theoretically complete accuracy in computing accelerations, the fact that the time intervals must be made discrete introduces approximations into any numerical calculation of the  $N$ -body problem. By choosing the parameters so that the errors introduced by each part (the clump approximation and the discrete-time approximation) are equal, the resulting error is about equal to that of the standard algorithm.

Since the use of a clumping algorithm to study the formation of galaxy clusters might conceivably be a cause of systematic error, the result of a simulation using this algorithm in which no clustering occurred is of interest. In this simulation, the galaxies were given higher initial velocities than predicted by theory, and no measurable clustering occurred (as seen both by the human eye and by a correlation function of interparticle distance).

**5. Reducing the number of iterations.** When two particles come very close to each other in an inverse-square force field, their accelerations become extremely high. To model their behavior accurately, extremely small time steps are required. In any simulation with a large number of particles, there are bound to be a few such pairs at any given time; these pairs require the time increments of the simulation to be so small that the number of iterations required to integrate over a significant interval of time becomes prohibitively large.

One widely used solution to this problem modifies the force law to limit the accelerations at small distances. The inherent problem with this approach in the modeling of galaxy clustering is that the clustering occurs (and should be examined by the simulation) over all distance scales. To tamper with the force law at small distances makes any conclusions about clustering at these distances suspect.

Fortunately, the data structure introduced in the previous section leads to a solution to this problem that preserves the inverse-square properties of the force law at all distance scales. In § 5.1 an aspect of the calculation open to algorithmic attack is described, and the attack itself is explained in §§ 5.2 and 5.3.

**5.1. Characteristic times.** The time increment  $dt$  between iterations is determined after each iteration. The usual approach is to use a global  $dt$  for all particles. In order to avoid gross inaccuracies at very small distances, the minimum characteristic time over all particles must be used for  $dt$ . The characteristic time of an object is a measure of how long it takes for that object's acceleration to change significantly; the time will be much shorter for a particle tightly orbiting a neighbor. The occasional tight pairs and threesomes require an expensively small value for  $dt$  in the naive algorithm.

The characteristic time for a clump  $C$  is the time in which a child of  $C$  will move a distance of approximately  $\delta$  times the child's distance from  $C$ 's center of mass. The characteristic time could come from a high velocity relative to distance ( $t_v$ ) or from a high acceleration ( $t_a$ ). This is easy to calculate, since the position vector of each child is stored as the vector from (the center of mass of)  $C$ . So the characteristic time of  $C$  is the minimum over both children of  $t_v$  and  $t_a$ , where

$$\delta \times |P| = t_v \times |V|, \quad \delta \times |P| = |A| \times \frac{1}{2} t_a^2.$$

(Note that  $P$ ,  $V$ , and  $A$  are the position, velocity, and acceleration vectors of the children relative to the center of mass of  $C$ .) In each iteration, the accelerations are computed by `ComputeAccel` and the minimum characteristic time  $dt$  is found. The procedure `Move` calculates the new velocities and positions:

$$\mathbf{V}_{\text{new}} = \mathbf{V}_{\text{old}} + \mathbf{A} \cdot dt, \quad \mathbf{P}_{\text{new}} = \mathbf{P}_{\text{old}} + \mathbf{V}_{\text{new}} \cdot dt.$$

There are schemes such as Richardson extrapolation [6] which improve on this “naive” method of integration and allow the use of a larger timestep with great accuracy. However, the timestep will still be no larger than that allowed by setting  $\delta$  to 2 in the equations for  $(t_v)$  and  $(t_a)$ , whereas in a typical simulation using the approximation algorithm  $\delta$  will be on the order of a tenth anyway. Furthermore, the extremely high accuracy produced by this method is not particularly useful in the presence of the acceleration approximation. Finally, using the extrapolation method would attack the wrong problem: the characteristic time is so small not because  $\delta$  is very small, but because of the existence of a few extremely “tight” clumps where there is a very high ratio of velocity to separation.

Calculating the minimum characteristic time of the entire universe leads to an exceedingly small  $dt$ . Suppose two or three galaxies get into a tight orbit around each other; their characteristic time may be an order of magnitude shorter than the characteristic time of any other object in the universe.

It would be nice to be able to iterate small, very tight clusters at shorter time intervals than the rest of the universe, saving a large amount of calculation. This is not too difficult; what is needed is a concise criterion to distinguish such clumps.

**5.2. Indivisible clumps.** Let such a clump be considered to be one object, indivisible, of nonzero radius. Indivisibility will be defined as follows: a clump is indivisible if for *all* clumps outside it, its ratio of size to distance is less than  $\delta$ . What indivisibility effectively means is that an indivisible clump will never—throughout the course of the acceleration calculations for one iteration—be “split” by procedure `TwoNode` to calculate accelerations of its subclumps with respect to any other clump. This is easy to detect—simply mark clump  $A$  in the first **then** clause or clump  $B$  in the second **then** clause of procedure `TwoNode`. Any clump that is never marked during the process of computing all the accelerations is indivisible.

The reason that this criterion is chosen is that it characterizes very well the set of clumps such that the external gravitational field acting upon them is an almost constant function of position within the clump. In fact, the monopole approximation has the effect of assuming that this field *is* constant, and the improved moving algorithm described below takes advantage of this fact.

Procedure `Move`, procedure `ComputeAccel`, and the procedure that determines  $dt$  will be altered so that they never look at the internal structure of such a clump. Note that `TwoNode` need not be altered, since the way indivisible clumps are defined implies that `TwoNode` never looks at their internal structure. Now the problem is gone: the small, tight cluster of galaxies has become a point (although with nonzero radius). The time increment  $dt$  will be much larger than it could have been otherwise.

The internal motions and accelerations of these tight clumps will have to be computed every iteration, and in fact it will take several iterations of the tight clump to compute its motion over the time interval  $dt$ . However, these iterations of three or four objects are replacing iterations over the entire universe.

**5.3. Closed form calculations.** When an indivisible object itself is a clump containing two indivisible subclumps (these will usually be simply individual galaxies), then



its orbit may be solved in closed form. In this case, the calculations to resolve internal motion may be postponed until another clump gets near enough to see the internal structure of the object. This may be many iterations of the universe later—and many times more iterations of the tight pair, which typically has a much shorter characteristic time. Only one calculation needs to be made in closed form to replace these many iterations; furthermore, this calculation will be exceedingly accurate, since no approximations are being made internally to the subsystem.

Since indivisibility may occur at several distance scales (indivisible clumps may contain clumps which themselves contain indivisible clumps, and so on), the tight-clump calculations (of which the two-body closed form calculation is a special case) may be done recursively.

**6. Managing the data structure.** The efficiency of all parts of the algorithm depends on having the structure of the tree of clumps accurately reflect the structure of the particles in the simulated space. Under the influence of gravity, the particles move, distorting the tree. The structure must be maintained and the distortions removed regularly. Fortunately, this can be done in a simple way.

**6.1. Reorganizing the tree.** After moving clumps that are not indivisible, the coordinates of a clump will no longer correspond exactly to the center of mass of the two subclumps. This is due to a nearby object attracting one subclump more strongly than the other. It is a simple matter, however, to adjust the position of each clump after its subclumps have been moved. Sometimes, however, another subclump will intrude into a clump so that the clumps no longer represent disjoint (in the simulated three-space) clusters. In this case, it is necessary that the clumps be rearranged (while keeping the actual galaxies fixed). The condition to aim for is this: for every clump  $C$ , the closest clump to  $C$  external to  $C$  shall be its parent clump. Let  $\text{Closest}(C)$  be the nearest clump with which  $C$  is compared during the execution of procedure  $\text{TwoNode}$ . If the distance from  $C$  to  $\text{Closest}(C)$  is less than the distance from  $C$  to its parent, then a new clump  $W$  will be formed, which will become the subclump of  $\text{Parent}(C)$  in place of  $C$ .  $W$  will contain as subclumps  $C$  and  $\text{Closest}(C)$ . Now the old parent clump of  $\text{Closest}(C)$  has only one subclump, so it can be liquidated, “promoting” its subclump. This process is represented in Fig. 6.1.

These adjustments (which shall be known as *Grabs*) take place immediately after procedure  $\text{ComputeAccel}$  finishes running. Each *Grab* is a purely local phenomenon in the data structure (only affecting four nodes), and preserves the positions, velocities, accelerations, and all other important data of the clumps involved. The process of *Grabbing* guarantees that close pairs will be subclumps of the same clump, and that

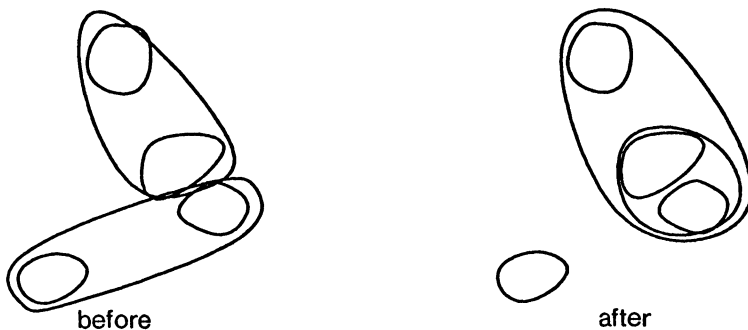
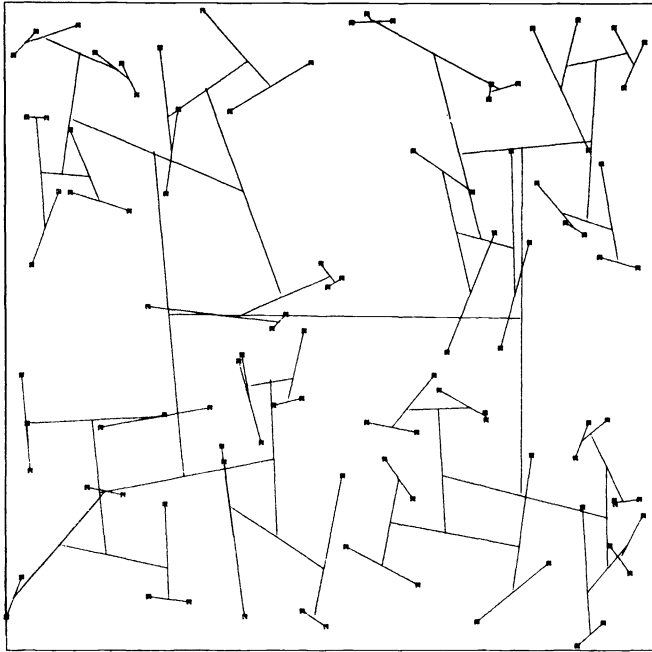
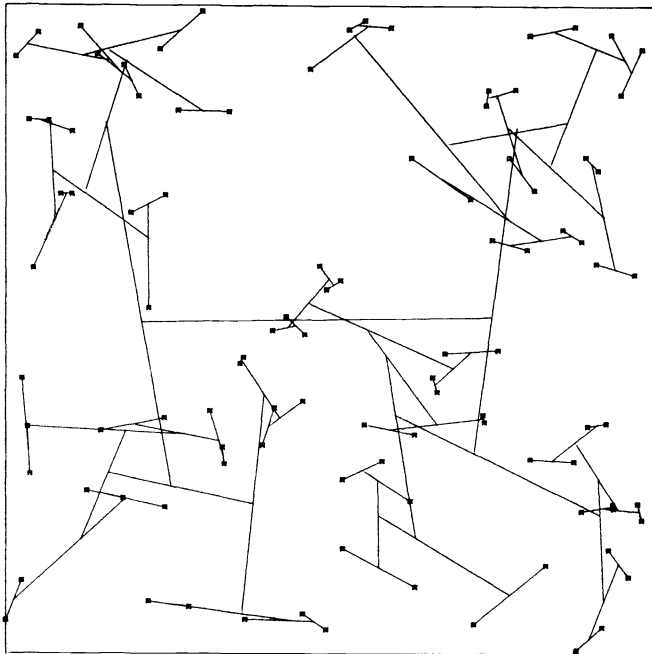


FIG. 6.1. *Rearrangement of clumps.*



(a)



(b)

FIG. 6. Effect of the Grab algorithm. Figure 6.2 illustrates the effect of the Grab algorithm on a two-dimensional universe. The diagram on the left depicts the clump structure as first created, by alternately splitting at the median  $x$  and  $y$ . The diagram on the right shows the structure after several iterations of Grab. Note that the particles are in the same positions, but the structure is cleaner—close pairs are now all linked directly together. This improved structure may be measured by the fact that the acceleration calculation on the improved structure is empirically observed to be about twice as efficient as on the original structure.

the clumps will be close to optimally arranged for quickly computing accelerations. Although the Grab algorithm does not find the “best” arrangement of clumps, it has been observed to do a fairly good job in a very short time (see Fig. 6.2).

The TwoNode procedure that calculates the accelerations throughout the tree also stores information that is used by the rearrangement algorithm in finding candidates for Grabs. The rearrangement is done after every iteration; it takes time linear in the number of particles.

**6.2. Creating the tree.** While grabbing is very useful in maintaining the clump structure in the face of distortions, it will not be able to create one in the first place from a randomly arranged set of galaxies. This will be done as follows. The universal clump—which contains all the galaxies—will be divided initially into two subclumps chosen so that the first contains all galaxies whose  $x$  coordinate is less than the median  $x$  coordinate, and the other subclump will contain all galaxies with  $x$  larger than or equal to the median  $x$ .

Each of those subclumps will be divided into two sub-clumps using the median  $y$  as the splitting criterion. Each lower level of clump will be split on  $z$ , then  $x$ , then  $y$  then  $z$ , until the clumps consist of one galaxy each. Note that this procedure does not require that the number of clumps be a power of two, although that might seem most natural.

This structure is known as a  $k$ -d tree [4]. It has a variety of applications in multidimensional problems, including searching, nearest-neighbor calculations, classification, numerical integration, and computing minimum spanning trees [10], [5], [11]. For a many-body application, a standard  $k$ -d tree will be far from optimal—nearby objects will not be in the same clump much of the time. The Grab procedure, though its behavior is difficult to analyze theoretically, has been observed to do a very good job of cleaning up the structure in just two or three iterations (see Fig. 6.2).

**7. Implementation of the program.** Various algorithmic attacks using the center-of-mass tree structure have been described in the preceding sections. It is inappropriate to stop seeking reductions in running time after a good algorithm has been found, however; significant efficiencies can be achieved in the implementation of a given algorithm.

The algorithm as described was first implemented in about 1,200 lines of Pascal on a VAX-11/780. For a problem size of 10,000 galaxies, this first implementation runs in about forty minutes per iteration, and about 500 iterations are required to simulate the expansion of the universe by a factor of 100. Under the general relativistic assumptions made, letting  $t$  run from 1 to 1,000 causes the distance scales to run from 1 to 100, because distance is proportional to  $t^{2/3}$ . Accelerations are transformed at each iteration to correspond with the changing distance scale [3]. Thus, 340 hours of execution time would be needed for this program, as opposed to 8,000 for the  $O(N^2)$  algorithm. The times given throughout this paper are for a slight modification of the algorithm to simulate a periodic distance function, which was necessary in the initial application. This adds a small constant factor to all distance calculations (eighteen floating point instructions, or about 25% of the running time).

A profiler was used to identify those parts of the program that consumed most of the processing time [20]. The profiler operates by asynchronously sampling the computer’s program counter 60 times per second and incrementing the appropriate bin of a distribution function. The results showed that all but two percent of the execution time was spent in the TwoNode procedure. This was not unexpected, as TwoNode is the only part of the algorithm with an order time of  $O(N \log N)$ ; the

rest of the procedures run in  $O(N)$  time. Since TwoNode is relatively small, hand optimization of the machine code was an obvious step. Writing this procedure in assembly language resulted in a speedup by a factor of two and a half. This rewriting used standard techniques, such as keeping more quantities in registers, putting procedure calls in-line, and using the addressing modes of the VAX more effectively.

At this point we found that the use of the Floating Point Accelerator option on the VAX significantly improves the performance of the program. The program was sped up by a factor of two by moving the calculations to a VAX on which an FPA had been installed.

Many-body calculations usually require double-precision arithmetic because of the wide range of distances involved. Close orbits are often more than four orders of magnitude—a dozen binary digits—closer than the distance to a far-away galaxy. Since the improved algorithm stores all positions relative to the parent clump, this problem disappears—typically only one order of magnitude, or less, is involved in the difference between the size of a clump and the size of its parent clump. The use of 32-bit floating point numbers in place of 64-bit floating point halved the running time of the algorithm.

The factors of two in speed from the use of the Floating Point accelerator and from the use of single precision are approximate and interdependent. Table 7.1 shows the running time as a function of these variables.

TABLE 7.1  
*Running times, in seconds, of an acceleration calculation for 1,000 bodies on a VAX-11/780.*

	32-bit floating point	64-bit floating point
With FPA hardware	16	28
Without FPA hardware	25	74

Since tight, “indivisible” clumps are recognized and their small time constant does not affect the time constant of the universe, far fewer iterations are required. Typically, such clumps are iterated about four times for each iteration of the universal clump. Each of those iterations would have been a global iteration in the straightforward algorithm, as in that algorithm there is no way to detect a tight clump. A conservative estimate of the number of iterations saved is 50%—a factor of two speedup.

In the astrophysical applications described in § 2, in which galaxy clustering occurs, the development of clusters among the particles simulated leads to greater opportunities for procedure TwoNode to apply its approximation. The resulting gain is an empirically observed two-fold speed-up in computation of accelerations.

The program that resulted from these modifications to a very simple iteration method succeeded in reducing the running time of a simulation from 4,000 hours to 20 hours—a factor of two hundred (for ten thousand bodies, with  $\delta^2 = 0.3$ ). This saving was achieved by attacking the problem from several angles at once.

**8. Conclusions.** It is often difficult to make one change in a program that makes it faster by more than an order of magnitude. In this case, even a change that reduced the order time of the algorithm from  $O(N^2)$  to  $O(N \log N)$  increased the efficiency for a typical problem size by only a factor of 12—one order of magnitude. The four-hundred-fold reduction in running time was the product of savings at all levels of the conceptual hierarchy, from the idea that some galaxies are in systems by themselves, to the idea that keeping certain points in registers saves memory references

(see Table 8.1). They are in some sense independent—improving the efficiency of one level of the hierarchy does not preclude improving the efficiency of another. Most importantly, all of the savings are multiplied together.

Reddy and Newell [18] have characterized the type of problem for which this multiplicative speedup can be expected: such a problem has four to eight layers of implementation, such as computer technology, architecture, algorithm, et cetera. Other programs for the  $N$ -body problem have achieved substantial speedups over the most simplistic implementation, by attacking two or three layers of this hierarchy. This paper has been concerned with ways to avoid changes in the technology and architecture layers (i.e., using Cray-1) because of their expense. Rather, the algorithms, “knowledge sources,” and implementation layers have been attacked.

TABLE 8.1  
*Summary of the speedups attained at various levels.*

Level	Speedup factor	Description
Algorithm	12	Changing to the $O(N \log N)$ algorithm
Problem-Specific Knowledge	2	Iterating indivisible clumps by themselves and using closed-form solutions, thus halving the number of global iterations
Algorithm (Problem-Specific)	2	Clustering behavior in the simulation produces a clump structure well-suited to the algorithm
System-Independent Code Tuning	2	Use of single precision floating point rather than double precision, made possible by the data structure
System-Dependent Code Tuning	2.5	Hand-coding the routine where most of the time was spent
Hardware	2	Use of the Floating-Point Accelerator

This brings the running time of the algorithm on a relatively small and inexpensive computer such as the VAX down to what it would be on a large, extremely fast, and expensive Cray-1. Of this speedup, a factor of about two was attributable to technology (the use of the Floating Point Accelerator) and two to implementation (hand coding a critical routine)—these could be done for any program, probably with similar results. The other factor of a hundred (for ten thousand bodies) came from the exploitation of the data structure in various ways. The use of a good data structure to provide an asymptotically fast algorithm is especially important for large problems.

Since the layers of the problem are relatively independent, the technology and architecture layers are still available for additional speedup factors. If the program were run on a Cray-1 or a Cyber 205, the 20 hours of runtime might be reduced to 1 or 2 hours, since most of the efficiency improvements described in this paper are machine-independent, and these computers are much faster than the VAX (and almost proportionally more expensive).

The data structure is a variant of one already known in the literature (the  $k$ -d tree), but the reorganization of the tree with the Grab procedure changes it substantially—it loses the useful (for some applications) property of being split along planes of constant  $x$ ,  $y$ , and  $z$ , and gains the useful (for this application) property of joining mutually nearest neighbors at all levels of the hierarchy. For the simulation of gravitational attractions, this turned out to better than halve the number of calculations.

Reorganized trees may have other applications as well; for example, the recognition of individual objects from the point set obtained from a television camera might be facilitated by an algorithm that could group points together in  $O(N \log N)$  time. Some sorts of nearest-neighbor searching might also be made easier.

It is difficult to analyze the properties of the Grab algorithm. It is low-level in nature: when two points are found to be closer to each other than to their parent nodes, a local rearrangement is done without regard for the global structure of the tree. That it works as well as it does was difficult to predict. Its behavior is dependent on  $\delta$ , since these closest pairs are detected during the TwoNode procedure; the question of what  $\delta$  to use to most efficiently produce a reorganized tree (independent of gravitational considerations) might be investigated if reorganized trees are found to be useful in other applications.

## REFERENCES

- [1] SVERRE J. AARSETH, J. RICHARD GOTT III AND EDWIN L. TURNER, *N-body simulations of galaxy clustering; I. Initial conditions and galaxy collapse times*, *Astrophys. J.*, 228 (1979), pp. 664–683.
- [2] ALFRED V. AHO, JOHN E. HOPCROFT AND JEFFREY D. ULLMAN, *The Design and Analysis of Computer Algorithms*, Addison-Wesley, Reading, MA, 1974.
- [3] ANDREW W. APPEL, *An investigation of galaxy clustering using an asymptotically fast N-body algorithm*, Undergraduate Thesis, Princeton Univ., Princeton, NJ, April 1981.
- [4] JON LOUIS BENTLEY, *Multidimensional binary search trees used for associative searching*, *Comm. ACM*, 18 (1975), pp. 509–517.
- [5] JON LOUIS BENTLEY AND JEROME H. FRIEDMAN, *Fast algorithms for constructing minimum spanning trees in coordinate spaces*, *IEEE Trans. Comput.*, C-27 (1978), pp. 97ff.
- [6] ROLAND BULIRSCH AND JOSEPH STOER, *Numerical treatment of ordinary differential equations by extrapolation methods*, *Numer. Math.*, 8 (1966), pp. 1–13.
- [7] EDWARD A. DESLOGE, *Classical Mechanics*, John Wiley, New York, 1982.
- [8] R. H. DICKE AND P. J. E. PEEBLES, *The big bang cosmology—enigmas and nostrums*, in *General Relativity: An Einstein Centenary Survey*, Cambridge Univ. Press, Cambridge, 1979, pp. 504–517.
- [9] A. G. DOROSHKEVICH, E. V. KOTOK, I. D. NOVIKOV, A. N. POLYUDOV, YU. G. SIGOV AND S. F. SHANDARIN, *Dvumernaya model obrazovaniya krupnomasshtabnoi struktury vselenoi (A two-dimensional model of the formation of large-scale structures of the universe)*, Preprint 83, IPM AN SSSR (Institute for Problems of Mechanics, Academy of Science, USSR), Moscow, USSR, 1978.
- [10] JEROME H. FRIEDMAN, JON LOUIS BENTLEY AND RAPHAEL ARI FINKEL, *An algorithm for finding best matches in logarithmic expected time*, *ACM Trans. Math. Software*, 3 (1977), pp. 209ff.
- [11] JEROME H. FRIEDMAN AND MARGARET H. WRIGHT, *A nested partitioning procedure for numerical multiple integration*, *ACM Trans. Math. Software*, 7 (1981), pp. 76ff.
- [12] EDWARD J. GROTH, P. JAMES E. PEEBLES, MICHAEL SELDNER AND RAYMOND M. SONEIRA, *The clustering of galaxies*, *Scientific American*, 237 (1977), pp. 76ff.
- [13] ROGER W. HOCKNEY AND JAMES W. EASTWOOD, *Computer Simulation Using Particles*, McGraw-Hill, New York, 1981.
- [14] M. JOEVEER, J. EINASTO AND E. TAGO, *Yacheiskaya struktura vselenoi (The cell structure of the universe)*, Preprint A-1, AN Estonskoi SSR, Tartu, Estonian SSR, USSR, 1977.
- [15] R. H. MILLER AND K. H. PRENDERGAST, *Stellar dynamics in a discrete phase space*, *Astrophysical J.*, 151 (1968), pp. 699ff.
- [16] R. H. MILLER, K. H. PRENDERGAST AND WILLIAM J. QUIRK, *Numerical experiments on spiral structure*, *Astrophysical J.*, 161 (1970), pp. 903–916.
- [17] P. J. E. PEEBLES, *The Large-Scale Structure of the Universe*, Princeton Univ. Press, Princeton, NJ, 1980.
- [18] R. REDDY AND ALLEN NEWELL, *Multiplicative Speedup of Systems*, in *Perspectives on Computer Science*, A. K. Jones, ed., Academic Press, New York, 1977, pp. 183–198.
- [19] RICHARD M. RUSSELL, *The CRAY-1 computer system*, in *Computer Structures: Principles and Examples*, Daniel P. Siewiorek, C. Gordon Bell, and Allen Newell, eds., McGraw-Hill, New York, 1982, pp. 743–752.

- [20] *Unix Programmer's Manual*, Computer Science Division, Department of Electrical Engineering and Computer Science, University of California, Berkeley, CA. Sections **prof(1)**, **pc(1)**, and **profil(2)** contain information about the execution profiler.
- [21] M. MITCHELL WALDROP, *The large-scale structure of the universe*, *Science*, 219 (1983), pp. 1050–1052.
- [22] YA. B. ZELDOVICH, *The theory of the large scale structure of the universe*, IAU symposium #79, International Astronomical Union, Dordrecht, 1978, pp. 409ff.

## A DIRECT EULERIAN MUSCL SCHEME FOR GAS DYNAMICS\*

PHILLIP COLELLA†

**Abstract.** We present a second order extension of Godunov's method for gas dynamics in Eulerian coordinates patterned after van Leer's MUSCL scheme for gas dynamics in Lagrangian coordinates. The present method performs the Eulerian calculation in a single step by solving Riemann problems and characteristic equations for the fluxes in the Eulerian frame. We also make several modifications in the formulation of MUSCL, applicable to both this scheme and to the original Lagrangian scheme, all aimed at making a more robust and accurate scheme. We present the results of test calculations in one and two space variables.

**Key words.** hyperbolic conservation laws, Godunov's method, Riemann problem

**1. Introduction.** In [7], van Leer described MUSCL, a second order accurate extension of Godunov's method [4], [5] for solving the equations of gas dynamics in one space variable in Lagrangian coordinates. van Leer presented this Lagrangian scheme as the core of a multidimensional Eulerian code, developed by van Leer and Woodward [8]. One time step of a one-dimensional Eulerian calculation is done by performing a one-dimensional Lagrangian step, then mapping the results back to a fixed Eulerian grid. The multidimensional algorithm is obtained by using the one-dimensional Eulerian algorithm with operator splitting.

In this paper, we present a different MUSCL algorithm, based on some of the ideas in [7], for computing gas dynamics in Eulerian coordinates in one space dimension. The present algorithm is not formulated as a Lagrangian step, followed by a remap, but performs the Eulerian calculation in a single step. This direct Eulerian MUSCL bears the same relation to the nonlinear Eulerian Godunov algorithm discussed in [1], [5], as the Lagrangian MUSCL does to Godunov's method in Lagrangian coordinates. As in [7], the extension to multidimensional calculations is then performed using operator splitting.

Because we work in Eulerian coordinates, the details of the direct Eulerian algorithm are substantially different than those of the Lagrangian scheme. In MUSCL, dissipation at shocks is introduced by the constant reaveraging of a discrete travelling wave solution on the mesh. Since shocks always move relative to the mesh in Lagrangian coordinates, there is always introduced a certain minimum amount of dissipation in the solution near shocks in Lagrangian calculations. In Eulerian coordinates, it is possible to have nearly stationary shocks where the dissipation vanishes; consequently, it is necessary to introduce dissipative mechanisms for strong nonlinear waves beyond those described in [7]. More generally, care is required at places where one of the characteristic speeds associated with sound waves vanishes. This, plus the additional logic involved with both solving the Riemann problem and tracing characteristics in the Eulerian frame, make for a slightly more complicated algorithm than the simplest form of the 1D Lagrange plus remap MUSCL discussed in [7]. On the other hand, there is no remap to perform. Furthermore, we introduce some innovations whose analogues are not present in the Lagrangian method in [7]. In particular, we use the

---

\* Received by the editors March 16, 1982, and in revised form June 15, 1983. This work was supported in part by the Office of Energy Research, Office of Basic Energy Sciences, Engineering, Mathematical and Geosciences Division of the U.S. Department of Energy under contract DE-AC03-76SF00098.

† Lawrence Berkeley Laboratory, University of California, Berkeley, California 94720.



simplified Riemann problem solver discussed in [1]. Also, we derive the slopes of the distributions of the dependent variables from the average values, rather than treating them as separate dependent variables, as was done in the code which generated the results presented in [7]; thus, the present one-dimensional algorithm is compatible with any multidimensional Eulerian code which performs its hydrodynamics calculation in a series of one-dimensional passes. The interpolation algorithm for deriving the slopes is slightly more complicated than the second order central difference algorithm discussed in [7], but yields a steeper representation of discontinuities, particularly contact discontinuities. Finally, we take advantage of the fact that for gas dynamics, the characteristic equations for the hydrodynamic waves are well approximated by the shock jump relations for the waves of the opposite family. We exploit this relation in such a way that the solution of the characteristic equations reproduces the correct shock jump relations in the presence of strong gradients.

**2. Description of the method.** We will be constructing approximate solutions to Euler's equations describing the motion of an inviscid compressible fluid in one space variable  $r$ :

$$(2.1) \quad \frac{\partial U}{\partial t} + \frac{\partial(AF)}{\partial V} + \frac{\partial H}{\partial r} = 0,$$

$$U = \begin{pmatrix} \rho \\ \rho u \\ \rho v \\ \rho E \end{pmatrix}, \quad F(U) = \begin{pmatrix} \rho u \\ \rho u^2 \\ \rho uv \\ \rho uE + up \end{pmatrix}, \quad H(U) = \begin{pmatrix} 0 \\ p \\ 0 \\ 0 \end{pmatrix}.$$

Here  $V = V(r)$  is a generalized volume coordinate,  $A(r) = dV/dr$ . These equations describe one-dimensional inviscid compressible flow with either planar, cylindrical, or spherical symmetry, or flow in a duct whose cross-section at  $r$  is  $A(r)$ , depending on whether  $V(r) = r$ ,  $r^2/2$ ,  $r^3/3$ , or  $\int_{r_0}^r A(r) dr$ , respectively. Here  $\rho$  is the density,  $u$  is the component of velocity in the direction of the one-dimensional sweep,  $v$  is the component of velocity orthogonal to  $u$  (hereafter,  $u$  and  $v$  will be referred to as the velocity and transverse velocity, respectively), and  $E$  is the total energy per unit mass. We define  $e$ , the internal energy per unit mass, and  $p$ , the pressure, as

$$e = E - \frac{1}{2}(u^2 + v^2), \quad p = (\gamma - 1)\rho e$$

where  $\gamma$  is the ratio of specific heats. Throughout this paper,  $\gamma$  will be assumed to be a constant,  $\gamma > 1$  (polytropic gas); for a discussion of the modifications required for a more general equation of state, see Colella and Glaz [2].

There are several other derived quantities which will be of interest:  $\tau$ , the specific volume,  $c$ , the speed of sound, and  $\lambda_{\pm,0}$ , the three characteristic velocities:

$$\tau = \tau(U) = \frac{1}{\rho}, \quad c = c(U) = \sqrt{\frac{\gamma p}{\rho}}, \quad \lambda_{\pm} = \lambda_{\pm}(U) = u \pm c, \quad \lambda_0(U) = u.$$

Let  $\Delta t$  be a time increment,  $r_{j+1/2}$  the boundary between zones  $j$  and  $j+1$ , and define  $r_j = \frac{1}{2}(r_{j+1/2} + r_{j-1/2})$ ,  $\Delta r_j = r_{j+1/2} - r_{j-1/2}$ , and  $\Delta V_j = V(r_{j+1/2}) - V(r_{j-1/2})$ . We assume that, at time  $t^n$ , we know  $U_j^n$ , the averages of the conserved quantities across each zone:

$$U_j^n = \frac{1}{\Delta V_j} \int_{r_{j-1/2}}^{r_{j+1/2}} U(r, t^n) dV.$$

We wish to compute  $U_j^{n+1}$ , the averages of the conserved quantities at the new time  $t^{n+1} = t^n + \Delta t$ :

$$U_j^{n+1} = \frac{1}{\Delta V_j} \int_{r_{j-1/2}}^{r_{j+1/2}} U(r, t^{n+1}) dV.$$

In outline, the procedure followed by MUSCL for calculating  $U_j^{n+1}$  can be divided into five steps:

1) Compute linear profiles of the dependent variables in each zone by interpolating slopes at the centers of zones, subject to certain monotonicity constraints. This gives rise to a global distribution of the dependent variables which is piecewise linear, linear in each zone, with jump discontinuities at the edges of zones.

2) Compute  $U_{j+1/2}^n$ , the solution at the old time at the edges of zones, by solving the Riemann problems which resolve the jump discontinuities at the edges of the zones.

3) Compute  $U_{j+1/2}^{n+1}$ , an approximation to the solution at the edge of the zones at the new time, by tracing approximate characteristics, and solving difference approximations to the characteristic equations.

4) Compute time-averaged values of  $F$  and  $H$ , using the values computed in 2) and 3), and the following formula:

$$F_{j+1/2} = \frac{\Delta t}{2} (F(U_{j+1/2}^n) + F(U_{j+1/2}^{n+1})) = \int_{t^n}^{t^{n+1}} F(U(r_{j+1/2}, t)) dt + O(\Delta t^3 + \Delta r_j^2 \Delta t),$$

$$H_{j+1/2} = \frac{\Delta t}{2} (H(U_{j+1/2}^n) + H(U_{j+1/2}^{n+1})) = \int_{t^n}^{t^{n+1}} H(U(r_{j+1/2}, t)) dt + O(\Delta t^3 + \Delta t \Delta r_j^2).$$

5) Calculate the conserved quantities using divided differences of the values calculated in 4):

$$(2.2) \quad U_j^{n+1} = U_j^n - \frac{A_{j+1/2} F_{j+1/2} - A_{j-1/2} F_{j-1/2}}{\Delta V_j} - \frac{(H_{j+1/2} - H_{j-1/2})}{\Delta r_j}.$$

Clearly, most of the work in this algorithm is in steps 1)–3). We will proceed to describe those steps in more detail.

*Step 1. Interpolation of slopes.* Given our discrete data  $U_j^n$ , we will interpolate a global description for our dependent variables at all points  $(r, t^n)$  which is piecewise linear, and linear in each zone:

$$(2.3) \quad q(r) = q_j^n + \delta q_j \frac{(r - r_j)}{\Delta r_j}, \quad r_{j-1/2} < r < r_{j+1/2}.$$

Here  $q = q(U)$  represents any useful flow variable, conserved or not. For  $q = p, \rho, u, v$ , we will take  $q_j^n = q(U_j^n)$ , and construct the slopes  $\delta q_j$  by a suitable difference formula. The distributions of other quantities  $h = h(p, \rho, u, v)$  are then derived from those of  $p, \rho, u, v$  in the following fashion:

$$(2.4) \quad \begin{aligned} q_{j+1/2,L} &= q_j^n + \frac{1}{2} \delta q_j, & q_{j+1/2,R} &= q_{j+1}^n - \frac{1}{2} \delta q_{j+1}, & q &= p, \rho, u, v, \\ h_{j+1/2,S} &= h(p_{j+1/2,S}, \rho_{j+1/2,S}, u_{j+1/2,S}, v_{j+1/2,S}), & S &= L, R, \\ \delta h_j &= (h_{j+1/2,L} - h_{j-1/2,R}), \\ h_j^n &= \frac{1}{2} (h_{j+1/2,L} + h_{j-1/2,R}), \\ h(r) &= h_j^n + \delta h_j \frac{(r - r_j)}{\Delta r_j}, & r_{j-1/2} &< r < r_{j+1/2}. \end{aligned}$$

In the case of equally spaced zones  $\Delta r_j = \Delta r$ ,  $\delta q_j$  is calculated using the following two-step algorithm. We first calculate  $\delta_f q_j$ , a first guess to the slope using the monotonized central difference algorithm discussed in [7].

$$(2.5) \quad \delta_{\text{lim}} q_j = \begin{cases} \min(|q_{j+1}^n - q_j^n|, |q_j^n - q_{j-1}^n|)2 & \text{if } (q_{j+1}^n - q_j^n)(q_j^n - q_{j-1}^n) > 0, \\ 0 & \text{otherwise,} \end{cases}$$

$$\delta_f q_j = \min \left( \frac{|q_{j+1} - q_{j-1}|}{2}, \delta_{\text{lim}} q_j \right) \text{sgn} (q_{j+1} - q_{j-1}).$$

Finally, we calculate  $\delta q_j$  by differencing the values at two points  $\delta q$  on either side of  $r$  obtained by using the interpolant given using  $\delta_f q$  as the slope:

$$(2.6) \quad \delta q_j = \min \left\{ \frac{2}{3} |q_{j+1} - \frac{1}{4} \delta_f q_{j+1} - q_{j-1} - \frac{1}{4} \delta_f q_{j-1}|, \delta_{\text{lim}} q_j \right\} \text{sgn} (q_{j+1} - q_{j-1}),$$

$$\delta q_j = \delta(q_{j-2}, \dots, q_{j+2}).$$

To obtain  $\delta q_j$  in the case of unequal zones, calculate  $\bar{\delta} q_j = \delta(q_{j-2}, \dots, q_{j+2})$ ,  $\bar{\delta} r_j = \delta(r_{j-2}, \dots, r_{j+2})$ , using (2.6). Then we calculate

$$(2.7) \quad \delta q_j = \Delta r_j \min \left\{ \frac{|\bar{\delta} q_j|}{|\bar{\delta} r_j|}, \frac{2|q_{j+1} - q_j|}{\Delta r_j}, \frac{2|q_j - q_{j-1}|}{\Delta r_j} \right\} \text{sgn} (q_{j+1} - q_{j-1}).$$

In the case where the minima in (2.5)–(2.7) are obtained in the first arguments, one obtains

$$\frac{\delta q_j}{\Delta r_j} = \frac{(\frac{2}{3}(q_{j+1} - q_{j-1}) - \frac{1}{12}(q_{j+2} - q_{j-2}))}{(\frac{2}{3}(r_{j+1} - r_{j-1}) - \frac{1}{12}(r_{j+2} - r_{j-2}))},$$

which is a fourth order finite difference approximation to  $dq/dr|_{r_j}$ , and thus is well behaved in regions where the solution is smooth. The fact that  $\delta q_j$  is obtained from  $\delta_f q$ , a monotonized first guess, gives rise to steeper profiles representing discontinuities than those obtained using either the fourth order accurate formula by itself, or by setting  $\delta q_j = \delta_f q_j$ , as was suggested in [7].

There are situations, however, in which the above slope setting procedure leads to profiles which are too steep, in the sense that the scheme will not provide sufficient dissipation to ensure that the correct amount of entropy production occurs. This situation arises when the speed of the characteristic of the family associated with the shock changes sign across the shock, i.e., where the shock is nearly stagnant. In such situations, the calculation remains stable, but there is a small amplitude ( $\approx 5\%$ ), low frequency error in the post shock values generated at the shock. In this case, we reduce the slopes computed by the above procedure by some fraction  $\chi_j$ ,  $0 \leq \chi_j \leq 1$ :  $\delta q_j^{\text{reduced}} = \delta q_j \chi_j$ . We want  $\chi_j$  to have the following properties. If the  $j$ th zone is not inside a shock, or if the  $j$ th zone is inside the shock, but the speed of the characteristics of the family associated with that shock does not change sign, then  $\chi_j = 1$ . If the  $j$ th zone is inside a shock having zero velocity, then  $\chi_j = 0$ , thus reducing the method locally to Godunov's method. Intermediate cases should have an intermediate amount of flattening. Finally,  $\chi_j = 1$  if there is not the possibility of a significant amount of entropy production across the zone. The formula given below for  $\chi_j$  satisfies the above requirements.

$$W_j^2 = |p_{j+1} - p_{j-1}| \left/ \left| \frac{1}{p_{j+1}} - \frac{1}{p_{j-1}} \right| \right.,$$

$$s_j = \text{sgn} (p_{j-1} - p_{j+1}),$$

$$\lambda_{j,R} = u_{j+1} + s_j c_{j+1}, \quad \lambda_{j,L} = u_{j-1} + s_j c_{j-1},$$

$$U_j = \frac{W_j}{\rho_{j+\xi_j}} + s_j u_{j+s_p}$$

$$\bar{\chi}_j = \begin{cases} \frac{|U_j|}{|U_j| + \min(|\lambda_{j,L}|, |\lambda_{j,R}|)} & \text{if } \varepsilon_p - \frac{|p_{j+1} - p_{j-1}|}{\min(p_{j+1}, p_{j-1})}, \lambda_{j,R} \lambda_{j,L}, \text{ and } (u_{j+1} - u_{j-1}) < 0, \\ 1 & \text{otherwise,} \end{cases}$$

$$\chi_j = \max(0, 1 - (1 - \bar{\chi}_j)/\eta).$$

Here  $0 < \eta \leq 1$  and  $\varepsilon_p$  is the minimum pressure jump which would be considered a shock; in the calculations presented here  $\eta = \frac{1}{2}$ ,  $\varepsilon_p = \frac{1}{4}$ .

*Steps 2-3. Calculation of interface values.* We must calculate  $U_{j+1/2}^n, U_{j+1/2}^{n+1}$ , approximate values to the solution at the old and new times, at the zone edges  $r_{j+1/2}$ . To obtain  $U_{j+1/2}^n$ , we calculate the solution to the Riemann problem. Since the solution to be obtained from the Riemann problem is for an infinitesimal time after the breakdown of the initial jump, the geometric source terms have no effect on the solution, so that the Riemann problem we solve is for the equations of gas dynamics in Cartesian coordinates. As is well known (for a detailed discussion, see Collella [1], and the references cited there), the solution to that Riemann problem with left and right states  $U_L, U_R$  is  $\psi((r/t), U_L, U_R)$ ; i.e., it depends on  $r, t$  only in the ratio  $r/t$ .

To calculate  $U_{j+1/2}^n$ , we take our states

$$U_L, U_R = U_{j+1/2,L}^n, U_{j+1/2,R}^n$$

(see Fig. 1), and set

$$U_{j+1/2}^n = \psi(0; U_L, U_R).$$

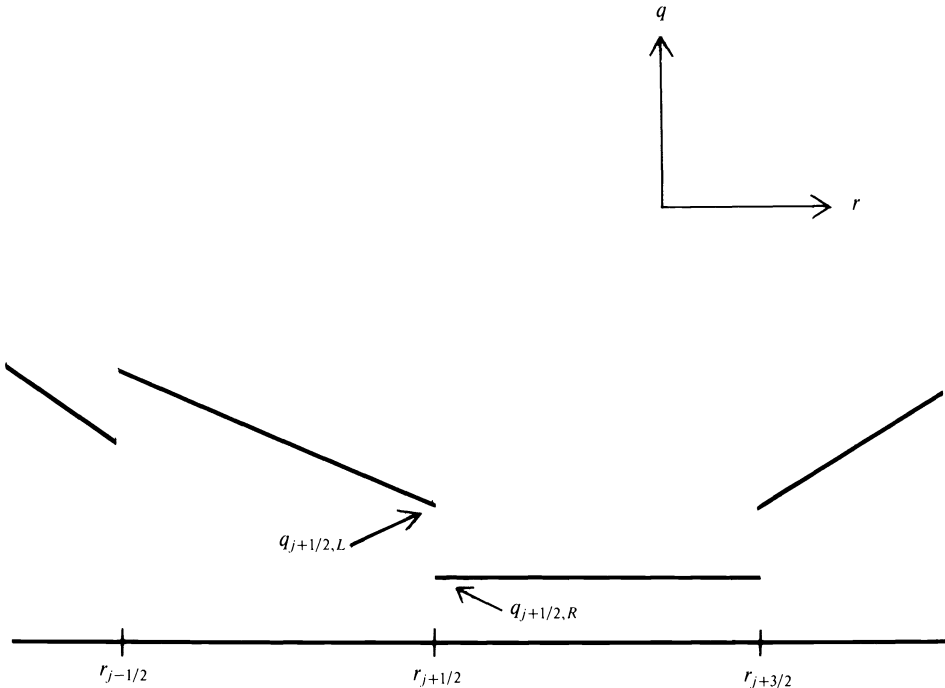


FIG. 1. Spatial distribution of  $q$  at initial time  $t^n$ .

If  $U^n(r, t)$  is the exact solution to the initial value problem given by the global piecewise linear distribution (2.3), then  $\lim_{t \downarrow t^n} U^n(r_{j+1/2}, t) = U_{j+1/2}^n$ . As was the case for the Eulerian Godunov's method, the approximate Riemann problem solver described in [1] appears to be both inexpensive and sufficiently accurate, without introducing rarefaction shocks into the solution.

To calculate  $U_j^{n+1}$ , we solve a finite difference approximation to the characteristic equations, which we review briefly below. Given a solution  $U(x, t)$  to (2.1), we say that a curve  $\sigma_{\pm,0} \rightarrow (r(\sigma_{\pm,0}), t(\sigma_{\pm,0}))$  is a characteristic of the  $+, -, 0$  family if  $U$  is continuous in a neighborhood of that curve, and if the following ordinary differential equations hold

$$(2.8)_{\pm,0} \quad \frac{dr}{d\sigma_{\pm,0}} = \lambda_{\pm,0}, \quad \frac{dt}{d\sigma} = 1,$$

$$(2.9)_{\pm} \quad \frac{1}{\rho c} \frac{dp}{d\sigma_{\pm}} \pm \frac{du}{d\sigma_{\pm}} + \frac{A'}{A} uc = 0,$$

$$(2.9)_0 \quad \frac{d\tau}{d\sigma_0} = -\frac{1}{(\rho c)^2} \frac{dp}{d\sigma_0}.$$

Here  $dh/d\sigma = (d/d\sigma)(h(r(\sigma), t(\sigma)))$  and all functions of  $(r, t)$  are evaluated at  $(r(\sigma), t(\sigma))$ .

The equations (2.8), (2.9) completely describe the solution in regions where  $U$  is continuous or near contact discontinuities. However, in the neighborhood of a shock, the equations (2.9) no longer hold along the curves described by (2.8), and some modification to the equations must be introduced which takes into account this fact.

Our strategy for calculating  $U_{j+1/2}^{n+1}$  proceed as follows, considering, for the moment, the case  $dA/dr \equiv 0$ . First, we find approximations to the paths described by  $(2.8)_{\pm,0}$  which intersect the point  $(r_{j+1/2}, t^{n+1})$ , taking due care to trace backwards to the origins of centered rarefaction fans if  $(r_{j+1/2}, t) > t^n$  is inside such a fan. Then we calculate  $U_{j+1/2}^{n+1}$  in three stages. First, we solve a pair of nonlinear algebraic equations for  $p_{j+1/2}^{n+1}, u_{j+1/2}^{n+1}$ , given the values of the solution at the base of the  $+, -$  characteristics. These are the same nonlinear algebraic equations as those for the values of  $p, u$  between the two sonic waves in the Riemann problem with left and right states, given by, respectively, the values of the solution at the base of the  $+$  and  $-$  characteristics. Intuitively, what we are doing is lumping all the waves of the  $+$  (resp.  $-$ ) family which are crossed by the  $-$  (resp.  $+$ ) characteristic into a single shock or rarefaction shock jump. In the case where the solution is continuously differentiable, we obtain a nonlinear finite difference approximation to  $(2.9)_{\pm}$ . If the solution is not smooth, this procedure gives values for  $p_{j+1/2}^{n+1}, u_{j+1/2}^{n+1}$  which are well behaved. We then solve an explicit equation for the  $\rho_{j+1/2}^{n+1}$ , given the value of the solution at time  $t^n$  and that of  $p_{j+1/2}^{n+1}$ , which again lumps the pressure wave crossing the streamline into a single shock or rarefaction shock. In the limit that the pressure jump is small, we similarly obtain a solution to a finite difference approximation to  $(2.9)_0$ . Finally,  $v_{j+1/2}^{n+1}$  is just set equal to its value at the base of the approximate characteristic of the 0-family.

We now give the details of the procedure outlined above. First, we want to determine points  $(r_{j+1/2,\#}, t^n)$  such that  $(r_{j+1/2,\#}, t^n)$  and  $(r_{j+1/2}, t^{n+1})$  are connected by a straight line which approximates a solution to  $(2.8)_{\#}$ . To this end, we define  $\Delta r_{j+1/2,\#}^{rr}, s^{rr}, r_{j+1/2,\#}^{rr}, q_{j+1/2,\#}^{rr}, \delta q_{j+1/2,\#}^{rr}, q = p, \rho, u, v, \lambda_{\#}$ , as follows:

$$(q_{j+1/2,\#}^{rr}, \delta q_{j+1/2,\#}^{rr}, \Delta r_{j+1/2,\#}^{rr}, s_{j+1/2,\#}^{rr}) = \begin{cases} (q_{j-1}, \delta q_{j-1}, \Delta r_{j-1}, r_{j-1}, 1) & \text{if } \lambda_{\#}(U_{j+1/2}^n) \geq 0, \\ (q_j, \delta q_j, \Delta r_j, r_j, -1) & \text{if } \lambda_{\#}(U_{j+1/2}^n) < 0. \end{cases}$$

These are the quantities which describe the linear distribution of the dependent variables in the zone which contains  $(r_{j+1/2,\#}, t^n)$ . Given these quantities, we define  $r_{j+1/2,\#}$  to be

$$(2.10) \quad d = \frac{-\frac{1}{2}s_{j+1/2,\#}^{tr} + \lambda_{j+1/2,\#}^{tr} \Delta t / \Delta r_{j+1/2,\#}^{tr}}{1 + \delta \lambda_{j+1/2,\#}^{tr} \Delta t / \Delta r_{j+1/2,\#}^{tr}},$$

$$r_{j+1/2,\#} = r_{j+1/2} - s_{j+1/2,\#}^{tr} \max(\min(d, \frac{1}{2}), -\frac{1}{2}) \Delta r_{j+1/2,\#}^{tr}.$$

In the case where the maxima and minima are obtained in their first arguments, this is a formula for the point where a straight line with slope  $\lambda_{\#}(r_{j+1/2,\#})$  passing through the point  $(r_{j+1/2}, t^{n+1})$  intersects the line  $\{t = t^n\}$  (Fig. 2). If we were integrating a single conservation law, this line would coincide exactly with the characteristic through  $(r_{j+1/2}, t^{n+1})$ , given that the characteristic velocity had the linear distribution given by (2.4). To the extent that we use (2.10) for a system, we are neglecting the effect of the interaction between waves of different families on the wave speeds in tracing the characteristics. This introduces an  $O(\Delta r_{j+1/2,\#}^{tr} \Delta t)$  error into the value of  $r_{j+1/2,\#}$ .

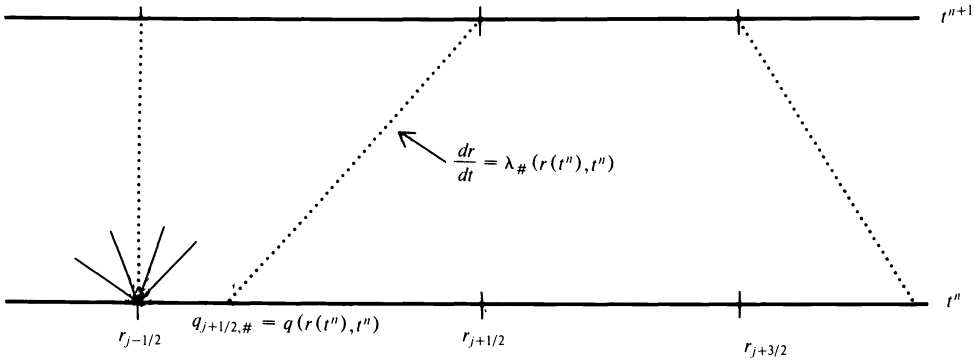


FIG. 2. Approximate solution to the characteristic equation of the # family.

Given  $r_{j+1/2,\#}$ , we can also define  $q_{j+1/2,\#}$ , the value of the solution at the base of the characteristic passing through  $(r_{j+1/2}, t^n)$ :

$$(2.11) \quad q_{j+1/2,\#} = q(r_{j+1/2,\#}) = q_{j+1/2,\#}^{tr} + \frac{r_{j+1/2,\#} - r_{j+1/2,\#}^{tr}}{\Delta r_{j+1/2,\#}^{tr}} \delta q_{j+1/2,\#}^{tr}, \quad q = p, \rho, u, v.$$

In the case that  $U_{j+1/2}^n$  came from evaluating the solution of the Riemann problem inside a centered rarefaction fan of the + or - family, we assume that the characteristic of that family passing through the point  $(r_{j+1/2}, t^{n+1})$  originates from the Riemann problem at  $(r_{j+1/2}, t^n)$  and define  $r_{j+1/2,\#}, q_{j+1/2,\#}$  accordingly:

$$r_{j+1/2,\#} = r_{j+1/2}, \quad q_{j+1/2,\#} = q_{j+1/2}^n, \quad q = p, \rho, u, v.$$

Given  $q_{j+1/2,\#}, q = p, \rho, u, v, \# = 0, +, -$ , we can now express  $p_{j+1/2}^{n+1}, p_{j+1/2}^{n+1}, u_{j+1/2}^{n+1}, v_{j+1/2}^{n+1}$  in terms of those quantities. First, we require that  $p_{j+1/2}^{n+1}, u_{j+1/2}^{n+1}$  satisfy the pair of equations

$$(2.12) \quad \frac{(p_{j+1/2}^{n+1} - p_{j+1/2,\pm})}{W(p_{j+1/2}^{n+1}, p_{j+1/2,\pm}, p_{j+1/2,\pm})} \pm (u_{j+1/2}^{n+1} - u_{j+1/2,\pm}) = 0,$$

$$W(p^*, p, \rho) = \left( \gamma p \rho \left( 1 + \frac{\gamma + 1}{2\gamma} \frac{p^* - p}{p} \right) \right)^{1/2}.$$

If  $|p_{j+1/2,+} - p_{j+1/2,-}|$ ,  $|u_{j+1/2,+} - u_{j+1/2,-}|$  are  $O(\Delta r_j, \Delta r_{j+1})$ , then this is just a finite difference approximation to (2.9)<sub>±</sub>. If either the quantities  $|p_{j+1/2,+} - p_{j+1/2,-}|$ ,  $|u_{j+1/2,-} - u_{j+1/2,+}|$  is  $O(1)$ , then we have the interpretation of the equations (2.12) given above. The equations (2.12) for  $p^{n+1}$ ,  $u^{n+1}$  are exactly the ones given in [1] for the central pressure and velocity for the approximate Riemann problem solver given in [1], and the iteration scheme given there can be used to solve (2.12) for  $p^{n+1}$ ,  $u^{n+1}$ .

The value for  $\rho_{j+1/2}^{n+1}$ ,  $v_{j+1/2}^{n+1}$  are given by the following explicit expressions:

$$(2.13) \quad \begin{aligned} \rho_{j+1/2}^{n+1} &= \left( \frac{1}{\rho_{j+1/2,0}} - \frac{(p_{j+1/2}^{n+1} - p_{j+1/2,0})}{W(p_{j+1/2}^{n+1}, p_{j+1/2,0}, \rho_{j+1/2,0})^2} \right)^{-1}, \\ v_{j+1/2}^{n+1} &= v_{j+1/2,0}. \end{aligned}$$

Again, if  $|p_{j+1/2}^{n+1} - p_{j+1/2,0}|$  is small, then (2.13) is a finite difference approximation to (2.9)<sub>0</sub>. If the pressure jump is large, then the change in density is given by lumping all of the pressure variation along the streamline into a single shock or rarefaction shock jump. An immediate consequence of the above formulas is that, if all of the slopes on either side  $r_{j+1/2}$  are zero, then  $U_{j+1/2}^{n+1} = U_j^n$ , and we recover Godunov's method, with the Riemann problem solution algorithm in [1], for calculating the fluxes.

In the case where  $A' \neq 0$ , we want to include the effect of the source terms in the calculation of  $p^{n+1}$ ,  $\rho^{n+1}$ ,  $u^{n+1}$ ,  $v^{n+1}$ . Let  $\bar{p}^{n+1}$ ,  $\bar{u}^{n+1}$ , be the values obtained by the procedure leading up to the equations (2.12) i.e., not including the effect of source terms. We obtain  $p_{j+1/2}^{n+1}$ ,  $u_{j+1/2}^{n+1}$ , by solving the following set of linear equations, which approximate (2.9)<sub>±</sub>

$$\frac{1}{W_{\pm}} - (p_{j+1/2}^{n+1} - p_{j+1/2,\pm}) \pm (u_{j+1/2}^{n+1} - u_{j+1/2,\pm}) + \frac{A'(r_{j+1/2,\pm})}{A(r_{j+1/2,\pm})} u_{j+1/2,\pm} c_{j+1/2,\pm} = 0,$$

where  $W_{\pm} = W(\bar{p}_{j+1/2}^{n+1}, p_{j+1/2,\pm}, \rho_{j+1/2,\pm})$  and  $c_{j+1/2,\pm} = c(r_{j+1/2,\pm})$  have already been obtained above in calculating the solution without source terms. After a little algebra, one finds

$$(2.14) \quad \begin{aligned} p_{j+1/2}^{n+1} &= \bar{p}_{j+1/2}^{n+1} - \left( \frac{A'(r_{j+1/2,+})}{A(r_{j+1/2,+})} u_{j+1/2,+} c_{j+1/2,+} \right. \\ &\quad \left. + \frac{A'(r_{j+1/2,-})}{A(r_{j+1/2,-})} u_{j+1/2,-} c_{j+1/2,-} \right) \Delta t \frac{W_+ W_-}{W_+ + W_-}, \\ u_{j+1/2}^{n+1} &= \bar{u}_{j+1/2}^{n+1} + \frac{\Delta t}{W_+ + W_-} \left( \frac{W_- A'(r_{j+1/2,-})}{A(r_{j+1/2,-})} u_{j+1/2,-} c_{j+1/2,-} \right. \\ &\quad \left. - \frac{W_+ A'(r_{j+1/2,+})}{A(r_{j+1/2,+})} u_{j+1/2,+} c_{j+1/2,+} \right). \end{aligned}$$

Given  $p_{j+1/2}^{n+1}$ ,  $u_{j+1/2}^{n+1}$ , the values for the other variables are obtained using (2.13).

This completes the calculation of  $U_{j+1/2}^{n+1}$ ,  $U_{j+1/2}^{n+1}$ . These values are then inserted into (2.2) to obtain  $U_j^{n+1}$ , the conserved quantities at the new time. The time step must satisfy the usual CFL condition for stability:

$$(2.15) \quad \Delta t \leq \sigma \max_j \left( \frac{\Delta r}{|u_j^n| + c_j^n} \right)$$

where  $0 < \sigma < 1$ . The smallest  $\sigma$  for which (2.15) is satisfied is called the CFL number for that time step of the calculation.

### 3. Numerical results.

**Boundary conditions.** In order to calculate  $U_j^{n+1}$ ,  $j = M_L, \dots, M_R$ , it suffices to specify  $q_j^n$ ,  $(\delta q)_j$ ,  $j = M_L - 1, \dots, M_R + 1$ . Then one has sufficient data to calculate  $U_{j+1/2}^n$ ,  $U_{j+1/2}^{n+1}$ ,  $j = M_L - 1, \dots, M_R$ , and the  $U_j^{n+1}$ 's. If we can specify  $q_j^n$ ,  $j = M_L - 3, \dots, M_R + 3$ , then it follows from (2.6) that we can calculate  $\delta q_j$ ,  $j = M_L - 1, \dots, M_R + 1$ . In one dimension, or for two-dimensional problems for which the boundaries are aligned with the mesh directions, this is straightforward. For example, for the left boundary, we have

$$\begin{aligned} \text{Reflecting wall:} & \quad q_{M_L-l} = q_{M_L+l-1}, \quad u_{M_L-l} = u_{M_L+l-1}, \\ \text{Continuation:} & \quad q_{M_L-l} = q_{M_L+1-l}, \\ \text{Inflow:} & \quad q_{M_L-l}^n = q_0(t^n), \end{aligned}$$

where  $q = \rho, p, v$  for the reflecting wall, and  $q = \rho, p, v, u$  for the subsequent boundary conditions, with  $l = 1, 2, 3$ . For a reflecting wall, we have chosen to change the slope limiting procedure slightly. We allow the values extrapolated to the wall to take on the values which are obtained at the wall by solving a Riemann problem with left and right states  $(U_L, U_R) = (U_{M_L-1}^n, U_{M_L}^n)$ . This procedure seems to improve the resolution of shock reflections in multidimensional calculations.

Specifically, we define  $p_{\text{lim}}$  to be one of the roots of

$$(3.1) \quad u_{M_L}^2 W(p_{\text{lim}}, p_{M_L}, \rho_{M_L})^2 - (p_{\text{lim}} - p_{M_L})^2 = 0$$

where  $p_{\text{lim}}$  is the root  $\leq p_0$  if  $u_{M_L} \leq 0$ . Then we define

$$u_{\text{lim}} = 0, \quad \rho_{\text{lim}} = \left( \frac{1}{\rho_{M_L}} - \frac{p_{\text{lim}} - p_{M_L}}{W^2} \right)^{-1}.$$

These are used in the equations for  $\delta_{\text{lim}} q_{M_L, M_L-1}$ ,  $q = p, \rho, u$ :

$$\begin{aligned} \delta_{\text{lim}} q_{M_L} &= \begin{cases} \min(2|q_{M_L} - q_{M_L+1}|, 2|q_{M_L} - q_{\text{lim}}|) & \text{if } (q_{M_L} - q_{\text{lim}})(q_{\text{lim}} - q_{M_L+1}) > 0, \\ 0 & \text{otherwise,} \end{cases} \\ \delta_{\text{lim}} q_{M_L-1} &= \begin{cases} \min(2|q_{M_L} - q_{M_L-1}|, 2|q_{M_L} - q_{\text{lim}}|) & \text{if } (q_{M_L} - q_{\text{lim}})(q_{\text{lim}} - q_{M_L-1}) > 0, \\ 0 & \text{otherwise.} \end{cases} \end{aligned}$$

The corresponding procedure for a reflecting wall at  $M_R$  is obtained by exchanging  $>$ ,  $<$  in choosing the root of (3.1), and replacing  $M_L + 1$ ,  $M_L$ ,  $M_L - 1$  with  $M_R - 1$ ,  $M_R$ ,  $M_R + 1$ .

If  $u_{M_L-1/2}$  is the axis of symmetry for a cylindrically or spherically symmetric problem, then we treat it as a reflecting wall, except that the geometric source terms  $\pm ucA'/A$  in the characteristic equations (2.9) are set equal to zero in calculating  $U_{M_L-1/2}^{n+1}$ , i.e.,  $\bar{q}_{M_L-1/2}^{n+1} = q_{M_L-1/2}^{n+1}$ .

Finally, in the diverging duct problem discussed below, we use a characteristic boundary condition at the right-hand side of the duct. The density  $\rho_0$  is specified to be a constant at the right end of the duct. Then, as a function of time values of  $p, u$  are specified using the characteristic equations, using the assumption that the  $-$ ,  $0$  characteristics point to the right:

$$(3.2) \quad \begin{aligned} \rho_{M_R+l} &= \rho_0, \quad p_{M_R+l} = \left( \frac{\rho_0}{\rho_{M_R}} \right)^\gamma p_{M_R}, \quad \bar{\rho c} = \frac{1}{2} (\sqrt{\rho_{M_R} p_{M_R} \gamma} + \sqrt{\rho_{M_R+l} p_{M_R+l} \gamma}), \\ u_{M_R+l} &= u_{M_R} - \frac{(p_{M_R+l} - p_{M_R})}{\bar{\rho c}}. \end{aligned}$$



**Test problems.** This method has been tested on a variety of test problems in one space dimension, including shock tubes in Cartesian, cylindrical and spherical geometry. Results were obtained for one-dimensional problems which were indistinguishable from those shown in [7] and [1], obtained using the Lagrange plus remap MUSCL. This method has also been used to calculate the oblique reflection of a shock against an inclined plane in two space variables [11], successfully resolving multiple Mach stem configurations.

We present here two test calculations. As a one-dimensional test problem, we calculated the steady state solution to the duct flow problem in Shubin, Stephens, and Glaz [6], marching in time until the steady state was reached. The duct is specified by  $A(r) = 1.398 + .347 \tanh(.8r - 4)$ ,  $0 \leq r \leq 10$ , with boundary conditions

$$p(0, t) = .3809, \quad \rho(0, t) = .502, \quad u(0, t) = 1.299, \quad \rho(10, t) = .776, \quad t \geq 0.$$

The initial conditions are given by setting  $q(x, 0) = q(0, 0)$ , i.e., impulsive start. Inflow boundary conditions are imposed at the left boundary, and the characteristic boundary conditions (3.2) are imposed at the right boundary. The density profiles at  $t = 200$  are shown in Fig. 3, for  $\Delta r = \frac{5}{8}$  and  $\Delta r = \frac{5}{16}$ , plotted as a dotted line, with circles

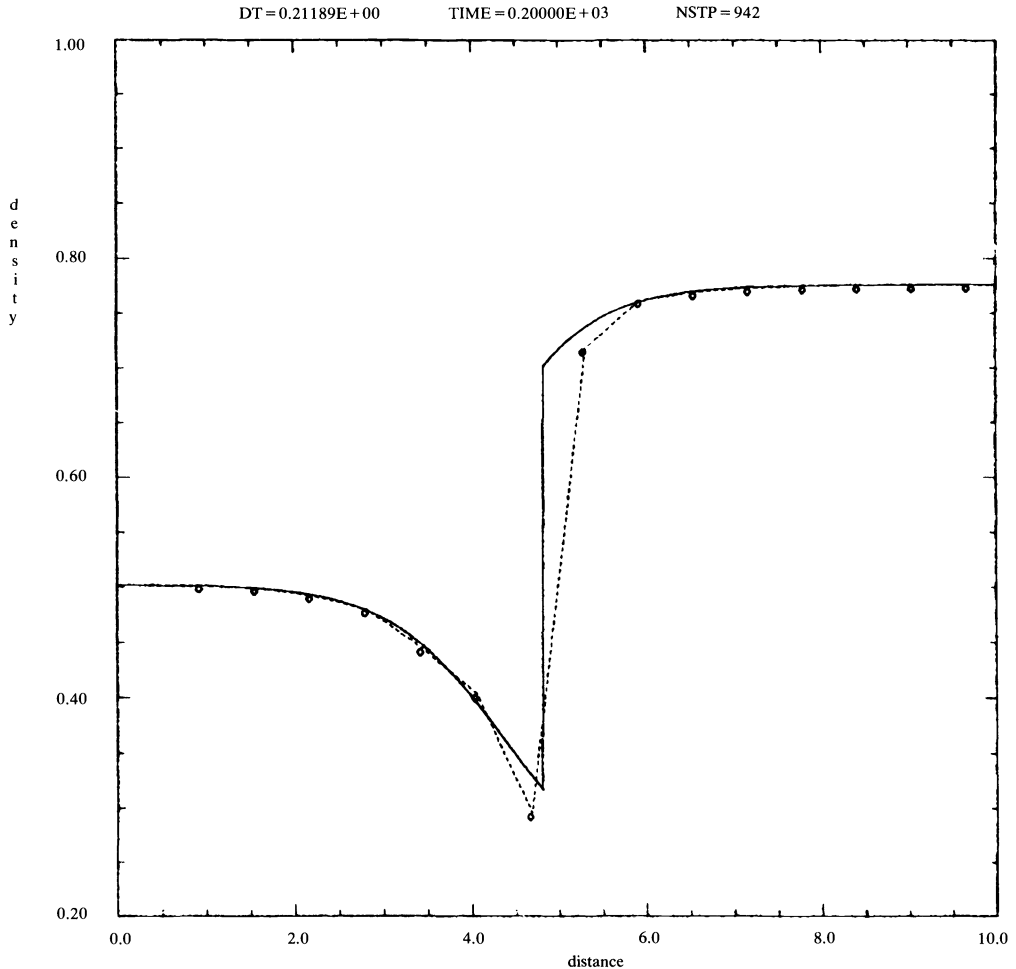


FIG. 3. Steady state density profiles for one-dimensional duct problem. a)  $\Delta r = \frac{5}{8}$ , b)  $\Delta r = \frac{5}{16}$ .

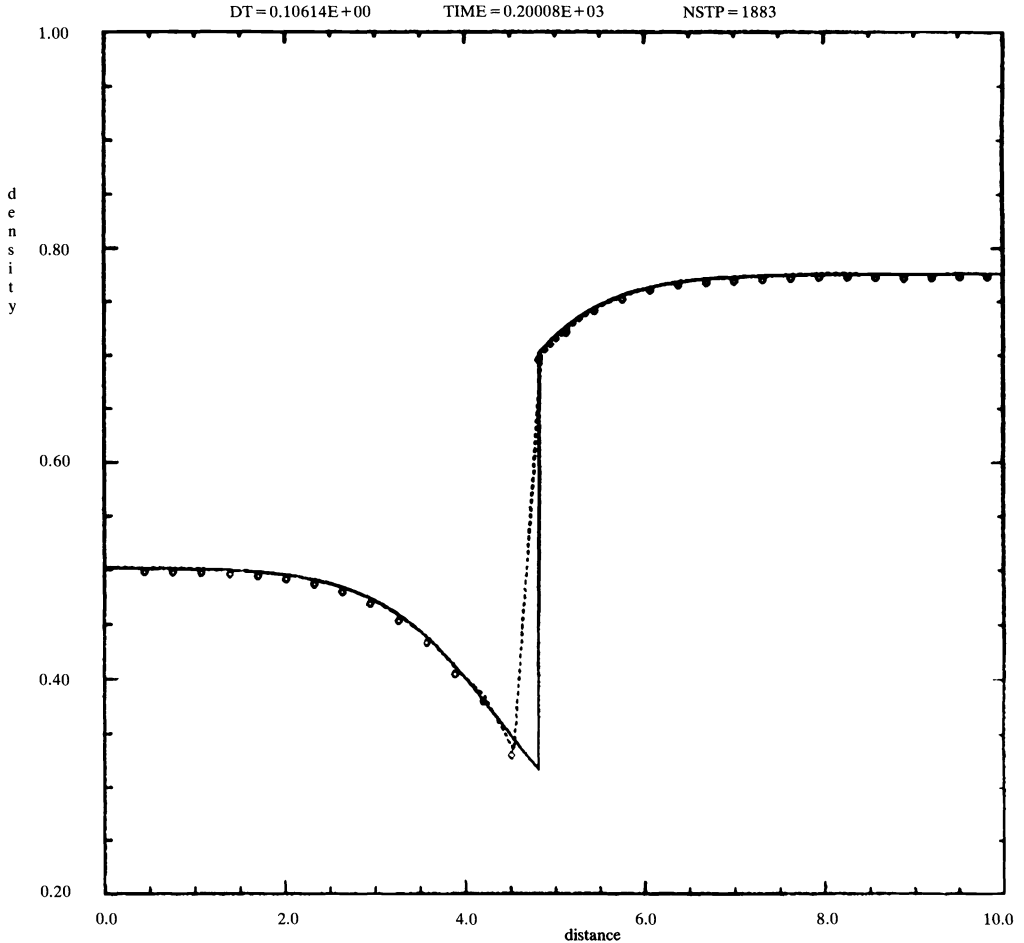


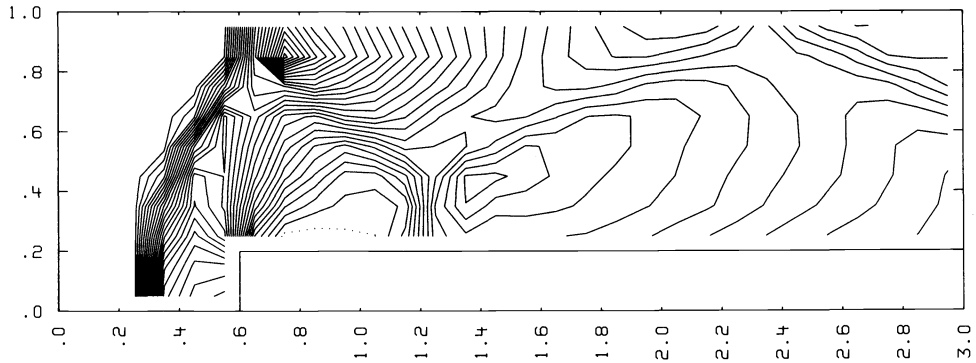
FIG. 3 (cont.).

at the data points. This is to be compared with the exact solution, plotted as a solid line. We obtain good agreement with the exact solution, even for the coarsely zoned calculation.

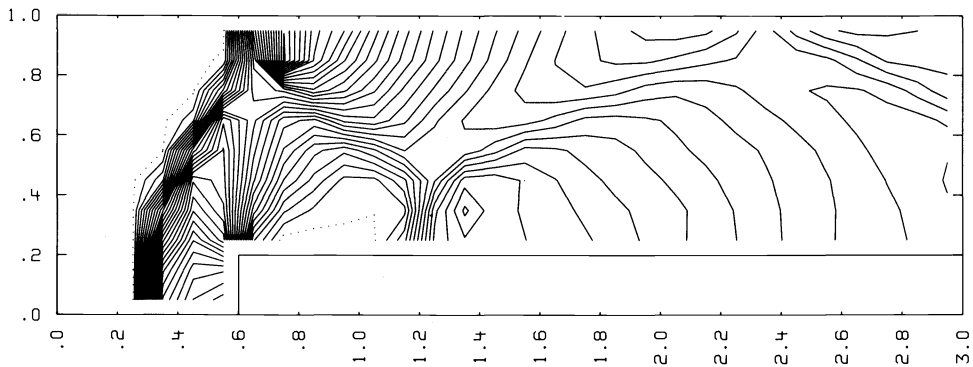
We also calculated the two-dimensional Cartesian shock reflection problem used by van Leer [7] as a test problem for the Lagrange plus remap versions of MUSCL; see also Woodward and Colella [9]. The computational domain is a channel of length 3 in the  $x$  direction, and of width 1 at the left end in the  $y$  direction, with a step of height 2 extending to the right beginning at  $x = 6$ . The step and the upper and lower walls of the channel are reflecting boundaries, with a Mach 3 uniform inflow on the left, and continuation boundary conditions on the right. The initial conditions are that of uniform flow throughout the channel:

$$\rho(x, y, 0) = 1, \quad \rho(x, y, 0) = 1.4, \quad u_x(x, y, 0) = 3, \quad u_y(x, y, 0) = 0.$$

In Figs. 4 and 5, we show the density and pressure contours of the solution at  $t = 4$ , with  $\Delta x = \Delta y = .1$  and  $.05$ , respectively. The first shock reflection point along the upper wall has been seen in other calculations [9] to be a Mach reflection, located directly above the edge of the step. The present calculations obtain the correct location of the reflection point, although the Mach stem in the  $\Delta x = .1$  is two zones long;



(a)



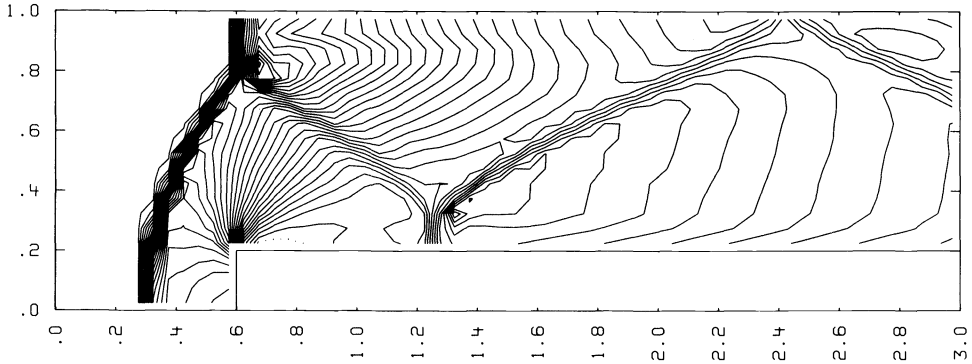
(b)

FIG. 4. Contour plots for two-dimensional test problem,  $\Delta x = .1$ . a) Density, 30 contours between .98 and 6.38. b) Pressure, 30 contours between 1.11 and 11.6.

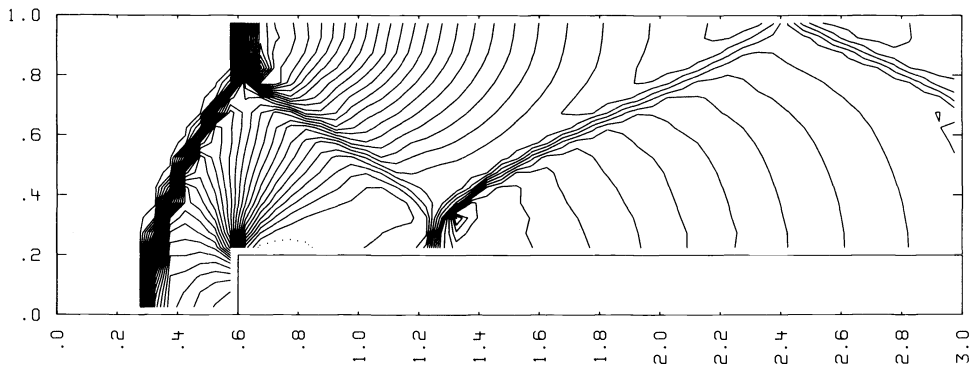
consequently, the slip line extending to the right from the triple point is not resolved, as it is in the  $\Delta x = .05$  calculation. The other reflected shocks are well resolved in both calculations, even though they are quite weak.

These results represent an improvement over the the results in [7] in two respects. First, the overall resolution of the shocks, particularly in the  $\Delta x = .1$  calculation, is substantially better. Second, the numerical boundary layer generated at the corner along the upper surface of the step is far weaker than that generated in the Lagrange plus remap results. In the latter calculation, the boundary layer separates at  $x = 1$ , changing somewhat the shock pattern downstream. The numerical boundary layer does not separate in the present calculations.

These two-dimensional problems were run on the Cray-1 at LLNL using a fully vectorized implementation of the algorithm, the  $\Delta x = .1$  calculation taking .066 minutes to run 194 time steps, and the  $\Delta x = .05$  calculation taking .36 minutes to run 376 time steps. However, the vector lengths in these calculations were that of the number of zones in a one-dimensional sweep, and were hence too short to observe the full speed of a fully vectorized calculation on the Cray. A more typical speed for larger problems is  $20 \mu\text{s}/\text{zone}/\text{time step}/\text{dimension}$ .



(a)



(b)

FIG. 5. Contour plots for two-dimensional test problem,  $\Delta x = .05$ . a) Density, 30 contours between .68 and 6.29. b) Pressure, 30 contours between .72 and 11.8.

**4. Discussion and conclusions.** The direct Eulerian MUSCL algorithm described above follows the basic conceptual framework given by van Leer for the Lagrangian MUSCL scheme. There are, however, substantial technical differences, all aimed at producing a more robust, and in certain ways, simpler scheme. A central feature to the engineering of the scheme is that of solving the characteristic equations (2.8)–(2.9) directly, rather than, as in [7], deriving a formula based on Taylor expansions, for the time derivative of the flux. The present approach makes it much easier to account correctly for sonic points in rarefaction waves (2.11), to introduce tracing characteristics forward in time (2.10); and to exploit the duality between the Riemann problem and the characteristic equations for gas dynamics by introducing the nonlinear algorithm for calculating  $U_{j+1/2}^{n+1}$ . The latter two procedures were essential for calculating strong shocks with CFL numbers close to 1, and appear to be necessary for Lagrangian calculations using MUSCL as well [12].

We have presented here the basic framework for extending the Lagrangian algorithm of van Leer to Eulerian gas dynamics. This approach can be easily modified to an arbitrary moving coordinate system, in one dimension, or a moving rectangular

coordinate system in more than one dimension. A central issue which remains to be fully resolved for this method, as well as other higher order extensions of Godunov's method is controlling the behavior of such schemes when one of the characteristic speeds, measured relative to the mesh motion, vanishes. The treatment of sonic centered rarefaction waves and the flattening of slopes at nearly stationary shocks constitute a first step, but more work is required. A fuller analysis of these problems appears in [3], [10], along with some proposals for ameliorating them.

**Acknowledgments.** The author wishes to thank Paul Woodward for many helpful discussions, and Bram van Leer for a critical reading of the manuscript. The author also wishes to thank the Theoretical Physics Division at LLNL for making available the time on the LLNL Cray I.

## REFERENCES

- [1] P. COLELLA, *Glimm's method for gas dynamics*, this Journal, 3 (1982), pp. 76–110.
- [2] P. COLELLA AND H. M. GLAZ, *Efficient solution algorithms for the Riemann problem for real gases*, Lawrence Berkeley Lab., Report LBL-15776, Univ. California, Berkeley, 1983.
- [3] P. COLELLA AND P. R. WOODWARD, *The piecewise parabolic method for gas-dynamical simulations*, Report LBL-14661, Lawrence Berkeley Lab., Univ. California, Berkeley 1982, J. Comp. Phys., to appear.
- [4] S. K. GODUNOV, *Difference methods for the numerical calculation of the equations of fluid dynamics*, Mat. Sb., 47, (1959), pp. 271–306. (In Russian.)
- [5] S. K. GODUNOV, A. W. ZABRODYN AND G. P. PROKOPOV, *A computational scheme for two-dimensional nonstationary problems of gas dynamics and calculations of the flow from a shock wave approaching a stationary state*, USSR Comput. Math. Math. Phys., 1, (1961), pp. 1187–1218.
- [6] G. R. SHUBIN, A. B. STEPHENS AND H. M. GLAZ, *Steady shock tracking and Newton's method applied to one-dimensional flow*, J. Comp. Phys., 39, (1981), pp. 364–374.
- [7] B. VAN LEER, *Towards the ultimate conservative differences scheme, V. A second order sequel to Godunov's methods*, J. Comp. Phys., 32 (1979), pp. 101–136.
- [8] B. VAN LEER AND P. R. WOODWARD, *The MUSCL code for compressible flow: philosophy and results*, Proc. TICOM Conf., Austin, TX, 1979.
- [9] P. R. WOODWARD AND P. COLELLA, *High resolution difference schemes for compressible gas dynamics*, Lecture Notes in Physics 141, Springer-Verlag, New York, 1979.
- [10] ——— *The numerical simulation of two-dimensional fluid flow with strong shocks*, J. Comp. Phys., to appear.
- [11] P. COLELLA AND H. M. GLAZ, *Calculation of complex shocked flows using a direct Eulerian MUSCL algorithm*, paper presented at the 5th AIAA CFD Conference, Palo Alto, CA, June 1981.
- [12] P. R. WOODWARD, private communication.

## A ROOTFINDER USING A NONMONOTONE RATIONAL APPROXIMATION\*

G. K. KRISTIANSEN†

**Abstract.** A rootfinder using a nonmonotone rational approximation is described. To aid the analysis, a general expression in terms of divided differences is derived for the error associated with rational approximation.

**Key words.** Cauchy interpolation, rational interpolation, rootfinder

**1. Interpolation by means of rational functions.** First some notation: The set  $S$  of real-valued functions defined on a real interval  $I$  is an algebra over the real field with multiplication and addition defined pointwise. The function  $x \mapsto x$  will be denoted simply by  $x$ , so that, for instance, for  $x_0 \in I$  and  $f \in S$ , we have  $(x'f)(x_0) = x_0'f(x_0)$ . Let  $f \in S$ ,  $n$  be a nonnegative integer, and  $\{x_0, x_1, \dots, x_n, x\}$  be a set of  $n+2$  different points in  $I$ . Then, the  $(n+1)$ th divided difference  $f[x_0, x_1, \dots, x_n, x]$  can be defined as the ratio

$$(f(x) - P_n(x)) / \prod_{j=0}^n (x - x_j),$$

where  $P_n$  is the polynomial in  $x$  coinciding with  $f$  at the points  $x_j (0 \leq j \leq n)$ .

If we define the 0th divided difference by  $f[x_0] = f(x_0)$ , we can construct the divided differences recursively.

$$f[x_0, \dots, x_n] = (f[x_1, \dots, x_n] - f[x_0, \dots, x_{n-1}]) / (x_n - x_0).$$

For fixed values of the abscissas, the divided difference  $f[x_0, \dots, x_n]$  is a linear functional of  $f$ . For fixed  $f$  it is a symmetric function of the abscissas.

Finally we mention the useful identity

$$((x - x_n)f)[x_0, \dots, x_n] = f[x_0, \dots, x_{n-1}].$$

The general theory of Cauchy interpolation, i.e. pointwise interpolation by means of rational functions, is discussed for instance in [5]. We shall recall some facts: Let  $P_l$  and  $Q_m$  be polynomials of degree at most  $l$ , resp.  $m$ , with  $l+m=n$ , such that  $(P_l - Q_m f)(x_j) = 0$  ( $0 \leq j \leq n$ ). There is at least one set, but there may be several linearly independent sets of coefficients of  $P_l$  and  $Q_m$  solving this system of equations. However, the function  $R_{l,m} = P_l / Q_m$  is uniquely determined. If, for some point  $x_j$ ,  $R_{l,m}(x)$  does not have a limit equal to  $f(x_j)$  for  $x \rightarrow x_j$ , the Cauchy interpolation problem is unsolvable and the point  $x_j$  is called unattainable. We shall prove the following result.

**THEOREM.** *Let  $f: I \rightarrow \mathbb{R}$  be a function defined on the real interval  $I$ , and let  $x_j (0 \leq j \leq n+1)$  be  $n+2$  different points of  $I$ . Assume that the Cauchy interpolation problem  $f(x_j) = R_{l,m}(x_j)$  ( $0 \leq j \leq n$ ) is solvable. In the expression  $R_{l,m} = P_l / Q_m$ , where  $P_l$  and  $Q_m$  are polynomials of degree at most  $l$ , resp.  $m$ , with  $l+m=n$ , we assume that  $P_l$  and  $Q_m$  have no common factors. The exact degree of  $Q_m$  is  $m_1 \leq m$ . Then*

$$(1) \quad Q_m(x_{n+1}) = \sum_{j=0}^{m_1} q_j x_{n+1}^j = \det(q_{r,s}),$$

\* Received by the editors February 17, 1983, and in revised form August 30, 1983.

† Risø National Laboratory, Postbox 49, DK-4000 Roskilde, Denmark.

$$(2) \quad (f - R_{l,m})(x_{n+1}) = \frac{\det(t_{r,s})}{\det(q_{r,s})} \prod_{j=0}^n (x_{n+1} - x_j),$$

where, for  $0 \leq s \leq m_1$ ,

$$t_{0,s} = f[x_s, \dots, x_{n+1}], \quad t_{r,s} = f[x_s, \dots, x_{n+1-r}] \quad \text{for } 1 \leq r \leq m_1,$$

$$q_{0,s} = \prod_{j=0}^{s-1} (x_{n+1} - x_j), \quad q_{r,s} = f[x_s, \dots, x_{n+1-r}] \quad \text{for } 1 \leq r \leq m_1.$$

*Remark.* In the limit where the points  $x_j (0 \leq j \leq n + 1)$  coincide, we get Tornheim's result [7] for the ratio

$$(f - R_{l,m})(x_{n+1}) \bigg/ \prod_{j=0}^n (x_{n+1} - x_j).$$

The above expression for the error in rational approximation is, of course, valid also under weaker assumptions than those stated.

*Proof.* The polynomial of degree at most  $n$  coinciding with the function  $P_l - Q_m f$  at the points  $x_j (0 \leq j \leq n)$  must be identically zero. Therefore, it follows from the definition of the divided difference that for  $x_{n+1} \in I \setminus \{x_0, x_1, \dots, x_n\}$

$$(3) \quad \frac{1}{\prod_{j=0}^n (x_{n+1} - x_j)} (P_l - Q_m f)(x_{n+1}) = (P_l - Q_m f)[x_0, \dots, x_{n+1}] \\ = - \sum_{j=0}^m q_j(x^j f)[x_0, \dots, x_{n+1}].$$

We have used that  $P_l$  is of degree at most  $l = n - m \leq n$ , so that any divided difference of  $P_l$  of order at least  $n + 1$  will be 0. Similarly, we get for  $0 \leq k \leq m - 1$ ,

$$(x^k Q_m f)[x_0, \dots, x_n] = (x^k P_l)[x_0, \dots, x_n] = 0,$$

i.e.,

$$(4) \quad \sum_{j=0}^m q_j(x^{j+k} f)[x_0, \dots, x_n] = 0.$$

As noted above we assume that the given rational interpolation problem has a solution which is then unique. But we cannot be certain that the polynomial  $Q_m$  has exact degree  $m$  (i.e.,  $q_m \neq 0$ ). Let  $m_1$  be the exact degree of  $Q_m$ . Then (4) is equivalent to

$$(5) \quad \sum_{j=0}^{m_1} q_j(x^{j+k} f)[x_0, \dots, x_n] = 0.$$

This is true for  $0 \leq k \leq m - 1$ . We shall show that the rank of this system of equations is  $m_1$ . It cannot be greater since we know that a nonvanishing solution  $Q_m$  exists. If it were smaller, the null-space of the coefficient matrix of (5) would have dimension at least two; thus we would have two linearly independent solutions  $(q_j | 0 \leq j \leq m_1)$  and  $(q_j^{(1)} | 0 \leq j \leq m_1)$ , corresponding to the polynomials  $Q_m$  and  $Q_{m_1}$ .

But from the equations

$$(6) \quad (P_{l_1} - Q_{m_1} f)(x_j) = 0 \quad (0 \leq j \leq l)$$

the coefficients of  $P_{l_1}$  (where  $l_1 \leq l$ ) are determined uniquely from those of  $Q_{m_1}$  (the coefficient matrix is of Vandermonde-type). If  $P_{l_1}$  and  $Q_{m_1}$  satisfy (5) and (6), we can show (see below) that the equations of the form (6), but with  $l + 1 \leq j \leq n$ , are satisfied too.

But then we must have  $P_l Q_{m_1} = P_{l_1} Q_m$ ; as  $P_l$  and  $Q_m$  have no common factors,  $Q_m$  must divide  $Q_{m_1}$ . The degree of  $Q_{m_1}$  is at most equal to  $m_1$ , the degree of  $Q_m$ , so that  $Q_{m_1}/Q_m$  must be constant, which contradicts the linear independence of the two solutions.

Assume, then, that we have already shown that  $(P_{l_1} - Q_{m_1} f)(x_j) = 0$  for  $0 \leq j \leq L$ , where  $l \leq L \leq n - 1$ . To deduce that  $(P_{l_1} - Q_{m_1} f)(x_{L+1}) = 0$ , we need only show that  $(P_{l_1} - Q_{m_1} f)[x_0, \dots, x_{L+1}] = 0$ , i.e., that

$$\sum_{j=0}^{m_1} q_j^{(1)}(x^j f)[x_0, \dots, x_{L+1}] = 0.$$

However, from (5) it follows that

$$\sum_{j=0}^{m_1} q_j^{(1)}(P_{m-1} x^j f)[x_0, \dots, x_n] = 0,$$

where  $P_{m-1}$  is an arbitrary polynomial of degree at most  $m - 1 = n - l - 1 \geq n - L - 1$ . If we choose

$$P_{m-1}(x) = \prod_{j=L+2}^n (x - x_j),$$

we get the desired result.

The set of coefficients of  $Q_m$  can then be chosen proportional to the set of cofactors to the last row in the coefficient matrix of (5). Both  $Q_m(x_{n+1})$  and the right-hand side of (3) can then be written as determinants. After some elementary transformations we get the equations (1) and (2).

**2. Description of the rootfinder HYPAR.** The most efficient rootfinders in common use employ monotone functions for higher order approximation (for instance inverse interpolation [1] or homographic interpolation [4]). Often the given function  $f$  has several roots (perhaps an infinite number), and although it is usually possible in a simple way to find an interval  $[a, b]$  containing the wanted root  $\zeta$ , it is more difficult to ensure that the restriction of  $f$  to  $[a, b]$  is monotone. If a traditional rootfinder is used, an interval of monotonicity containing the root is found by primitive methods (for instance bisection). Parabolic interpolation does not seem to be much better than these methods (apparently the approximation of nonmonotone functions is usually too inaccurate); however, the method described below using an  $R_{2,1}$  rational approximation has proved satisfactory.

Going back for a moment to the general situation discussed earlier, we shall briefly discuss the occurrence of unattainable points. As noted above, we can always determine solutions  $P_l$  and  $Q_m$  to the equations  $(P_l - Q_m f)(x_j) = 0$  ( $0 \leq j \leq n$ ). Each unattainable point is a root of both  $P_l$  and  $Q_m$ , which means that the rational function

$$R_{l,m} = \frac{P_l}{Q_m}$$

after removal of common nonconstant factors of  $P_l$  and  $Q_m$  has a sum of degrees of numerator- and denominator-polynomials which is less than the number of attainable points, so that  $R_{l,m}$  is, in fact, the unique solution to the Cauchy problem restricted to these points. For the present case ( $l = 2, m = 1, n = 3$ ), an unattainable point is a point not on some straight line, which contains the remaining three points. Moreover, we should avoid the case where the singularity of  $R$  is in the smallest interval containing



the four interpolation points. To find a simple criterion we write

$$R(x) = ax + b + \frac{c}{x-d}.$$

Evidently the graph  $\{(x, R(x)) | x \in \mathbb{R}\}$  is a hyperbola with asymptotes  $y = ax + b$  and  $x = d$ . Let  $x_0 < x_1 < x_2 < x_3$ . Then

$$R[x_0, x_1, x_2]R[x_1, x_2, x_3] = \frac{c^2}{(x_0-d)(x_1-d)^2(x_2-d)^2(x_3-d)},$$

which is positive if and only if all points are on the same branch, and  $c \neq 0$  ( $c = 0$  would mean that all points were on the same straight line).

But the requirement  $f[x_0, x_1, x_2]f[x_1, x_2, x_3] > 0$  is, in fact, sufficient to ensure that the  $R_{2,1}$ -approximation is applicable since it is violated if just 3 of the points are on a line.

If, for instance,  $f[x_0, x_1, x_3] = 0$ , but  $f[x_0, x_1, x_2] \neq 0$ , we have

$$f[x_0, x_1, x_2, x_3] = \frac{f[x_0, x_1, x_2]}{x_2 - x_3} = \frac{f[x_1, x_2, x_3]}{x_2 - x_0},$$

so that  $f[x_0, x_1, x_2]$  and  $f[x_1, x_2, x_3]$  have opposite signs.

To explain the use of  $R_{2,1}$ -approximation HYPAR we change the notation to that used in this procedure and set  $z_j = (x_j, y_j)$  ( $1 \leq j \leq 4$ ) for four points of the graph of  $f$ . The root sought,  $\zeta$ , is in the interval  $[x_2, x_3]$ , while  $y_1$  has the same sign as  $y_2$ , and  $z_4$  is the point that was  $z_1$  in the previous step; the point  $z_2$  is always the latest point calculated. We see that  $z_2$  becomes either  $z_1$  or  $z_3$  in the next step;  $z_1$  becomes  $z_4$ , and  $z_4$  is forgotten; but  $z_3$  may stay  $z_3$ , namely, if the new ordinate  $y_2$  has the same sign as the old ordinate  $y_2$  (which now becomes  $y_1$ ). This shows that even if the  $R_{2,1}$ -approximation is applicable in each step, only the three most recent points are certain to be among the four interpolation points. If the  $R_{2,1}$ -approximation is not possible, parabolic (i.e.,  $R_{2,0}$ -) interpolation is used. If the average convergence rate for the latest hypmax+1 iterations (hypmax is a built-in parameter with 10 as its recommended value) is below that of bisection, all future steps use bisection. Thus there is a difference in strategy between this method and the usual approach (see the next section).

An investigation of the asymptotic error for rootfinding based on Cauchy interpolation using only the latest calculated points is given in [7]. An error analysis for the special case of  $R_{2,1}$ -approximation, when we are not very close to the solution, is given in the following. It should be noted that some of the results below may be found elsewhere in the literature (see for instance [8, pp. 336–338]).

Substituting  $l = 2$  and  $m = m_1 = 1$  in (1), we get

$$\begin{aligned} Q(x_4) &= f[x_1, x_2, x_3] - (x_4 - x_0)f[x_0, x_1, x_2, x_3] \\ (7) \quad &= \frac{1}{x_3 - x_0} ((x_3 - x_0)f[x_1, x_2, x_3] - (x_4 - x_0)(f[x_1, x_2, x_3] - f[x_0, x_1, x_2])) \\ &= \frac{1}{x_3 - x_0} ((x_3 - x_4)f[x_1, x_2, x_3] + (x_4 - x_0)f[x_0, x_1, x_2]), \end{aligned}$$

where we have used some elementary identities satisfied by divided differences. If  $x_0 < x_1 < x_2 < x_3$ , and if  $x_4 \in (x_0, x_3)$ , we see that  $Q(x_4)$  has the same sign as  $f[x_1, x_2, x_3]$  and  $f[x_0, x_1, x_2]$  (remember that the  $R_{2,1}$ -approximation is used only if  $f[x_1, x_2, x_3] \times$

$f[x_0, x_1, x_2] > 0$ ). The error in  $x_4$  is determined by substitution in equation (2):

$$(f - R_{2,1})(x_4) = \left( \frac{(x_4 - x_0)(x_4 - x_1)(x_4 - x_2)(x_4 - x_3)}{Q(x_4)} \right) \cdot (f[x_0, x_1, x_2, x_3, x_4] \cdot f[x_1, x_2, x_3] - f[x_1, x_2, x_3, x_4]f[x_0, x_1, x_2, x_3]).$$

We see that if  $f$  has a bounded 4th derivative, and if  $|f''|$  has a positive lower bound in either  $[x_0, x_2]$  or  $[x_1, x_3]$ , we can find a constant  $A$ , such that

$$(8) \quad |f(x_4) - R_{2,1}(x_4)| \leq A |(x_4 - x_0)(x_4 - x_1)(x_4 - x_2)(x_4 - x_3)|$$

for  $x_4 \in [x_0, x_3]$ .

If  $f''$  assumes the value 0 in both intervals, but  $f[x_0, x_1, x_2] f[x_1, x_2, x_3] > 0$ , we may still obtain an error-estimate of this type, if we assume that  $|f'''|$  has a positive lower bound in  $[x_0, x_3]$ . We have, in fact,

$$\begin{aligned} & \max \{ |f[x_0, x_1, x_2]|, |f[x_1, x_2, x_3]| \} \\ & > |f[x_1, x_2, x_3] - f[x_0, x_1, x_2]| \\ & = (x_3 - x_0) |f[x_0, x_1, x_2, x_3]| \geq \frac{x_3 - x_0}{6} \min \{ |f'''(x)| \mid x \in [x_0, x_3] \}. \end{aligned}$$

This gives  $|Q(x_4)| > A_1 \min \{(x_3 - x_4), (x_4 - x_0)\}$ , with  $A_1$  a positive constant.

Thus, in this case,

$$(9) \quad |f(x_4) - R_{2,1}(x_4)| \leq A_2 |(x_4 - x_1)(x_4 - x_2)| \max \{x_4 - x_0, x_3 - x_4\}$$

for  $x_4 \in [x_0, x_3]$ . (Of course, we still have  $f(x_j) = R_{2,1}(x_j)$  for  $j = 0, 1, 2, 3$ .)

If the  $R_{2,0}$ -approximation is used, we have an inequality of the type

$$(10) \quad |f(x_3) - R_{2,0}(x_3)| \leq A_3 |(x_3 - x_0)(x_3 - x_1)(x_3 - x_2)|$$

for  $x_3 \in [x_0, x_2]$  (assuming that  $f$ , for instance, has a bounded third derivative in  $[x_0, x_2]$ , where  $x_0 < x_1 < x_2$ ). Here we must also admit the fact that not all three points (but two of them) need be recent.

In conclusion, the interpolation error in point number  $n + 1$  will, for reasonably well-behaved functions, have a bound

$$(11) \quad A \prod_{j=n-p}^n |x_{n+1} - x_j|,$$

where  $A$  is a positive constant, and  $p$  may be 1, 2, or 3.

From this point onwards we can use the traditional arguments to derive lower bounds for the weak convergence order in the various cases. The convergence order is usually not very important for the success of a rootfinder; a value of 1.6 (corresponding to  $p = 1$  in (11)) suffices for most practical problems.

**3. A comparison with the rootfinder ZEROIN.** One of the best procedures available for solving real transcendental equations in one variable is ZEROIN, Brent's improvement of Dekker's algorithm (see [3]). One of its distinctive features is that it never gives up, but tries in each step to find a higher order approximation to the root. This is sometimes a drawback, since the number of function calls may be very large for pathological functions. In the example of the next section the necessary number of steps with ZEROIN is 264, while the number,  $n$ , of bisections needed is 29, and the number of iterations with HYPAR is 36. In general, the maximum number of

function calls in HYPAR is  $n + \max\{\text{hypmax}, n - 1\}$ , while the bound given by Brent [1] for the number of ZEROIN function calls is  $n^2$  (the example shows that something like this can be attained, although only by means of specially contrived examples). HYPAR changes to bisection for all future steps, as soon as the apparent convergence rate is slower than for bisection, or when the distance between the two latest approximations to the root has come down to the user-specified tolerance,  $\text{eps}$ . One might say that HYPAR assumes that the function  $f$  has the same character far from the root  $\zeta$  as close to  $\zeta$ . In case  $f$  changes character, being difficult far from the root and well-behaved close to  $\zeta$ , the procedure ZEROIN should be preferred. In [2] a good example is given of a type of function changing character: Set  $f(x) = x^n - \zeta^n$ , where  $\zeta \neq 0$ , and  $n$  is a large odd integer. This function looks like  $x^n$ , a difficult function with a multiple root, seen from afar, but well-behaved close to the root  $\zeta$ . The algorithms of Bus and Dekker [2] are quite successful for these functions, in particular the procedure ZEROINRAT, which has a high asymptotic order of convergence.

ZEROIN does not answer all meaningful questions. Assume that a user wants to determine the root  $\zeta$  of a function with the greatest possible accuracy; that is, if  $\text{macheps}$  is the least positive number such that  $x$  and  $x \cdot (1 + \text{macheps})$  are distinguishable by the computer for all nonzero values of  $x$  that are of interest, the user wants two numbers  $x_1$  and  $x_2$ , such that  $f(x_1) \cdot f(x_2) \leq 0$ , with  $x_2 = x_1 \cdot (1 + \text{macheps})$ . This is impossible in ZEROIN (but not in HYPAR), since the internal tolerance in ZEROIN is corrected for round-off.

**4. Examples.** The first version of HYPAR was written in 1965, so that some experience with this procedure has accumulated over the years. To get an idea of its performance relative to ZEROIN we used Naur's set of functions (see [6]), designed for testing of students' algorithms. The total number of function calls for ZEROIN was 105, which is as good as any of the results obtained with the algorithms tested by Naur. The number of function calls for the improved algorithms ZEROIN and ZEROINRAT of Bus and Dekker were 127 and 151, resp. But HYPAR did with the

TABLE 1

HYPAR		ZEROIN	
$x$	$f(x)$	$x$	$f(x)$
-5	-121	-5	-121
10	989	10	989
2.5	12.1	-3.3648649	-35.7
2.3333333	9.37	-2.7035154	-18.1
1.7705266	2.78	-2.0435732	-7.49
1.5466392	1.15	3.9782134	58.0
1.4004592	0.35	-1.3546140	-2.13
1.3421611	$7.6_{10^{-2}}$	-1.0903881	-1.21
1.3264413	$7.4_{10^{-3}}$	-0.7544571	-0.67
1.3247613	$1.8_{10^{-4}}$	1.6118782	1.58
1.3247181	$4.7_{10^{-7}}$	-0.0448915	-0.96
1.3247180	$1.5_{10^{-11}}$	0.7834933	-1.30
1.3247170	$-4.3_{10^{-6}}$	1.1976857	-0.48
		1.3736061	0.22
		1.3186178	$-2.6_{10^{-2}}$
		1.3244480	$-1.2_{10^{-3}}$
		1.3247181	$4.9_{10^{-7}}$
		1.3247176	$-1.6_{10^{-6}}$

amazing number of 77 function calls. Of course, a bit of luck is involved, but some of the functions have some of the features which HYPAR was designed to handle. Take, for instance, the function  $f(x) = x^3 - 1 - x$  in the interval  $[-5, 10]$ . Its intervals of monotonicity are  $[-5, -1/\sqrt{3}]$ ,  $[-1/\sqrt{3}, 1/\sqrt{3}]$  and  $[1/\sqrt{3}, 10]$ . The accuracy required is  $10^{-6}$ . HYPAR exhibits one-sided convergence with the point  $(-5, -121)$  fixed; the order of convergence is 1.84; the number of function calls is 13. It takes ZEROIN 12 iterations to find the interval of monotonicity containing the root, and the total number of function calls is 18 (the order of convergence is again 1.84). This example also shows one of the less elegant features of HYPAR: In case of one-sided convergence an “ $\varepsilon$ -step” is taken to ensure that a root has been found. If this fails, the function is considered pathological, and the method changes to bisection. The list of function calls is given in Table 1.

The function constructed to fool ZEROIN is the following (described as an Algol procedure; the interval is  $[0, 1]$ , and the tolerance  $\text{eps} = 10^{-8}$ ).

```
real procedure f(x);
value x; real x;
begin
integer m; real t, e;
e := ln(3)/ln(2);
m := if x > 2*eps then entier(ln((x-eps)/2)/ln(0.4)) else 18;
t := (0.4)**m;
f := if m > 17 then -1 else ((5*x/t-4)**e)/9;
end;
```

(remark concerning notation: the exponentiation operator is written \*\*).

The graph of this function consists of a number of arcs, each pointing towards a false root. After the procedure has been led down one arc with an apparent convergence rate better than for bisection, it continues on the next arc.

HYPAR has been used, for instance, for determination of extremal eigenvalues of differential operators on  $\mathbb{R}$  with piecewise constant coefficients. In each region the eigenfunction is written as a linear combination of particular solutions. For each interface or boundary-condition we get a linear equation in the expansion coefficients. The eigenvalue sought is a root of the determinant of the resulting system of homogeneous linear equations. Bounds for the eigenvalue are easy to find; but they are not sufficiently accurate to guarantee monotonicity of the determinant.

### Appendix.

*The Algol procedure HYPAR*

```
boolean procedure hypar (f, x, y, d, eps);
real x, y, d, eps; real procedure f;
comment
```

*purpose:* The boolean procedure hypar assumes the value true if the values  $f(x)$  and  $f(x+d)$  have the same sign, otherwise it changes the values of  $x$  and  $d$  so that we still have  $f(x)*f(x+d) \leq 0$ , but so that  $\text{abs}(d)$  is not greater than  $\max(\text{eps}, \text{mini}*\text{abs}(x))$ , where mini is the relative machine precision.

*description of parameters:*

*input parameters*

$f$ : real procedure, defining the number  $f(x)$ , when  $x$  belongs to the interval  $[x, x+d]$ .

$x$ : real, one of the endpoints of the initial interval.

$y$ : real, equals  $f(x)$ .

$d$ : real, nonzero,  $x+d$  is the other endpoint of the initial interval.

$\text{eps}$ : real, positive, the user-specified tolerance, see the explanation above.

*output parameters*

```

x: real, one endpoint of final interval.
y: real, equals f(x). we have  $\text{abs}(y) \cong \text{abs}(f(x+d))$ .
d: real, x+d is the other endpoint of the final interval. if y=0, d is 0.
begin real x1, x2, x3, x4, y1, y2, y3, y4, a, b, c, mini, eps1;
integer p, hypmax; boolean bool, full; real delta;
comment
the following value of mini is appropriate for a B7800 computer;
mini := 4*8*(-13);

comment
the latest hypmax+1 function evaluations are used to estimate the convergence rate;
hypmax := 10;
delta := 1/2**hypmax;
begin array cr [0: hypmax-1]; integer q;
label out, par, hyp, bis;
full := false;
bool := false; x1 := x; y1 := y;
if y=0 then go to out;
x2 := x := x+d; y2 := y := f(x);
if y=0 then go to out;
if sign(y1)*sign(y2) > 0 then
begin bool := true; go to out end;
d := -d/2; cr [0] := abs(d);
x := x+d; y := f(x);
if y=0 then go to out;
x3 := x2; y3 := y2;
if sign(y) = sign(y2) then
begin x3 := x1; x1 := x2; y3 := y1; y1 := y2 end;
x2 := x; y2 := y; p := 0;
go to par;
hyp: a := (y1-y4)*(y2-y3)*(x1-x2)*(x4-x3) + (x3-x2)*(x1-x4)*(y1-y2)
      *(y4-y3);
b := (y1*(x3-x2) - y3*(x1-x2))*(y4-y2)*(x1-x2)*(x3-x2)
      + (y4*(x3-x2) - y3*(x4-x2))*(y2-y1)*(x3-x2)*(x4-x2)
      + (y1*(x4-x2) - y4*(x1-x2))*(y2-y3)*(x1-x2)*(x4-x2);
c := (x1-x2)*(x3-x2)*(x4-x2)*y2*((x4-x3)*(y1-y3) + (x3-x1)*(y4-y3));

comment
r2,1-approximation if permitted. otherwise r2,0-approximation;
if sign((x1-x2)*(y3-y2) + (x3-x2)*(y2-y1)) ≠
(if (x4-x2)*(x3-x2) < 0 then sign((x1-x2)*(y4-y2) + (x4-x2)*(y2-y1))
else
sign((x4-x2)*(y3-y2) + (x3-x2)*(y2-y4)) then
par: begin a := (y2-y1)*(x3-x2) + (y3-y2)*(x1-x2);
      b := (y2-y1)*(x3-x2)**2 + (y3-y2)*(x1-x2)**2;
      c := y2*(x1-x2)*(x3-x2)*(x3-x1);
end;

y := b**2 - 4*a*c;
if y < 0 then y := 0; y := sqrt(y);
y := if b = 0 then (if a ≠ 0 then sign(x3-x2)*y/2/abs(a)
else (x3-x2)/2) else if b*c*(x3-x2) > 0 then
2*c/(b+sign(b)*y) else if a ≠ 0 then
(b+sign(b)*y)/2/a else c/b;

```

```

comment
the new approximation must lie in [x2, x3];
if sign (y) ≠ sign (x3 - x2) then y := 0;
if sign (y - x3 + x2) = sign(x3 - x2) then y := x3 - x2;
p := (p + 1) mod hypmax; if not full then full := p = 0;

comment
change to bisection if this appears to be faster;
if full then
begin if abs(y) > delta*cr[p] then to to bis;
cr[p] := abs(y);
end;
x := x + y;
d := abs(x2 - x); x4 := x1; y4 := y1; y := f(x);
if y = 0 then go to out;
if sign (y) = sign (y3) then
begin. x1 := x3; x3 := x2; y1 := y3; y3 := y2 end
else begin x1 := x2; y1 := y2 end;
x2 := x; y2 := y;
if abs (x3 - x2) < eps then to to out;
if d > eps then go to hyp;
eps1 := max (eps, mini*abs(x));
comment eps-step;
x := x + eps1*sign(x3 - x2); y := f(x);
if sign (y)*sign(y2) ≤ 0 then
begin x3 := x; y3 := y; go to out end;
x2 := x; y2 := y;
comment bisection;
bis: for x := (x3 - x2)/2 while abs(x) ≥
max (eps/2, mini*abs (x2)) do
begin x := x2 + x; y := f(x); if y = 0 then go to out;
if sign (y) = sign (y3) then
begin x3 := x; y3 := y end else
begin x2 := x; y2 := y end;
end bis;
out. if y = 0 then d := 0 else if not bool then
begin if abs(y2) < abs(y3) then
begin x := x2; y := y2; d := x3 - x2 end else
begin x := x3; y := y3; d := x2 - x3 end;
end not bool;
hypar := bool;
end;
end of hypar;

```

(Remarks concerning notation: Labels are declared. The expression  $x \bmod y$ , where  $x$  is an integer and  $y$  a positive integer, means that integer in  $[0, y-1]$  which is congruent to  $x$  modulo  $y$ . Capitals have been replaced by small letters throughout to make the reading easier.)

**Acknowledgments.** I am indebted to mag. scient. O. Lang Rasmussen, Risø, for critical reading of the manuscript, and to Professor Åke Björck for bringing Bus and Dekker's paper [2] to my attention.

#### REFERENCES

- [1] R. P. BRENT, *Algorithms for Minimization Without Derivatives*, Prentice-Hall, Englewood Cliffs, NJ, 1973.
- [2] J. C. P. BUS AND T. J. DEKKER, *Two efficient algorithms with guaranteed convergence for finding a zero of a function*, ACM Trans. Math. Software, 1 (1975), pp. 330-345.

- [3] G. E. FORSYTHE, M. A. MALCOLM, AND C. B. MOLER, *Computer Methods for Mathematical Computations*, Prentice-Hall, Englewood Cliffs, NJ, 1977.
- [4] G. K. KRISTIANSEN, *Zeros of arbitrary function*, BIT, 3 (1963), pp. 205–206.
- [5] J. MEINGUET, *On the solubility of the Cauchy interpolation problem*, in *Approximation Theory*, Proceedings of a symposium held in Lancaster, July 1969, Academic Press, London and New York, 1970, pp. 137–163.
- [6] P. NAUR, *Automatic grading of students' Algol programming*, BIT, 4 (1964), pp. 177–188.
- [7] L. TORNHEIM, *Convergence of multipoint iterative methods*, J. Assoc. Comp. Mach., 11 (1964), pp. 210–220.
- [8] H. WERNER, *Rationale Tschebyscheff-Approximation, Eigenwerttheorie und Differenzenrechnung*, Arch. Ratt. Mech. Anal., 13 (1963), pp. 330–347.

## THE GRAND TOUR: A TOOL FOR VIEWING MULTIDIMENSIONAL DATA\*

DANIEL ASIMOV†

**Abstract.** The grand tour is a method for viewing multivariate statistical data via orthogonal projections onto a sequence of two-dimensional subspaces. The sequence of subspaces is chosen so that it is dense in the set of all two-dimensional subspaces. Desirable properties of such sequences of subspaces are considered, and several specific types of sequences are tested for rapidity of becoming dense. Tabulations are provided of the minimum length of a grand tour sequence necessary to achieve various degrees of denseness in dimensions up to 20.

**Key words.** multivariate data, multidimensional data, exploratory data analysis, computer graphics, scatterplots, Andrews plot, grand tour

**1. Introduction.** The familiar “scatterplot” (of a finite sample of ordered pairs of variables) can be extraordinarily informative. Thus, it is very tempting to consider the  $p$ -dimensional scatterplot—a finite sample of ordered  $p$ -tuples of variables—and to devise ways to view it.

Even for  $p = 3$ , we have no magic pen that draws points in mid-air. Resorting to computer graphics [FFT], however, will permit us to see the three-dimensional scatterplot on a display screen just as if the points were drawn in mid-air. With the aid of a graphical input device like a “trackball,” we may even rotate the scatterplot in real time.

For  $p$  greater than 3, we are faced with serious problems. How can computer graphics technology be used, in conjunction with our visual abilities, to better grasp the structure of the  $p$ -dimensional data?

A simple answer to this question is to project the data orthogonally onto some two-dimensional subspace of  $p$ -dimensional Euclidean space, and then to view the resulting projected image.

A problem immediately arises: Which of the infinitely many two-dimensional subspaces shall we choose for viewing? The idea of the grand tour is to move through a sequence of projections, chosen to be dense in the set of all projections. As a result, we can view (or else have the computer apply some analysis or measurement to) a sequence of two-dimensional scatterplots which, asymptotically, come arbitrarily close to *all* 2-dimensional scatterplots projectable from the given data.

Historically, the grand tour is a descendant of the Andrews plot [Andr] which dates to 1972. This plot is often realized as a stationary set of function graphs  $y = f_i(t)$  where  $f_i(t) = x_1/\sqrt{2} + x_2 \sin t + x_3 \cos t + x_4 \sin 2t + x_5 \cos 2t + \dots$  for the  $i$ th data point  $(x_1, x_2, x_3, \dots, x_p)$ . This can, however, be interpreted also as a time sequence  $\{f_1(t), \dots, f_N(t)\}$  of points in  $R$ , where at time  $t_0$  we are viewing the dot-products of all the data points with the vector given by  $(1/\sqrt{2}, \sin t_0, \cos t_0, \sin 2t_0, \cos 2t_0, \dots)$ .

Then, in 1977, Paul and John Tukey [TT77] presented some further thoughts on Andrews plots, including an example of a dense curve of directions in  $R^4$ . They also considered briefly a two-dimensional (not necessarily dense) version of Andrews plots which they called “ouija” plots.

---

\* Received by the editors November 8, 1983. This work was supported by the U.S. Department of Energy under contracts DE-AC03-76SF00515 and DE-AT03-81-ER10843.

† Department of Computer Science, University of California, Berkeley, California 94720.



Meanwhile, real-time computer graphical visualization of three-dimensional (or higher) rotation had been achieved when the PRIM-9 system was implemented at the Stanford Linear Accelerator Center (SLAC) in the early 1970's [FFT].

Much of the work described herein was performed at Harvard University in 1980-81 and at SLAC in 1981-83.

**2. Overview.** In order to implement a grand tour on a computer graphics system, it is necessary to have an explicitly computable sequence of orthonormal 2-frames (a 2-frame is an orthonormal pair of vectors) in  $p$ -dimensional Euclidean space. The  $p$ -dimensional data is then projected, in turn, onto the 2-plane<sup>1</sup> spanned by each 2-frame. If desired, each projected image may be displayed on the screen, or else processed somehow by the computer (or both). We list below some desiderata for this sequence of 2-frames:

*Desiderata.* A) The sequence of planes should be *dense* in the space of all planes. Precisely, let  $G_{2,p}$  stand for the space of unoriented 2-planes through the origin in Euclidean  $p$ -space (a so-called "Grassmannian manifold"). Let  $P_1, P_2, \dots$  be the infinite sequence of 2-planes (spanned by the infinite sequence of 2-frames generating the grand tour). Then our condition A says that for every 2-plane  $P$  and for every  $\varepsilon > 0$ , there exists an  $n$  such that the distance  $d(P, P_n)$  from  $P$  to  $P_n$  is less than  $\varepsilon$ . (Our definition of the distance function  $d$  is in § 4.) Note that this denseness is not just a desideratum, but part of our definition of "grand tour."

B) Our sequence of planes should become dense in  $G_{2,p}$  *rapidly*. This means finding an efficient algorithm to compute the sequence of 2-frames and to project the  $p$ -dimensional data onto each pair of vectors in turn.

C) It would be useful for the sequence of planes to be uniformly distributed in  $G_{2,p}$ . That is to say, for each open measurable subset  $A$  of  $G_{2,p}$ , our sequence of planes  $P_1, P_2, \dots$  should pass through  $A$  with frequency proportional to the measure of  $A$ . We refer here to the invariant measure  $\mu$  on  $G_{2,p}$  (which is uniquely determined up to a positive constant factor). Precisely, we want

$$\lim_{n \rightarrow \infty} \frac{1}{n} \sum_{i=1}^n I_A(P_i) = \mu(A),$$

where  $I_A$  is the characteristic function of the set  $A$ .

D) Our sequence of planes should be continuous, in some sense, if its projections are to be apprehended by a human observer. Each plane should be perceptibly close to those planes just before and after it in the sequence. (This condition is of no importance in many applications of the grand tour in which no human observing occurs.)

E) For human observers, our sequence of planes should be as straight as possible. That is, if we think of the planes as being evenly-spaced points on a curve in  $G_{2,p}$ , then we should be able to choose that curve so that it is almost a geodesic. This is another way of assuring that the sequence of planes is both comprehensible to the observer, and also that it moves rapidly to new views, giving new information about the data being projected.

F) The grand tour ought to have a built-in degree of flexibility about it. This would enable the user to better optimize those qualities (among A) through E, for example) which may be important for the particular purpose he or she has in mind. Flexibility may be obtained by finding a parametric family of sequences of planes. There should then be some clear relationship between the parameter(s) and the desired

<sup>1</sup> *Note.* Unless otherwise specified, all "planes" referred to herein will be planes through the origin.

properties, so that the user can choose the parameter(s) wisely. It should also be possible to interactively change parameters after the grand tour has begun.

G) The sequence of planes should be reconstructible at any later occasion. In practice, this simply means that either the sequence of planes is chosen from a parametric family with parameters known to the user, or else there may be a pseudo-random component whose random number algorithm(s) and seeds(s) are known. It is, of course, desirable that in reconstructing a particular plane of our sequence, the other planes preceding it need not be computed all over again.

*Remarks.* R1. To require bona fide denseness of the infinite sequence of planes  $P_1, P_2, \dots$  is unnecessary for any real-world implementation of the grand tour. In particular, if we know in advance the number  $L$  of planes  $P_1, P_2, \dots, P_L$  we will be using, we can dispense entirely with the idea of an infinite sequence. We may also be able to better optimize the seven properties A) through G) once  $L$  is known.

R2. If the method for producing the sequence  $P_1, P_2, \dots$  is based on some random process, we will generally be able to claim properties such as denseness or uniformity as being “almost sure” rather than certain. (But this is almost surely sufficient for our purposes!)

R3. There is evidently a tradeoff between rapidity and continuity. This suggests using a *curve* of points in  $G_{2,p}$  and obtaining a sequence  $P_1, P_2, \dots$  by walking along this curve after choosing an appropriate stepsize. The human observer, of course, desires continuity. A machine alone, processing many planes, will rather need rapidity.

R4. To date, the known sequences  $P_1, P_2, \dots$  that are both *uniform* and *rapid* require sequences of pseudo-random numbers to compute them. Thus to achieve reconstructibility the algorithm and seed value of the pseudo-random sequence must be retained.

R5. Minor violations of uniformity are acceptable for the human observer. Strict uniformity is needed only when the computer is determining distributional properties of some statistic of two-dimensional scatterplots (see § 5).

R6. In all of the above, we have emphasized the choice of 2-planes  $P_1, P_2, \dots$ . In practice, however, when displaying a two-dimensional picture we must also choose its rotational position on the screen. This is accomplished by choosing not just a mere plane  $P_i$  but rather a pair of orthonormal vectors  $(\mathbf{v}_i, \mathbf{w}_i)$  spanning  $P_i$ : these are to be identified with the  $X$  and  $Y$  directions of the display screen. Even when no display is required, the computer must still hold internally a description of each plane  $P_i$ . The “2-frame”  $(\mathbf{v}_i, \mathbf{w}_i)$  is a convenient form for this information.

**3. Some specific grand tours.** In this section, we present three general methods for producing grand tours.

**I. Torus method.** The  $N$ -dimensional torus  $T^N$  may be defined as the Cartesian product of  $N$  identical unit circles. Equivalently,  $T^N$  may be thought of as Euclidean space  $R^N$  in which all arithmetic is performed modulo one. Symbolically,  $T^N \cong R^N / (2\pi Z^N)$  where  $Z^N$  is the integer lattice in  $R^N$ . It is well-known that dense curves may be found on  $T^N$  via the following.

PROPOSITION [HWTN]. *Let  $\{\lambda_1, \dots, \lambda_N\}$  be a set of real numbers that are linearly independent over the integers.<sup>2</sup> Then the curve  $\alpha: R \rightarrow T^N$  via  $\alpha(t) = (\lambda_1 t, \dots, \lambda_N t)$  has dense image in  $T^N$ . (Note that the coordinates  $\lambda_i t$  are interpreted modulo  $2\pi$ .)*

<sup>2</sup> Real numbers  $u_1, \dots, u_N$  are said to be *linearly independent over the integers* if the only sequence of integers  $\{K_1, \dots, K_N\}$  for which the equation  $\sum_{i=1}^N K_i u_i = 0$  holds is with all  $K_i = 0$ .

The special orthogonal group in dimension  $p$ , denoted  $SO(p)$ , is the set of all orthogonal  $p \times p$  matrices having determinant  $=+1$ .  $SO(p)$  has a topology induced from  $R^{p^2}$ , the space of all real  $p \times p$  matrices, and is in this way a compact manifold of dimension  $\frac{1}{2}(p^2 - p)$ .  $SO(p)$  may equivalently be thought of as the space of all rotations of the unit sphere in  $R^p$ . (As such, it is a “Lie group” [Chev].)

We let  $R_{ij}(\theta)$  denote the element of  $SO(p)$  which rotates the standard basis vector  $\mathbf{e}_i$  through an angle  $\theta$  towards the standard basis vector  $\mathbf{e}_j$  inside the  $i, j$  coordinate 2-plane of  $R^p$ , leaving fixed the orthogonal complement of this 2-plane.

We let  $G_{2,p}$  denote the space of all 2-planes in  $R^p$ . We also let  $V_{2,p}$  denote the space of all ordered pairs of orthonormal vectors in  $R^p$ . ( $V_{2,p}$  is topologized as a subset of  $R^p \times R^p$  and is compact.) We have the natural continuous surjections  $\pi: SO(p) \rightarrow V_{2,p}$  and  $\rho: V_{2,p} \rightarrow G_{2,p}$  given by  $\pi(Q) = (Q\mathbf{e}_1, Q\mathbf{e}_2)$  for any  $Q \in SO(p)$ , and  $\rho(\mathbf{v}, \mathbf{w}) =$  the 2-plane spanned by  $\mathbf{v}$  and  $\mathbf{w}$ , for any  $(\mathbf{v}, \mathbf{w}) \in V_{2,p}$ .

We are now ready to describe explicitly the torus method.

1. Let  $N = \frac{1}{2}(p^2 - p)$  and think of the coordinates of  $T^N$  as being indexed by all pairs  $i, j$  with  $1 \leq i < j \leq p$ .

2. Define a map  $f: T^N \rightarrow SO(p)$  via

$$f(x_{1,2}, \dots, x_{p-1,p}) = R_{12}(x_{1,2}) \circ \dots \circ R_{p-1,p}(x_{p-1,p}).$$

In words:  $f$  is the product of coordinate-plane rotations through angles determined by the toral coordinates. (Note: each  $x_{ij}$  is only well defined modulo  $2\pi$ , but since  $R_{ij}(\theta + 2\pi) = R_{ij}(\theta)$ ,  $f$  is well defined.)

3. We claim that  $f$  is a surjection. This fact was in essence discovered by L. Euler [MMCM]; the angles  $\{x_{ij}\}$  are referred to as “Euler angles.”

4. Choose real numbers  $\lambda_1, \dots, \lambda_N$  and a stepsize STEP such that the numbers  $\{2\pi, \text{STEP} \cdot \lambda_1, \dots, \text{STEP} \cdot \lambda_N\}$  are linearly independent over the integers. Use  $\lambda_1, \dots, \lambda_N$  to define the curve  $\alpha: R \rightarrow T^N$  via  $\alpha(t) = (\lambda_1 t, \dots, \lambda_N t)$  as in Proposition 1. Thus, we know that the image  $\alpha(R)$  of  $\alpha$  is dense in  $T^N$ .

5. We conclude, therefore that  $f \circ \alpha: R \rightarrow SO(p)$  has dense image  $f(\alpha(R))$  in  $SO(p)$ .

6. The discrete sequence  $\{f \circ \alpha(K \cdot \text{STEP}), K = 1, 2, \dots\}$  must therefore also be dense in  $SO(p)$ .

7. Finally, we define our sequence of 2-frames  $(\mathbf{v}_K, \mathbf{w}_K)$  as  $(\mathbf{v}_K, \mathbf{w}_K) = \pi \circ f \circ \alpha(K \cdot \text{STEP})$ ,  $K = 1, 2, \dots$ . By 6 above, this sequence must be dense in  $V_{2,p}$ .

8. We define our sequence of 2-planes  $P_1, P_2, \dots$  as, of course,  $P_K = \rho(V_K, W_K) = \rho \circ \pi \circ f \circ \alpha(K \cdot \text{STEP})$ .

It follows from 7 above that this sequence is dense in  $G_{2,p}$ . This concludes our description of how to compute a grand tour by the torus method.

*Remarks.* R1. The number  $N$ , the dimension of the torus used here, can be reduced from  $\frac{1}{2}(p^2 - p)$  to  $2p - 3$  (see Appendix). The resulting sequence of orthogonal matrices will no longer be dense in  $SO(p)$  but will be dense when pushed via  $\pi$  and  $\rho$  into  $V_{2,p}$  and  $G_{2,p}$ . This reduction achieves a considerable savings in computation time.

R2. The sequence given by  $\mathbf{z}_K = (K \cdot \text{STEP} \cdot \lambda_1, \dots, K \cdot \text{STEP} \cdot \lambda_N) \in T^N$  is uniformly distributed on  $T^N$ . But the maps  $f$ ,  $\pi$ , and  $\rho$  do not respect volumes. Thus, the sequences of 2-frames  $\{(\mathbf{v}_K, \mathbf{w}_K)\}$  and planes  $\{P_K\}$  are *not* uniformly distributed. This remark applies equally to the  $2p - 3$  version in the Appendix.

R3. The parameter STEP may be varied before, or even during, each run of the grand tour. The effect of increasing the size of STEP is to trade continuity for rapidity. More accurately, this is true for some range of values  $0 < \text{STEP} \leq M$ , after which there

is very little noticeable effect of STEP on either continuity (which is totally lost) or rapidity (which is at a maximal level).

R4. Although it is convenient to fix the values of  $\lambda_1, \dots, \lambda_N$  and vary STEP, it is in fact the vector  $\mathbf{x} = (\text{STEP} \cdot \lambda_1, \dots, \text{STEP} \cdot \lambda_N)$  in the torus  $T^N$  which determines the characteristics of the grand tour, torus method. If the total number  $L$  of planes to be used is known, then Korobov [MCTP] has determined vectors  $\mathbf{x}$  which behave optimally vis-a-vis the distribution of the sequence  $\mathbf{x}, 2 \cdot \mathbf{x}, 3 \cdot \mathbf{x}, \dots$  in  $T^N$ . It seems likely that Korobov coefficients will give rise to sequences of 2-frames and 2-planes which become dense rapidly, but their use is restricted to occasions when  $L$  is known in advance. Alternatively, some easy-to-compute values of  $\mathbf{x}$  seem to work very well. For example, two choices are

- a) Let  $\lambda_K = \sqrt{p_K}$  = the square root of the  $K$ th prime ( $p_1 = 2, p_2 = 3, \dots$ ). Let STEP = almost any irrational positive real.
- b) Let  $\lambda_K = e^K \bmod 1$  ( $e = 2.71828 \dots$ ) and again let STEP = almost any irrational positive real.

**II. At-random method.** In this method, each 2-frame is chosen independently, from the “uniform” distribution on  $V_{2,p}$ . This distribution is more accurately termed the “invariant” measure on  $V_{2,p}$ , because it is characterized up to constant factor by its invariance under the action of  $SO(p)$  on  $V_{2,p}$ . That is, if  $Q \in SO(p)$  and  $A \subset V_{2,p}$ , then we have for the invariant measure  $m$ ,

$$m(A) = m\{(Q\mathbf{v}, Q\mathbf{w}) \mid (\mathbf{v}, \mathbf{w}) \in A\}.$$

To pick the sequence  $\{(\mathbf{v}_K, \mathbf{w}_K)\}$  of 2-frames, we use the “rejection” method as follows:

1. Generate a sequence of pseudorandom numbers  $x_1, x_2, \dots$  in the unit interval.
2. Set  $y_1 = 2x_1 - 1, y_2 = 2x_2 - 1, \dots$ .
3. Assume we have already used the random numbers  $y_1, \dots, y_n$  (at the start  $n = 0$ ). Set  $z_i = y_{n+i}$  for  $i = 1, \dots, p$ .
4. Test for  $0 < z_1^2 + \dots + z_p^2 \leq 1.0$ . If not, return to Step 3 and try again.
5. Go through Step 3 again until a second set of  $p$  numbers are found (call them  $u_1, \dots, u_p$  this time) with  $0 < u_1^2 + \dots + u_p^2 \leq 1.0$ .
6. Letting  $\mathbf{z} = (z_1, \dots, z_p)$  and  $\mathbf{u} = (u_1, \dots, u_p)$ , apply the Gram-Schmidt procedure to obtain an orthonormal pair of vectors

$$\mathbf{v}_K = \mathbf{z} / \|\mathbf{z}\| \quad \text{and} \quad \mathbf{w}_K = \frac{\mathbf{u} - (\mathbf{u} \cdot \mathbf{v}_K)\mathbf{v}_K}{\|\mathbf{u} - (\mathbf{u} \cdot \mathbf{v}_K)\mathbf{v}_K\|}.$$

These constitute the next 2-frame  $(\mathbf{v}_K, \mathbf{w}_K)$  of our sequence. It is easy to verify that despite the apparent asymmetry in the use of the Gram-Schmidt procedure,  $(\mathbf{v}_K, \mathbf{w}_K)$  is in fact selected at random from the invariant distribution.

7. It follows immediately that the corresponding sequence of planes  $P_K = \langle \mathbf{v}_K, \mathbf{w}_K \rangle$  may be thought of as being selected from the corresponding invariant distribution on  $G_{2,p}$ .

*Remarks.* R1. The at-random method has in its favor the extreme ease of concept and computation. It is too discontinuous (totally) for movie viewing. (This is, of course, no problem if the viewer prefers to see only a sequence of still pictures.)

R2. The at-random method will produce, almost surely, a uniformly distributed sequence.

R3. There is no flexibility in the at-random method.

R4. The at-random method becomes dense about as fast as the torus method with large stepsize.

**III. Random-walk method.** The random-walk method was devised in an attempt to unite the flexibility of the torus method with the guaranteed uniform distribution of the at-random method. We describe here two methods, the plain random walk and the smoother random walk.

A. *The plain random walk.* Let  $\mu$  denote a measure on  $SO(p)$  satisfying the following condition:

*Condition D.* The support of  $\mu$  (i.e., the complement of the union of all open  $\mu$ -null sets) generates a dense subgroup of  $SO(p)$ . Then we obtain a sequence of orthogonal matrices  $Q_K \in SO(p)$  as follows:

1. Set  $Q_0 = I_p$ , the identity matrix.
2. For  $K = 1, 2, \dots$  we let  $g_1, g_2, \dots$  be selected i.i.d., according to the law  $\mu$ , from  $SO(p)$ .

3. For  $K = 1, 2, \dots$  we set  $Q_K = g_K \circ Q_{K-1}$ .

To now obtain our 2-frames and 2-planes, we proceed as usual.

4.  $(\mathbf{v}_K, \mathbf{w}_K) = \pi(Q_K) = (Q_K \mathbf{e}_1, Q_K \mathbf{e}_2)$ .

5.  $P_K = \rho(\mathbf{v}_K, \mathbf{w}_K) = \langle Q_K \mathbf{e}_1, Q_K \mathbf{e}_2 \rangle$ . ( $\mathbf{e}_i$  is the  $i$ th canonical basis vector in  $R^p$ .)

*Remarks.* R1. As a concrete example of an appropriate measure  $\mu$ , we take a discrete  $\mu$  concentrated on the finite set of rotations  $\text{supp}(\mu) = \{R_{ij}(\lambda_{ij}) | 1 \leq i < j \leq p\}$ , where  $\{\lambda_{ij} | 1 \leq i < j \leq p\} \cup \{1\}$  is a set of real numbers linearly independent over the integers. We simply set

$$\mu(R_{ij}(\lambda_{ij})) = \frac{2}{p^2 - p}$$

for all  $i, j$  with  $1 \leq i < j \leq p$ . We shall denote  $\mu$  by  $U\{R_{ij}(\lambda_{ij})\}$ . By our discussion of the torus method, it is easy to see that  $\text{supp}(\mu)$  generates a dense subgroup of  $SO(p)$ . Thus, Condition D is satisfied by  $\mu$ .

R2. As long as  $p \geq 2$ , Condition D guarantees that the distribution of  $Q_K$  (the position of the random walk at time  $K$ ) approaches the invariant distribution on  $SO(p)$ . Precisely,

$$\lim_{n \rightarrow \infty} \mu^{*n} = \text{invariant measure}$$

where  $\mu^{*n}$  denotes the  $n$ th convolution power of  $\mu$  with itself, and the limit is understood in the sense of weak convergence [MAGL]. (Note: the invariant measure on  $SO(p)$  is what is sometimes referred to as the Haar measure.)

R3. The random walk achieves its flexibility through the available choice of measures  $\mu$  satisfying Condition D. By using such a measure with  $\text{supp}(\mu)$  lying close to  $I_p$ , we may maintain a slow rate of change in the sequences of rotations, 2-frames, and 2-planes, and thus a high degree of continuity.

R4. The use of a measure  $U\{R_{ij}(\lambda_{ij})\}$ , as described in R1 above, has the following drawback. Regardless of the choice of parameters  $\lambda_{ij}$ ,  $1 \leq i < j \leq p$ , the resulting random walk will be as unstraight as can be. Thus, the human viewer may experience disorientation in attempting to follow the resulting sequence of scatterplots. To remedy this, we hereby propose the use of the following type of measure  $\mu$ .

B. *The smoother random walk.* For convenience, we first introduce the *size* of an orthogonal matrix  $M \in SO(p)$  as follows:

$$\text{size}(M) = \max_{\mathbf{v} \neq \mathbf{0}} \{\text{angle}(\mathbf{v}, M\mathbf{v})\}$$

(where angle is always chosen to lie between  $0^\circ$  and  $180^\circ$ ). Now pick any orthogonal

matrix  $Q$  having size  $(Q) = \varepsilon$  where  $\varepsilon$  is small. Also, pick a set of numbers  $\lambda_{ij}$ ,  $1 \leq i < j \leq p$ , such that  $\{\lambda_{ij} | 1 \leq i < j \leq p\} \cup \{1\}$  is linearly independent over the integers. Also, have the  $\lambda_{ij}$  satisfy

$$\delta \leq \lambda_{ij} \leq 2\delta, \quad 1 \leq i < j \leq p$$

for some  $\delta > 0$  satisfying  $\delta < \varepsilon$  ( $\delta = \varepsilon^2$  seems to work well). Finally, we define  $\mu$  to be

$$\mu(Q \circ R_{ij}(\lambda_{ij})) = \frac{2}{p^2 - p}, \quad 1 \leq i < j \leq p$$

on  $\text{supp}(\mu) = \{Q \circ R_{ij}(\lambda_{ij}) | 1 \leq i < j \leq p\}$ . We denote  $\mu$  by  $\cup\{Q \circ R_{ij}(\lambda_{ij})\}$ .

It is easy to verify that this  $\mu$  satisfies Condition D above and thus, as long as  $p \geq 2$ ,  $\mu^{*n} \rightarrow$  invariant measure on  $SO(p)$  as  $n \rightarrow \infty$ . The smaller the choice of  $\varepsilon$ , the smoother the grand tour will turn out to be.

**4. Testing of grand tours.** In order to assess the suitability of a grand tour for a specific application, we need to perform statistical tests on it. Two characteristics of particular concern to us are the *rapidity* with which a sequence of 2-planes becomes dense, and the asymptotic *uniformity* of the limiting distribution, if any. For each of these characteristics, there is a multitude of possible choices of how to measure them. We have chosen one test that we feel measures well the most important characteristic.

*Rapidity.* Here we rely on the following:

**FACT.** *Given two 2-planes  $P, Q \in G_{2,p}$ , the relative position of  $P$  and  $Q$  in  $R^p$  is described by two angles  $\theta_1, \theta_2$  with  $0 \leq \theta_1 \leq \theta_2 \leq \pi/2$ . Precisely, there exists a rotation  $M \in SO(p)$  such that  $M(P) = \langle \mathbf{e}_1, \mathbf{e}_2 \rangle$  and  $M(Q) = \langle \cos \theta_1 \mathbf{e}_1 + \sin \theta_1 \mathbf{e}_3, \cos \theta_2 \mathbf{e}_2 + \sin \theta_2 \mathbf{e}_4 \rangle$ .*

The cosines of  $\theta_1$  and  $\theta_2$  are the correlations encountered in canonical correlation analysis. Thus, we use the terminology “canonical angles” for  $\theta_1$  and  $\theta_2$ .

We define the distance between  $P$  and  $Q$  as the larger canonical angle:  $d(P, Q) = \theta_2$ . (It may happen that  $\theta_1 = \theta_2$ .)

Now let  $S = \{P_1, \dots, P_n\}$  be a finite set of planes. Then we define the gap of  $S$  via

$$\text{gap}(S) = \max_{P \in G_{2,p}} \min_{1 \leq i \leq n} \{d(P, P_i)\}.$$

(We are justified in using “min” and “max” rather than “inf” and “sup” since  $n$  is finite and  $G_{2,p}$  is compact.) Let us define the  $\varepsilon$ -neighborhood of a plane  $P_0$  to be

**DEFINITION.**

$$N_\varepsilon(P_0) = \{P \in G_{2,p} | d(P, P_0) < \varepsilon\}.$$

The number  $\varepsilon$  will be called the *radius* of  $N_\varepsilon(P_0)$ . In terms of this definition, it is clear that  $\text{gap}(S)$  is the radius of the largest neighborhood in  $G_{2,p}$  which lies in the complement of the set  $S$  of planes:

$$\text{gap}(S) = \sup \{ \varepsilon > 0 | \exists P_0 \in G_{2,p} \ni N_\varepsilon(P_0) \subset G_{2,p} - S \}.$$

Or expressed yet another way,

$$\text{gap}(S) = \inf \left\{ \varepsilon > 0 \left| G_{2,p} = \bigcup_{i=1}^n N_\varepsilon(P_i) \right. \right\}.$$

We now use this last equation to establish lower bounds for  $n = n(\varepsilon)$ , where  $n(\varepsilon)$  is the smallest number of planes needed to have  $\text{gap} \leq \varepsilon$ . Namely,

$$G_{2,p} = \bigcup_{i=1}^n N_\varepsilon(P_i) \quad (\text{except for a set of measure } 0)$$

and so

$$\text{vol}(G_{2,p}) \leq \sum_{i=1}^n \text{vol}(N_\varepsilon(P_i))$$

where “vol” stands for the invariant (Haar) measure on  $G_{2,p}$ . By invariance,  $\text{vol}(N_\varepsilon(P_i))$  is independent of  $i$ , so

$$\text{vol}(G_{2,p}) \leq n \cdot \text{vol}(N_\varepsilon(P_0))$$

where  $P_0$  is, let us say,  $\langle \mathbf{e}_1, \mathbf{e}_2 \rangle$ .

Thus

$$n = n(\varepsilon) \geq \frac{\text{vol}(G_{2,p})}{\text{vol}(N_\varepsilon(P_0))} \quad \text{or} \quad n(\varepsilon) \geq [\text{Prob}(d(P, P_0) < \varepsilon)]^{-1},$$

where  $P \in G_{2,p}$  is distributed according to the invariant measure. It can be shown [Hote] that the canonical angles  $\theta_1, \theta_2$  between  $P$  and  $P_0$  have joint density function given by the following:

$$f(\theta_1, \theta_2) = \begin{cases} (p-2)(p-3)(\sin \theta_1 \cdot \sin \theta_2)^{p-4}(\sin^2 \theta_2 - \sin^2 \theta_1), & 0 \leq \theta_1 \leq \theta_2 \leq \pi/2, \\ 0 & \text{otherwise.} \end{cases}$$

If  $p = 3$ , we have  $\theta_1 = 0$  always, and  $\theta_2$  has density given by  $g(\theta_2) = \sin \theta_2, 0 \leq \theta_2 \leq \pi/2$ , and  $g(\theta_2) = 0$  otherwise.

We shall use the terminology “the 2-planes  $P$  and  $Q$  lie within angle  $\text{Ang}$ ” to mean that the larger canonical angle  $d(P, Q) = \theta_2$  is less than  $\text{Ang}$  (where  $0 \leq \text{Ang} \leq \pi/2$ ).

In the tables in Table 1, obtained by Monte Carlo methods, the probability shown is the fraction of random pairs of 2-planes in Euclidean space of the given dimension which lie within angle  $\text{Ang}$ . The column labeled “No. of planes” gives a theoretical lower bound for the number of 2-planes which can be chosen in that Euclidean space so that all 2-planes lie within angle  $\text{Ang}$  of one of the chosen ones. Namely, that theoretical lower bound is the quantity: greatest integer in  $1/\text{Prob}(d(P, Q) < \text{Ang})$ .

These tables should be thought of as a standard against which to measure the rapidity with which a sequence of planes becomes dense. In fact, if we set  $N_{\text{poss}}(\varepsilon)$  = the smallest possible number of planes needed to achieve a gap of  $\varepsilon$ , and  $N_{\text{gt}}(\varepsilon)$  = the smallest number of planes (in sequence), from some particular choice of grand tour, needed to achieve a gap of  $\varepsilon$ , we have

$$n(\varepsilon) \leq N_{\text{poss}}(\varepsilon) \leq N_{\text{gt}}(\varepsilon)$$

for all  $\varepsilon > 0$ . These inequalities are, with very few exceptions, actually strict ones.

Figures 1–6 display the gap as a function of the number of planes, for three types of grand tour: (1) planes picked at random, (2) planes picked by the torus method, and (3) planes picked via plain random walk on  $SO(p)$ . The gap was not, in fact, computed but was instead estimated via  $\text{gap}(N) \approx \max_{1 \leq i \leq 100} \min_{1 \leq j \leq N} d(Q_i, P_j)$  where  $\{Q_i\}$  is a fixed set of planes picked at random. Due to the vast quantity of computing time necessary, we have restricted the calculations to only two values of the dimension:  $p = 4$  (using the average of 5 repetitions) and  $p = 8$  (using the average of 3 repetitions).

TABLE 1

Dimension 3			Dimension 4		
Ang	Probability	No. of planes	Ang	Probability	No. of planes
5	.38E-02	263	5	.19E-04	51684
10	.15E-01	66	10	.31E-03	3258
15	.34E-01	30	15	.15E-02	650
20	.60E-01	17	20	.48E-02	209
25	.94E-01	11	25	.12E-01	87
30	.13	8	30	.23E-01	43
35	.18	6	35	.42E-01	24
40	.23	5	40	.69E-01	15
45	.29	4	45	.11	10
50	.36	3	50	.16	7
55	.43	3	55	.22	5
60	.50	3	60	.30	4
65	.58	2	65	.39	3
70	.66	2	70	.49	3
75	.74	2	75	.61	2
80	.83	2	80	.73	2
85	.91	2	85	.86	2
90	1.0	2	90	1.0	2

Dimension 5			Dimension 6		
Ang	Probability	No. of planes	Ang	Probability	No. of planes
5	.11E-06	0.9E+07	5	.67E-09	0.2E+10
10	.70E-05	143529	10	.17E-06	0.6E+07
15	.78E-04	12805	15	.42E-05	239048
20	.43E-03	2349	20	.40E-04	24980
25	.16E-02	636	25	.23E-03	4407
30	.45E-02	222	30	.91E-03	1096
35	.11E-01	93	35	.29E-02	345
40	.23E-01	45	40	.77E-02	130
45	.43E-01	24	45	.18E-01	56
50	.75E-01	14	50	.37E-01	27
55	.12	9	55	.70E-01	15
60	.19	6	60	.12	9
65	.27	4	65	.20	6
70	.38	3	70	.30	4
75	.51	2	75	.44	3
80	.66	2	80	.61	2
85	.83	2	85	.80	2
90	1.0	2	90	1.0	2

Dimension 7			Dimension 8		
Ang	Probability	No. of planes	Ang	Probability	No. of planes
5	.42E-11	0.2E+12	5	.27E-13	0.4E+14
10	.42E-08	0.2E+09	10	.11E-09	0.9E+10
15	.23E-06	0.4E+07	15	.13E-07	0.7E+08
20	.39E-05	254912	20	.39E-06	0.3E+07
25	.34E-04	29331	25	.52E-05	190626
30	.19E-03	5185	30	.42E-04	23917
35	.81E-03	1230	35	.23E-03	4305
40	.27E-02	366	40	.99E-03	1011
45	.77E-02	130	45	.34E-02	293
50	.19E-01	53	50	.99E-02	102
55	.42E-01	25	55	.25E-01	41
60	.82E-01	13	60	.56E-01	18
65	.15	7	65	.11	9
70	.25	5	70	.20	5
75	.38	3	75	.34	3
80	.56	2	80	.52	2
85	.77	2	85	.75	2
90	1.0	2	90	1.0	2



TABLE 1 (cont.)

Dimension 9			Dimension 10		
Ang	Probability	No. of planes	Ang	Probability	No. of planes
5	.18E-15	0.6E+16	5	.12E-17	0.8E+18
10	.29E-11	0.3E+12	10	.77E-13	0.1E+14
15	.79E-09	0.1E+10	15	.47E-10	0.2E+11
20	.40E-07	0.2E+08	20	.42E-08	0.2E+09
25	.82E-06	0.1E+07	25	.13E-06	0.8E+07
30	.92E-05	108458	30	.21E-05	486134
35	.68E-04	14768	35	.20E-04	50072
40	.36E-03	2755	40	.14E-03	7398
45	.15E-02	657	45	.69E-02	1453
50	.52E-02	192	50	.28E-02	359
55	.15E-01	67	55	.93E-02	108
60	.38E-01	27	60	.26E-01	39
65	.84E-01	12	65	.64E-01	16
70	.17	6	70	.14	8
75	.30	4	75	.27	4
80	.49	3	80	.46	3
85	.73	2	85	.71	2
90	1.0	2	90	1.0	2

Dimension 12			Dimension 14		
Ang	Probability	No. of planes	Ang	Probability	No. of planes
5	.58E-22	0.2E+23	5	.28E-26	0.4E+27
10	.57E-16	0.2E+17	10	.44E-19	0.2E+20
15	.17E-12	0.6E+13	15	.65E-15	0.2E+16
20	.47E-10	0.2E+11	20	.55E-12	0.2E+13
25	.34E-08	0.3E+09	25	.93E-10	0.1E+11
30	.11E-06	0.9E+07	30	.56E-08	0.2E+09
35	.18E-05	558417	35	.17E-06	0.6E+07
40	.19E-04	52000	40	.28E-05	355788
45	.14E-03	6921	45	.31E-04	32130
50	.81E-03	1229	50	.24E-03	4121
55	.36E-02	279	55	.14E-02	708
60	.13E-01	78	60	.64E-02	157
65	.38E-01	27	65	.23E-01	44
70	.97E-01	11	70	.68E-01	15
75	.21	5	75	.17	6
80	.40	3	80	.36	3
85	.67	2	85	.64	2
90	1.0	2	90	1.0	2

Dimension 16			Dimension 20		
Ang	Probability	No. of planes	Ang	Probability	No. of planes
5	.14E-30	0.7E+31	5	.36E-39	0.3E+40
10	.34E-22	0.3E+23	10	.22E-28	0.5E+29
15	.25E-17	0.4E+18	15	.40E-22	0.3E+23
20	.65E-14	0.2E+15	20	.95E-18	0.1E+19
25	.26E-11	0.4E+12	25	.21E-14	0.5E+15
30	.31E-09	0.3E+10	30	.95E-12	0.1E+13
35	.16E-07	0.6E+08	35	.15E-09	0.7E+10
40	.42E-06	0.2E+07	40	.98E-08	0.1E+09
45	.68E-05	146196	45	.34E-06	0.3E+07
50	.74E-04	13516	50	.71E-05	140243
55	.57E-03	1767	55	.94E-04	10604
60	.32E-02	311	60	.85E-03	1180
65	.14E-01	71	65	.54E-02	184
70	.49E-01	21	70	.26E-01	39
75	.14	8	75	.92E-01	11
80	.32	4	80	.26	4
85	.61	2	85	.57	2
90	1.0	2	90	1.0	2

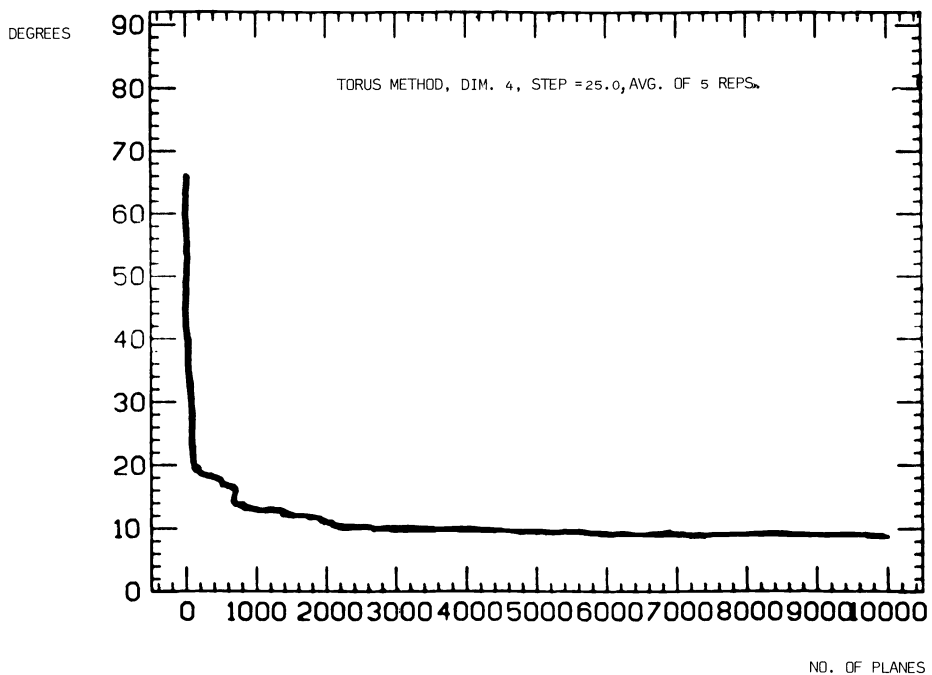


FIG. 1.

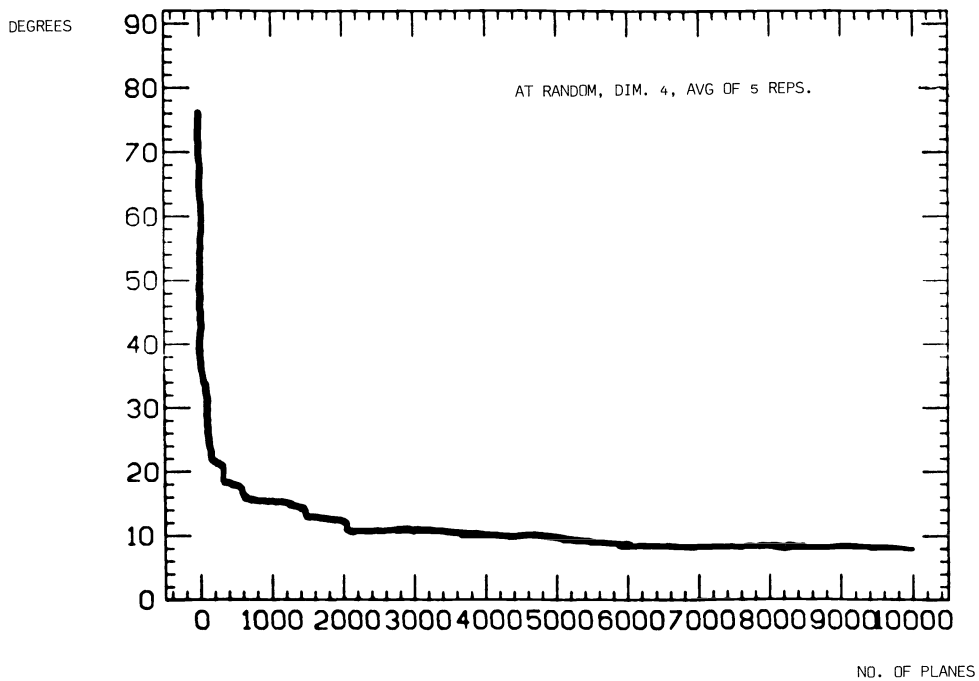


FIG. 2.

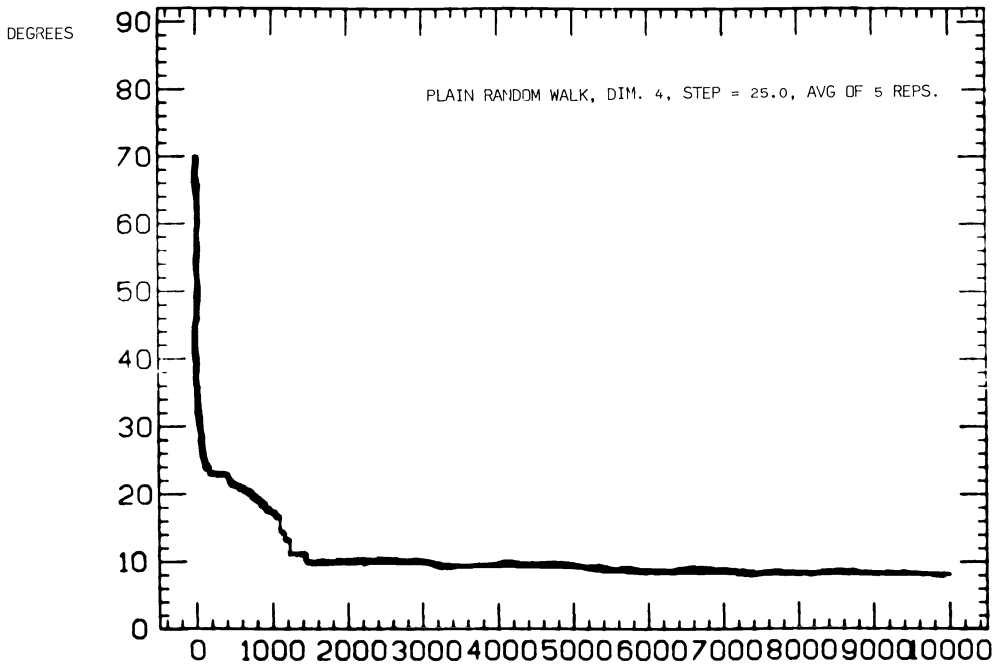


FIG. 3.

NO. OF PLANES

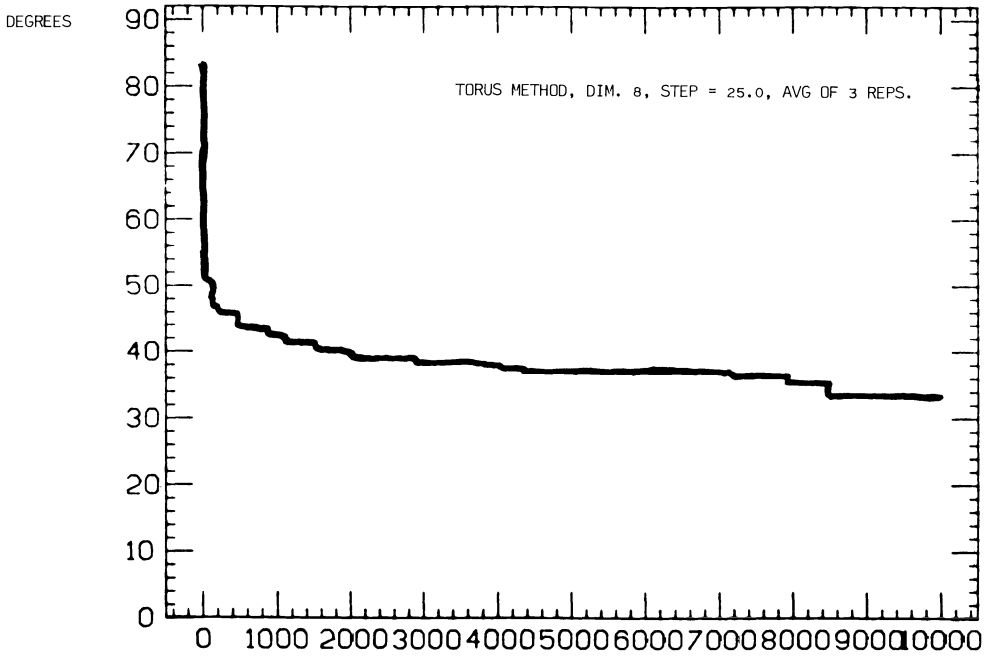


FIG. 4.

NO. OF PLANES

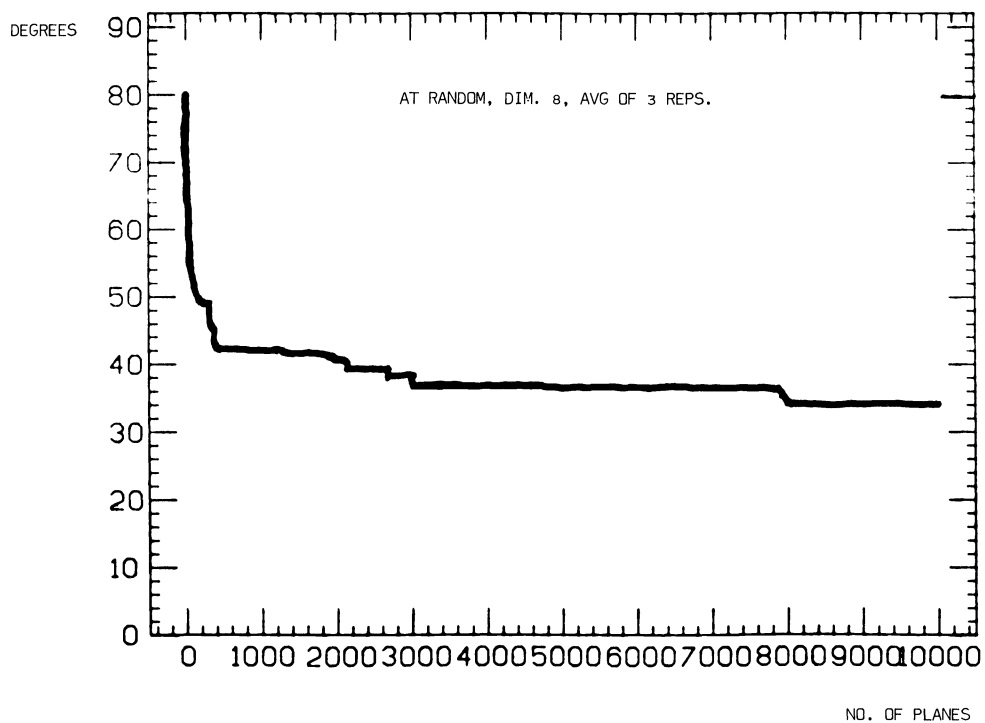


FIG. 5.

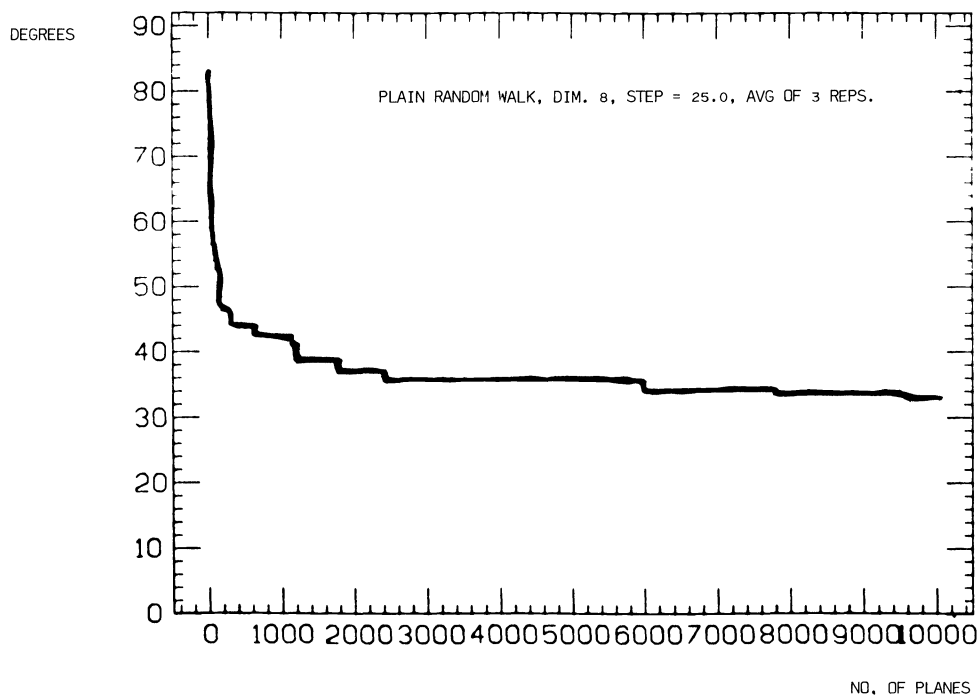


FIG. 6.

**5. Some applications of grand tours.** The most basic purpose to which one may put a grand tour is to try to understand the shape of data. This understanding will presumably be applied to interpreting the data—drawing real-world conclusions.

Unfortunately, we are a long way from the point where we can do this confidently. The grand tour can be said to approximate the information content of a  $p$ -dimensional scatterplot by a time-indexed family of two-dimensional images, i.e., a movie. In order that human observers be able to interpret this kind of movie visually, a great deal of experience viewing such movies would be advantageous.

Much is still to be learned when  $p = 3$ , and the case  $p = 4$  already presents a major challenge. Perhaps it would be of value to develop a *taxonomy of scatterplots* based on extensive experience with actual data. This may lead to the use of certain adjectives to describe the shapes of scatter-diagrams in greater than two dimensions. These adjectives would ideally correspond to measurements which the computer could make with great speed. An example of one such adjective-measurement pair might be the idea of “clottedness” as defined in Friedman–Tukey [PP] as their figure of merit for projection pursuit.

A useful genre of statistics may be compiled by applying a uniformly distributed grand tour to a particular scatterplot  $S$  in  $R^p$ . Let  $\psi$  be any measurement that can be applied to two-dimensional scatterplots, such as their clottedness. Then, for each 2-plane  $Q$  in  $R^p$ , we may apply  $\psi$  to the result of projecting  $S$  onto  $Q$ , obtaining  $\psi(\pi_Q(S))$ . As  $Q$  ranges over all 2-planes in  $R^p$  (with the invariant measure), there is a measure induced on the set of real numbers  $\{\psi(\pi_Q(S))\}$ . This measure carries significance especially when all coordinates represent identical units.

Statistics of this distribution of real numbers may be estimated by letting  $Q$  run through a long sequence  $P_1, \dots, P_N$  of a uniformly distributed grand tour. To take, for example, the *mean*  $m$  of this distribution, we may estimate  $m$  via

$$\hat{m} = \frac{1}{N} \sum_{i=1}^N \psi(\pi_i(S)),$$

where  $\pi_i$  denotes orthogonal projection onto the 2-plane  $P_i$ . This is a deep fact, provable by standard techniques in ergodic theory [Brei].

The advantages of using such measurements (and their corresponding adjectives) include 1) they are easy to compute, and 2) they convey an intuitive content based on the user’s knowledge of two-dimensional scatterplots.

Projection pursuit methods can be described as the study of the above paradigm where the *maximum* or *minimum* of the set  $\{\psi(\pi_Q(S))\}$  is the statistic of interest. These extreme values are usually sought via hill-climbing algorithms as in [PP].

One great mystery in projection pursuit is endemic to hill-climbing algorithms: how can we be confident that a local maximum is in fact the absolute maximum (or at least very near to it)? A grand tour which rapidly becomes dense in  $G_{2,p}$  may be used to help with this problem. Using, e.g., the torus method with large stepsize (step = 25.0 will work), we may let each grand tour plane  $P_1, P_2, \dots$  be the starting point for a hill-climbing procedure to maximize  $\psi(\pi_Q(S))$  locally. When the local max is found, a record is kept of that value  $\max_i$ . One may then determine from the distribution of  $\{\max_i\}$  an estimate of the absolute max.

Perhaps a better procedure would be to use the torus method with an intermediate stepsize, say step = 1.0. Then hill-climbing may be initiated from  $P_i$  whenever

$$\psi(\pi_{i-1}(S)) < \psi(\pi_i(S)) > \psi(\pi_{i+1}(S))$$

(where  $\pi_j$  again denotes orthogonal projection onto  $P_j$ ). Once again the local max values  $\{\max_i\}$  may be stored and eventually used to estimate the absolute max.

Several films demonstrating the SLAC implementation of the grand tour have been created by the author and in collaboration with A. Buja [AsiBu].

**Appendix.** We describe here a method for reducing the number of matrix multiplications in the grand tour, torus method to  $2p-3$ .

1. Let  $N = 2p-3$  and think of the coordinates  $x_{ij}$  of  $T^N$  as being indexed by all pairs  $i, j$  where  $i = 1$  or  $2$ , and  $2 \leq j \leq p$  if  $i = 1$ , but  $3 \leq j \leq p$  if  $i = 2$ .

2. Define a map  $f: T^N \rightarrow SO(p)$  via  $f(x_{1,2}, \dots, x_{2,p}) = R_{1,2}(x_{1,2}) \circ \dots \circ R_{2,p}(x_{2,p})$ .

3. Define a map  $F: T^N \rightarrow G_{2,p}$  via  $F = \rho \circ \pi \circ f$  where  $\pi: SO(p) \rightarrow V_{2,p}$  and  $\rho: V_{2,p} \rightarrow G_{2,p}$  are as in § 3, part I above.

4. We shall prove the

**THEOREM.**  $F: T^N \rightarrow G_{2,p}$  is a surjection.

*Proof.* Let  $P \in G_{2,p}$  be an arbitrary 2-plane. We must find  $x_{1,2}, \dots, x_{2,p}$  (real numbers mod  $2\pi$ ) such that  $R_{1,2}(x_{1,2}) \circ \dots \circ R_{2,p}(x_{2,p}) \langle \mathbf{e}_1, \mathbf{e}_2 \rangle = P$ . Letting  $\theta_{i,j} = -x_{i,j}$  this is equivalent to finding  $\theta_{i,j}$  (real numbers mod  $2\pi$ ) such that

$$R_{2,p}(\theta_{2,p}) \circ \dots \circ R_{1,2}(\theta_{1,2}) P = \langle \mathbf{e}_1, \mathbf{e}_2 \rangle.$$

Now pick any orthonormal basis  $\mathbf{v}, \mathbf{w}$  for  $P$ . First, we pick  $\theta_{1,2}$  so as to satisfy

$$-\sin \theta_{1,2} v_1 + \cos \theta_{1,2} v_2 = 0.$$

This assures that  $R_{1,2}(\theta_{1,2})$  annihilates the second component of  $v$ .

We similarly choose  $\theta_{1,j}$  so that

$$-\sin \theta_{1,j} u'_1 + \cos \theta_{1,j} v'_j$$

where  $v'_1 =$  the first component of

$$R_{1,j-1}(\theta_{1,j-1}) \circ \dots \circ R_{1,2}(\theta_{1,2}) \mathbf{v}$$

and  $v'_j$  is the  $j$ th component (in fact,  $v'_j = v_j$  for  $j \geq 3$ ). Thus,  $R_{1,p}(\theta_{1,p}) \circ \dots \circ R_{1,2}(\theta_{1,2}) \mathbf{v}$  must have its 2nd through  $p$ th components equal to 0, and since it is a unit vector, it must, in fact, be  $\mathbf{e}_1$ .

We now similarly choose  $\theta_{2,3}, \dots, \theta_{2,p}$  so the 3rd through  $p$ th components of  $\mathbf{w}'$  are annihilated by  $R_{2,3}(\theta_{2,3}), \dots, R_{2,p}(\theta_{2,p})$  in turn, where  $\mathbf{w}'$  denotes  $R_{1,p}(\theta_{1,p}) \circ \dots \circ R_{1,2}(\theta_{1,2}) \mathbf{w}$ . As a result, the vector  $R_{2,p}(\theta_{2,p}) \circ \dots \circ R_{1,2}(\theta_{1,2}) \mathbf{w}$  lies in the plane  $\langle \mathbf{e}_1, \mathbf{e}_2 \rangle$ . But, since the orthogonal pair  $\mathbf{v}, \mathbf{w}$  is taken to an orthonormal pair by the orthogonal transformation  $R_{2,p}(\theta_{2,p}) \circ \dots \circ R_{1,2}(\theta_{1,2})$  and since  $\mathbf{v}$  is taken to  $\mathbf{e}_1$ , we must have that  $\mathbf{w}$  is taken to  $\mathbf{e}_2$ . Thus, we have chosen  $\theta_{1,2}, \dots, \theta_{2,p}$  so that  $R_{2,p}(\theta_{2,p}) \circ \dots \circ R_{1,2}(\theta_{1,2})$  takes the frame  $\mathbf{v}, \mathbf{w}$  to the frame  $\mathbf{e}_1, \mathbf{e}_2$  (more than we needed!). As a result the plane  $P = \langle \mathbf{v}, \mathbf{w} \rangle$  is taken to  $\langle \mathbf{e}_1, \mathbf{e}_2 \rangle$  as desired.

5. Since  $F: T^N \rightarrow G_{2,p}$  is a surjection, it now follows, just as in § 3, part I, that a dense curve  $\alpha$  in  $T^N$  will be taken by  $F$  to a dense curve  $F \circ \alpha$  in  $G_{2,p}$ . We then just let the  $K$ th plane of our grand tour be defined as

$$P_K = F(\alpha(K \cdot \text{STEP})), \quad K = 1, 2 \dots$$

for some appropriate choice of stepsize STEP.

**Acknowledgments.** The author would like to thank Persi Diaconis, Jerry Friedman, Peter J. Huber, Ingram Olkin, Mathis Thoma, and above all, Andreas Buja, for valuable conversations. He is also extremely grateful to Harriet Canfield for the lengthy task of typing this paper.

## REFERENCES

- [Andr] D. F. ANDREWS, *Plots of high-dimensional data*, Biometrics, 28 (1972), pp. 125–136.
- [AsiBu] D. ASIMOV AND A. BUJA, *Finding structure in unstructured data* (a short film), Computation Research Group, SLAC, 1983.
- [Brei] L. BREIMAN, *Probability*, Addison-Wesley, Reading, MA, 1968.
- [Chev] C. CHEVALLEY, *Theory of Lie Groups*, Princeton Univ. Press, Princeton, NJ, 1946.
- [FFT] J. FRIEDMAN, M. A. FISHERKELLER AND J. TUKEY, PRIM-9: *An interactive multidimensional data display and analysis system*, Proc. Fourth International Congress for Stereology, 1974.
- [Hotel] H. HOTELLING, *Relations between two sets of variates*, Biometrika, 28 (1936), pp. 321–377.
- [HWTN] G. H. HARDY AND E. M. WRIGHT, *Theory of Numbers*, Clarendon Press, Oxford, p. 381ff.
- [MAGL] Y. GUIVARC'H, M. KEANE AND B. ROYNETTE, *Marches aléatoires sur les groupes de Lie*, Springer-Verlag, New York, 1977.
- [MCTP] F. JAMES, *Monte Carlo Theory and Practice*, CERN, 1980, p. 38.
- [MMCM] V. ARNOLD, *Mathematical Methods of Classical Mechanics*, Springer-Verlag, New York, 1978, pp. 148ff.
- [PP] J. FRIEDMAN AND J. TUKEY, *A projection pursuit algorithm for exploratory data analysis*, IEEE Trans. Comp., C-23 (1974), pp. 881–890.
- [TT77] J. TUKEY AND P. TUKEY, *Methods for direct and indirect graphic display for data sets in three and more dimensions*, Bell Laboratories, Murray Hill, NJ, 1977.

## FAST HIGH ORDER ACCURATE SOLUTION OF LAPLACE'S EQUATION ON IRREGULAR REGIONS\*

ANITA MAYO†

**Abstract.** Fast methods are developed for the numerical solution of Laplace's equation on irregular regions with smooth boundaries. The methods use an integral equation formulation in a small carefully chosen region near the boundary, and use fast solvers to extend the solution to the rest of the region. Our experiments show the method works well for problems on interior, multiconnected and exterior regions. Moreover, we have found the computation of fourth order accurate solutions to be only slightly more expensive than those of second order. In addition, since we use integral equations we find that when we compute a harmonic function its conjugate can be computed at small additional cost.

**Key words.** fast, high order accurate solver, Poisson's equation, integral equations, irregular regions

**1. Introduction.** In this paper we develop an efficient and high order accurate numerical method for solving Laplace's equation on general regions with smooth boundaries. In the past fifteen years many methods have been developed for solving the linear systems of equations that arise from discretizing separable elliptic equations on regular regions, that is on regions that are rectangular with respect to some coordinates. Specifically, methods are available for solving Poisson's equation which use techniques such as the fast Fourier transform and odd-even reduction (see Hockney [8], Buneman [3]). There are others which use Fourier Toeplitz factorization (Fisher et al. [7]). These methods are fast and require relatively low storage. For example, one can obtain a second order accurate solution to Poisson's equation on a square with  $n$  equally spaced points in each direction using only  $Cn^2 \log_2 n$  operations and  $n^2 + Cn$  storage locations.

One means by which these techniques can be applied to irregular regions is through the use of capacitance matrix methods. (See [4], [12], [13].) In this paper we develop a different method for solving Laplace's equation on such regions. It relies on the use of an integral equation formulation of the problem and is meant to be used when the solution is required at a large number of mesh points.

Because we use integral equation formulations our method has the advantage of being particularly well suited to problems on exterior regions. It is also sufficiently flexible to allow the use of extra data points on the boundary curve mesh where the boundary data is more oscillatory or where the boundary has high curvature.

We note that by using an appropriate integral equation formulation and the proper quadrature formula the solution of Laplace's equation can be computed surprisingly accurately at mesh points which aren't too close to the boundary of the given region. We use Fredholm integral equations of the second kind, and have found them relatively inexpensive to solve accurately. This is primarily because we use a Nystrom method with the trapezoid rule as the quadrature formula to discretize the integral equation, and because the trapezoid rule is so accurate on periodic regions. Since the accuracy of the solution of the integral equation is the same as the accuracy of the quadrature formula, we need very few mesh points on the boundary curve. However, it would still be quite expensive to compute the solution at many points by evaluating an integral. Instead, we imbed the irregular region in a regular region and only compute the solution by evaluating an integral at a small number of carefully chosen mesh points

---

\* Received by the editors June 4, 1981, and in final revised form July 29, 1983.

† Department of Mathematics, University of California at Berkeley, Berkeley, California 94720.



inside the irregular region. Then a fast Poisson solver is used to extend the solution to the rest of the irregular region.

Since we use integral equations which can be solved so accurately, we have found that other calculations such as computing the conjugate of a harmonic function or computing a fourth order accurate solution can be done at small additional cost. By using an appropriate integral equation formulation we have also been able to solve a modified Dirichlet problem. This is a Dirichlet problem on a nonsimply connected region where the solution is required to be the real part of single valued analytic function. In this case the boundary values can be specified only up to an additive constant on all but one of the boundary components. We also note that the methods were all easily programmed. Moreover, additional problems on the same region are less expensive to solve than the original calculation. This is because we solve the matrix equation that arises in the solution of the integral equation by Gaussian elimination. Since the matrix depends only on the geometry of the region, the same  $LU$  decomposition of the matrix can be used for subsequent calculations. Finally, we mention that although the following have not yet been tested, this method can be easily used to compute the solution on only a part of the region, to compute a higher order accurate solution on just part of the region, and to compute derivatives of the solution directly.

**2. The extension of the solution from mesh points near the boundary.** First we show how fast solvers can be used to extend the solution of an elliptic equation from mesh points near the boundary of a region to those in the interior.

Suppose  $Lu=0$  is a linear elliptic equation. Let  $L_h$  be an  $m$ th order accurate discrete approximation to  $L$  which has a fast  $m$ th order accurate solver  $L_h^{-1}$  on some regular region  $R$ , and let  $D$  be a set of mesh points contained in  $R$ .

Let  $S_{ij}$  be the set of mesh points used in forming  $(L_h u)_{ij}$  at the point  $(i, j) = (x_i, y_i)$ . For example,  $S_{ij}$  would be the point  $(i, j)$  and its four nearest neighbors if  $L_h = \Delta_h^5$ , the standard 5-point approximation to the Laplacian.

We define  $B = \{(i, j) \mid S_{ij} \cap D \neq \emptyset, \text{ and } S_{ij} \cap D^c \neq \emptyset\}$ . Thus  $B$  is the set of mesh points near the boundary of  $D$  where values from both inside and outside  $D$  are needed to form  $L_h$ . In the case of  $\Delta_h^5$ ,  $B$  would consist of 2 rings of points, one inside and one outside  $D$ . For higher order accurate approximations  $B$  would be larger.

Let  $u(x, y)$  be a solution of  $Lu=0$  on a region which includes all the mesh points of  $D$ . We define the mesh function

$$U_{ij} = \begin{cases} u(x_i, y_i), & (i, j) \in D, \\ 0, & (i, j) \in D^c. \end{cases}$$

Consider the set of points  $S = \cup S_{ij}$  where the union is over  $(i, j) \in B$ . Let  $\partial R$  be the set of mesh points at the boundary of  $R$ , and suppose for the moment that we can find  $U_{ij}$  for  $(i, j) \in S$  and  $(i, j) \in \partial R$ . In the part of  $S$  which is outside of  $D$  this is trivial since  $U_{ij} = 0$ . Assume that in  $S \cap D$  an approximation to  $U_{ij}$  can be found by using integral equation methods. We note that  $S \cap D$  contains a small number of mesh points so this is much less expensive than evaluating  $U_{ij}$  throughout  $D$  by integral equation methods. Once we know  $U_{ij}$  for all points of  $S$  we can compute  $(L_h U)_{ij}$  at all points of  $B$ . Then by using the fast solver  $L_h^{-1}$  we can find the mesh function  $v_{ij}$  such that

$$(L_h v)_{ij} = \begin{cases} (L_h U)_{ij}, & (i, j) \in B, \\ 0, & (i, j) \in R - B, \end{cases}$$

$$v_{ij} = 0, \quad (i, j) \in \partial R.$$

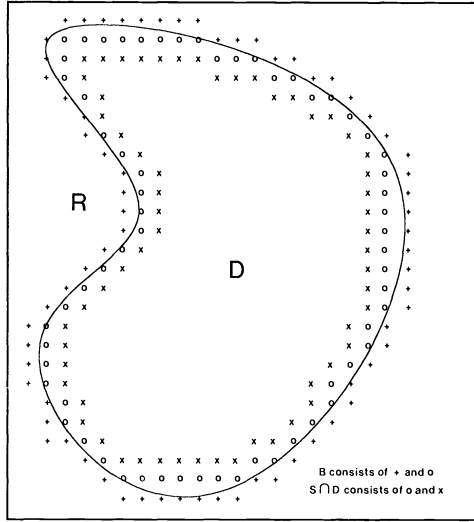


FIG. 1

It follows that  $v_{ij}$  satisfies the following equations:

$$(L_h v)_{ij} = 0 \quad \text{and} \quad (L_h U)_{ij} = O(h^m), \quad (i, j) \in D - B,$$

$$(L_h v)_{ij} = (L_h U)_{ij}, \quad (i, j) \in B,$$

and

$$(L_h v)_{ij} = (L_h U)_{ij} = 0, \quad (i, j) \in R - D - B,$$

$$v_{ij} = U_{ij}, \quad (i, j) \in \partial R.$$

Therefore  $(L_h U)_{ij} - (L_h v)_{ij} = O(h^m)$  at all points of  $R$ . Now  $L_h^{-1}$  is an  $m$ th order accurate solver, and consequently  $U_{ij} - v_{ij} = O(h^m)$ . It follows that  $\{v_{ij}\}$  will be an  $m$ th order accurate solution to  $Lu = 0$  on  $D$ .

**3. Second order accurate solution to Dirichlet problem.** We consider the problem  $\Delta u = 0$  on an irregular domain  $\Omega$  with boundary  $\partial\Omega = (x(s), y(s)) \in C^\delta(0, 1)$  on which  $u = f(s)$  is specified ( $s$  need not be arc length). We embed  $\Omega$  (or part of it when  $\Omega$  is an exterior region) in some larger domain which has a fast second order accurate Poisson solver. For example, we can let  $R$  be a square with a uniform mesh in each direction. We then extend the function  $u$  to be 0 on  $\Omega^c$  and discretize  $\partial\Omega$  with some other mesh of width  $k$ .

Suppose we let  $D$  be the set of mesh points in  $\Omega$ . Then by using an integral equation formulation we could try to compute  $u$  at points of  $S \cap D$ . The difficulty is that integral equation formulations are not well suited to standard numerical methods very close to the boundary for the following reason. The methods all require solving an integral equation on the boundary, and then computing an integral at points in the interior. The integral equations can be chosen to be Fredholm integral equations of the second kind, and effective methods are available for solving them, but the integrals have a kernel which becomes unbounded as the point at which the solution is to be evaluated approaches the boundary.

For example, we have used the formulation of the solution in terms of a double layer density function:

$$(3.1) \quad u(t) = \frac{1}{\pi} \int_{\partial\Omega} \frac{\partial \log r(s, t)}{\partial n_s} \mu(s) ds.$$

The integral equation for the density function  $\mu$  is

$$(3.2) \quad \mu(t) + \frac{1}{\pi} \int_{\partial\Omega} \frac{\partial \log r(s, t)}{\partial n_s} \mu(s) ds = 2f(s);$$

see [5 p. 229]. Here  $r(s, t)$  is the distance between the point  $(x(s), y(s))$  and the point  $(x(t), y(t))$ .

Since the interval of integration is periodic and since the kernel

$$\frac{\partial \log r(s, t)}{\partial n_s} = \frac{y'(s)(x(s) - x(t)) - x'(s)(y(s) - y(t))}{(x(s) - x(t))^2 + (y(s) - y(t))^2}$$

is bounded along the curve, (3.1) can be solved very accurately (see § 4). However, since the same kernel grows like  $1/r$  as the point  $t$  approaches the boundary point  $s$  from the interior, it is expensive to compute the integral (3.1) with any accuracy. The same type of difficulty occurs when other integral equation formulations are used.

Fortunately, this problem can be circumvented by choosing  $D$  to be smaller, so that the points of  $D \cap S$  are not too close to the boundary of  $\Omega$ . Let  $B1$  be the set of irregular points in  $\Omega$ , i.e., the set of points in  $\Omega$  which have neighboring mesh points outside  $\Omega$ . Let  $B2$  be the set of points in  $\Omega$  which have neighboring mesh points in  $B1$  but are not themselves in  $B1$ , and let  $B3$  be the set of points in  $\Omega$  which have neighboring mesh points in  $B2$ , but which are not in  $B1$  or  $B2$ . These points form three nested sets of points in  $\Omega$ . See Fig. 2.

Let  $D = \Omega \cap B1^c$ , that is, let  $D$  consist of all the mesh points in  $\Omega$  except those in  $B1$ . Since  $B$  consists of points in  $D$  with neighbors outside  $D$  as well as points exterior to  $D$  which have neighbors inside  $D$ , we have  $B = B1 \cup B2$ .

We note that our definitions do not imply that the region is always six mesh widths wide. In fact, if the region is narrow and the mesh is coarse, then points of  $B1$  need

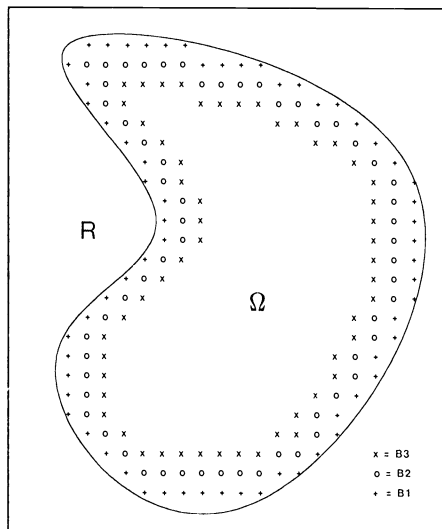


FIG. 2

not have any neighboring mesh points in  $B2$ , and points of  $B2$  need not have any neighboring mesh points in  $B3$ . However, no special programming need be done in these cases and the method is still effective.

Since they are at least at distance  $h$  away from  $\partial\Omega$ , we can compute the solution accurately at points of  $S \cap D = B2 \cup B3$ . (See § 4.) Therefore we can compute an accurate approximation to the discrete Laplacian  $\Delta_h^S$  at points of  $B$  and apply a fast Poisson solver on  $R$ . This method, of course, does not provide the potential at points of  $B1$ . However, since the potential is known on the boundary, once we obtain the solution at the other points in the interior we can use various types of interpolation procedures to obtain the solution there. In our experiments we used third order Lagrange polynomials.

This method can clearly be extended to other types of regions. We need only find a suitable integral equation formulation. For example, we next consider the case where the region  $\Omega$  is doubly connected and where the solution  $u$  is specified on the outer boundary curve  $L0$  and on the inner one  $L1$ . In this case we assume that the solution has the form

$$u(t) = \frac{1}{\pi} \int_{\partial\Omega} \mu(s) \frac{\partial \log r(s, t)}{\partial n_s} ds + A1 \log [(x(t) - x^*)^2 + (y(t) - y^*)^2],$$

where  $(x^*, y^*)$  is any point inside the region bounded by  $L1$ . The density function  $\mu$  is the solution of the integral equation

$$(3.3) \quad \mu(t) + \frac{1}{\pi} \mu(s) \int_{\partial\Omega} \frac{\partial \log r(s, t)}{\partial n_s} ds = 2u(t) + 2A1 \log [(x(t) - x^*)^2 + (y(t) - y^*)^2],$$

$t$  on  $\partial\Omega$ .

In order to find the solution we must determine the coefficient  $A1$  of the logarithm term as well as the density function  $\mu$ . To do this we first find the function  $\mu 1$  which satisfies the integral equation

$$(3.4) \quad \mu 1(t) + \frac{1}{\pi} \int_{\partial\Omega} \mu 1(s) \frac{\partial \log r(s, t)}{\partial n_s} ds = 2A1 \log [(x(t) - x^*)^2 + (y(t) - y^*)^2],$$

$t$  on  $L0$ ,

$$\mu 1(t) + \frac{1}{\pi} \int_{\partial\Omega} \mu 1(s) \left[ \frac{\partial \log r(s, t)}{\partial n_s} + 1 \right] ds = 2A1 \log [(x(t) - x^*)^2 + (y(t) - y^*)^2],$$

$t$  on  $L1$ ,

and then we find the function  $\mu 2$  which satisfies

$$(3.5) \quad \begin{aligned} \mu 2(t) + \frac{1}{\pi} \int_{\partial\Omega} \mu 2(s) \frac{\partial \log r(s, t)}{\partial n_s} ds &= 2u(t), & t \text{ on } L0, \\ \mu 2(t) + \frac{1}{\pi} \int_{\partial\Omega} \mu 2(s) \left[ \frac{\partial \log r(s, t)}{\partial n_s} + 1 \right] ds &= 2u(t), & t \text{ on } L1. \end{aligned}$$

The coefficient and density function can be computed in terms of  $\mu 1$  and  $\mu 2$ :

$$A1 = \int_{L1} \mu 1 ds / \int_{L1} \mu 2 ds,$$

and  $\mu = \mu 1 + A1\mu 1$ .

Both equations (3.4) and (3.5) are Fredholm integral equations of the second kind with bounded kernels. See [10], [11, p. 145–147]. Furthermore, the matrices obtained from discretizing the left-hand sides of (3.4) and (3.5) are the same. Therefore if the matrix equations that arise from using a Nystrom method are solved by Gaussian elimination then the same  $LU$  decomposition can be used for both calculations.

In some applications it is necessary to solve a modified Dirichlet problem, that is to find a function  $u$  that is harmonic on a multiconnected region if on each of the curves forming the boundary of the region  $u$  is prescribed up to an additive constant. More precisely, if  $\{L_k\}$  are the boundary curves, then  $u$  satisfies conditions of the form

$$u = f(t) + b_k \quad \text{on } L_k.$$

Only one of the constants, say  $b_0$ , may be specified arbitrarily. The others are determined by the condition that  $u$  be the real part of an analytic function which is single valued in the region.

In our experiments we considered an example where the region is doubly connected. In that case one assumes that  $u$  has the form

$$u(t) = \frac{1}{2\pi} \int_{\partial\Omega} \mu(s) \frac{\partial \log r(s, t)}{\partial n_s} ds.$$

It can be shown [11, p. 152] that  $\mu$  will be the solution of (3.5) and that  $b_0 = 0$  and

$$b_1 = \frac{1}{2\pi} \int_{L_1} \mu(s) ds.$$

We have also used the method to solve exterior Dirichlet problems. In order to solve  $\Delta u = 0$  in a finite region exterior to a simply connected domain we assume that  $u$  has the form

$$u(t) = \frac{1}{2\pi} \int_{\partial\Omega} \mu(s) \frac{\partial \log r(s, t)}{\partial n_s} ds + \frac{1}{2\pi} \int_{\partial\Omega} \mu(s) ds.$$

See [11, p. 140]. The dipole density is then the solution of the equation

$$\mu(t) + \frac{1}{2\pi} \int_{\partial\Omega} \mu(s) \left[ \frac{\partial \log r(s, t)}{\partial n_s} + 1 \right] ds = 2u(t).$$

It is now necessary to compute an approximation to  $u$  by evaluating an integral at the edge of the regular region as well as on the two exterior rings of mesh points  $B_2$  and  $B_3$ .

We note that without an integral equation formulation it is quite difficult to decide how to specify data on the edge of the grid when solving an exterior problem.

#### 4. Method of computation and estimate of accuracy.

**4.1. Solution of the integral equation.** We first solve the integral equation. In order to do this we use the Nystrom method with the trapezoid rule with uniform mesh width as the quadrature formula.

The integral equations which we solve are always Fredholm integral equations of the second kind, that is equations of the form:

$$(4.1) \quad \mu(t) + \frac{1}{\pi} \int \mu(s) g(s, t) ds = f(s).$$

The Nystrom method for (4.1) uses a quadrature formula to obtain the approximating equation:

$$(4.2) \quad \mu_n(s) + \sum_{j=1}^n g(s, t_j) \mu_n(t_j) = f(s)$$

where  $t_1, t_2, \dots, t_n$  are the nodes.

Equation (4.2) is solved by reducing it to the equivalent linear system (see [1, p. 13]):

$$(4.3) \quad \mu_n(t_i) + \sum_j g(t_i, t_j) \mu_n(t_j) = f(t_i).$$

We have chosen to use the trapezoid rule with mesh points equally spaced with respect to the boundary parameter as the quadrature formula in our experiments because it is extremely accurate on periodic regions. We can see why this is so by examining the Euler–Maclaurin summation formula. It says that if  $T(k)$  is the approximation to the integral of a function  $f \in C^{2r}(a, b)$  using the trapezoid rule with uniform mesh width  $k$  then

$$T(k) = \int_b^a f(x) dx + k^2/12[f'(b) - f'(a)] - k^4[f'''(b) - f'''(a)] \\ + k^6/30,240[f'(b) - f'(a)] + \dots + c_{2r}k^{2r}[f^{2r-1}(b) - f^{2r-1}(a)] + O(k^{2r+2}), \\ k \rightarrow 0.$$

Here  $c_r = (-1)^{r+1} B_r / (2r)!$  where  $\{B_r\}$  are the Bernoulli numbers. See [6, p. 297]. If  $f(x)$  is periodic and infinitely differentiable then all the boundary terms vanish, and so we see that the error approaches 0 faster than any power of the mesh width.

This fact guarantees the accuracy of the solution of (4.3). Specifically, Anselone [1, p. 297] has proved a result that shows that if  $n$  is sufficiently large then the accuracy of the solution of a Fredholm integral equation of the second kind with a unique solution is the same as the accuracy of the quadrature formula. This explains the unusually high accuracy we obtained when we solved problems on regions with smooth boundaries.

**4.2. Evaluation of the integrals.** Once the integral equation is solved we compute the solution by evaluating integrals at the appropriate mesh points.

In most of our experiments we used the trapezoid rule with uniform mesh width  $k$  to evaluate the integrals. The accuracy we can achieve when we have smooth data depends on the relative sizes of  $k$  and  $h$ , the mesh widths on the curve and on the regular region. Because the points at which we evaluate the integrals are no closer than distance  $h$  from  $\partial\Omega$  and because of our regularity assumptions we have  $|g(s, t)| < C/h$ . Similarly, since  $\mu(s)$ ,  $x(s)$ ,  $y(s)$  and their derivatives are bounded we have  $|(d^i/ds^i)g(s, t)\mu(s)| = O(1/h^{i+1})$ .

Therefore, if we choose  $k = O(h^{3/2})$ , then by using only the first three terms in the Euler–Maclaurin formula we see that the error we make in computing the integral is in theory  $O(k^4/h^4) = O(h^2)$ . Evidently we must choose  $k = O(h^{7/4})$  to obtain third order accuracy in  $h$ . This relationship has not been fully tested in experiments, but we did of course find that when we decreased the ratio  $k/h$  the accuracy we achieved in computing the integrals increased, and that when the boundary data was more oscillatory the accuracy decreased.

In any case, choosing  $k$  too small can be prohibitively expensive for solving the integral equations. However, since we solve the integral equation so accurately it is

easy to overcome this problem. In some examples we first solved the integral equation with a relatively large mesh width and then interpolated the values of discrete density function with a cubic spline  $\mu$ . Then we refined the mesh on the boundary and used the trapezoid rule on the refined mesh to compute the solution at the interior points.

**4.3. The effect of the error due to forming the discrete Laplacian.** Once we have computed the approximation  $\{U_{ij}\}$  to the potential function  $u$  on  $B_2$  and  $B_3$  we then form the discrete Laplacian  $(\Delta_h^5 U)_{ij}$  at points of  $B$  and apply the fast solver on  $R$ . In this way we obtain values  $\{v_{ij}\}$  for the potential function throughout  $D$ . If the discrete Laplacian of  $U$  is second order accurate, then  $v$  will also be second order accurate. This follows from the standard error estimate using the maximum norm for the solution of  $\Delta_h^5 W_{ij} = f_{ij}$  on a rectangle  $R$  with sides  $a$  and  $b$  [14, p. 221]:

$$(4.4) \quad \max_R |W_{ij}| < \frac{1}{4}(a^2 + b^2) \max_R |f_{ij}| + \max_{\partial R} |W_{ij}|.$$

Letting  $W_{ij} = u_{ij} - v_{ij}$  and noting that  $u_{ij} = v_{ij}$  on  $\partial R$  the result follows. However we could, it seems, lose two orders of accuracy by computing the discrete Laplacian. Indeed, if  $|U - u_{ij}| = O(\epsilon)$  on  $B_2$  and  $B_3$ , then we only have  $|\Delta_h^5 U - \Delta_h^5 v_{ij}| = O(\epsilon/h^2)$  on those points. Then (4.4) only implies  $\max |u_{ij} - v_{ij}| = O(\epsilon/h^2)$ .

Fortunately better bounds are possible. Certainly empirical evidence indicates that this is so. In fact we found that there was little or no loss of accuracy due to forming the discrete Laplacian and applying the fast solver. Part of the reason for this is because the values of the computed discrete Laplacian can be less than second order accurate only on  $B$  which is sparse in  $R$ .

In the appendix we present a result that shows why in many cases we can expect small loss of accuracy due to forming the discrete Laplacian and applying the fast solver, and that even in some bad cases we can only expect to lose one order of accuracy. This result also indicates that in order to minimize the possible loss of accuracy one should choose the boundary of the regular region so the average distance of points in  $B$  to  $\partial R$  will be small.

**5. The computation of the conjugate function.** Because we use an integral equation formulation to solve the Dirichlet problem the dipole density is available for further calculations. Therefore, when we compute a harmonic function  $u$  on a region we find it inexpensive to also compute its conjugate function.

If in particular if the region is simply connected then the conjugate function  $v(t)$  can be expressed as

$$v(t) = \frac{1}{\pi} \int_{\partial\Omega} \mu(s) \frac{\partial \log r(s, t)}{\partial s} ds.$$

The kernel which occurs in the formula for  $v(t)$  is very similar to the kernel for  $u(t)$ . It turns out that if  $\omega(n)$  is the number of operations required for a fast solver,  $p$  is the number of interior irregular mesh points, and  $q$  is the number of nodes on  $\partial\Omega$  then only an additional  $8pq + \omega(n)$  operations are needed to find  $v$  in the interior. If one wanted to compute  $v$  on  $\partial\Omega$  then the situation would be more difficult because the kernel is unbounded there.

We note that the conjugate of a harmonic function as well as the function itself are needed in certain methods for conformal mapping and in certain methods for solving the biharmonic equation. We also note that since we can compute the conjugate of a harmonic function that we can also solve Neumann problems. This is because by integrating Neumann data we obtain Dirichlet data for the conjugate function.

**6. Fourth order accurate solver.** One of the most important properties of this method is that it can be made higher order accurate so inexpensively. This is true primarily because the most time consuming part of the calculation, solving for the dipole density can be done so accurately with relatively few boundary nodes.

In order to obtain a fourth order accurate solution we first decide on a fourth order accurate approximation to the Laplacian. We again let  $D$  consist of all the points in  $\Omega$  that are at least  $h$  away from  $\partial\Omega$ .

In our experiments we used the following fourth order accurate approximation to the Laplacian:

$$L_h = \frac{1}{12h^2} \begin{bmatrix} & & & & 1 \\ & & & & -16 \\ & & 1 & -16 & 60 & -16 & 1 \\ & & & & -16 & & \\ & & & & & & 1 \end{bmatrix}.$$

When we use this approximation to the Laplacian the set  $S$  at which the discrete Laplacian is not set equal to zero consists of four rings of mesh points. It follows that we must evaluate integrals on four rings of points. Once we have done this we form the discrete Laplacian and apply a fast solver which uses the same approximation to the Laplacian. We note that in some cases it might be preferable to use a different fourth order approximation to the Laplacian.

**7. Program outline and operation count.** In this section we give details of the computational method and the operation counts.

We assume that the irregular boundary is given in parametric form  $\partial\Omega = (x(s), y(s))$ . Since we need to evaluate the curvature we assume that its first two derivatives are also available. If they are not they may be approximated by using splines.

(i) We first solve the integral equation. This requires first generating the matrix which is the discrete form of the integral operator. If there are  $q$  points on the boundary and the trapezoid rule is the quadrature formula then  $C1q^2$  operations are required to generate the matrix equation. Here an operation is defined to be one multiplication and one addition and  $C1$  is at most 10.

(ii) Next we perform the most expensive part of the calculation, the solution of the matrix equation. Since the matrix is dense and nonsymmetric we have been solving the equation by Gaussian elimination. The matrix is first factored into the product of a lower and an upper triangular matrix. This requires  $1/3q^3$  operations. If we are performing many calculations on the same region then we can save these factors. Therefore, each additional calculation requires only  $q^2$  operations. We note that since the trapezoid rule is so accurate we can sometimes choose  $q$  less than  $p$  where  $p$  is the number of interior irregular points. When this is so the matrix we have to invert is relatively small. One might also consider iterative methods of solving the integral equation.

(iii) After this we compute an approximation to the potential function on points of  $B2$  and  $B3$ . If forming the kernel function requires  $C$  operations per point, computing  $u$  on a ring of  $\kappa$  points will cost  $C\kappa q$  operations. If we are doing many calculations on the same region we may store the values of the kernel function.

(iv) Next we form the discrete Laplacian. This requires only  $O(n)$  operations.

(v) Finally we apply a fast solver. These typically require  $O(n^2 \log_2 n)$  operations [8].



**8. Results of numerical experiments.** In this section we report on results of numerical experiments. All were carried out on the CDC6600 computer at the Courant Institute except for the ones using a fourth order accurate solver which were performed on a CDC7600 machine at the University of California at Berkeley. In all cases the irregular region was embedded in the unit square, and the running times quoted include the time needed to set up the problems, to generate the data and to find the irregular mesh points. In the tables  $n$  is the number of mesh points in each direction across the square, and  $q$  is the number of mesh points we used on the boundary curve.

In the first set of experiments we tested our second order accurate solver on the problem  $\Delta u = 0$  with Dirichlet data prescribed on various regions. The results are summarized in Table 1. Table 2 gives the results of experiments where we computed the conjugate as well as the function. In Table 3 we give the results of solving a modified Dirichlet problem. (Recall a modified Dirichlet problem is a Dirichlet problem on a nonsimply connected region where the solution is required to be the real part of a single valued analytic function.) In these examples the integrals were evaluated using the trapezoid rule and a Buneman solver written by Anthony Jameson was used as the fast solver.

Our main purpose throughout has been to study the effectiveness of our method of using the fast Poisson solver to extend values of the potential function obtained near the boundary using the integral equation formulation. Since we knew that the fast Poisson solver was second order accurate we therefore worked primarily with the harmonic polynomial  $x^2 - y^2$  for which there is no discretization error. Any errors in the final solution were therefore due to errors in computing the integrals and the ensuing errors in the discrete Laplacian. We note that for this test function the maximum error in the interior was often very close to the maximum error obtained in computing  $u$  on the ring  $B_2$ . This tends to confirm what was suggested in § 4. When we used test functions for which there is discretization error, the error in the interior was of course larger.

We note that we were not primarily interested in the method of solving the integral equation or in the method of computing the integrals. However, we naturally found that when the data was more oscillatory we could not achieve the same degree of accuracy in computing  $u$  on  $B_2$  as we could when  $u = x^2 - y^2$ . We also noted that increasing  $n$  and not  $q$  did not improve the accuracy of the solution obtained after applying the fast solver. In particular, the error in computing  $x^2 - y^2$  with  $n = 64$  and  $q = 100$  was relatively large. This occurred because by making  $n$  so large some points at which we had to evaluate an integral were quite close to the boundary. In order to achieve greater accuracy we would have had to increase  $q$  much more. However increasing  $q$  too much proved very expensive. It was for this reason that in the starred examples we interpolated the solution of the integral equation with a cubic spline and refined the mesh on the boundary by a factor of two before computing the integrals. Of course this increased the running time somewhat. (Note the difference in running time for computing  $x^2 - y^2$  with  $n = 32$ ,  $q = 60$  and for  $x^4 - 6x^2y^2 + y^4$  with the same values for  $n$  and  $q$ .)

We note that in all cases tested when the test function  $u$  was the same and we had  $p$  and  $q$  equal then we achieved essentially the same accuracy on the ellipse, on the nonsimply connected region, and on the limaçon which is nonconvex. However, if we had used test regions which have regions of high curvature of corners then we could not have solved the integral equation as accurately without making  $q$  much larger.

The results listed in Table 2 show that for small additional cost we can compute the conjugate of a harmonic function as well as the function itself. Moreover, we

achieved essentially the same degree of accuracy in computing the conjugate. We note that the running time would have been approximately doubled if we had computed the solution on the boundary as well since that requires integrating a logarithmic singularity.

In Table 4 we give the results of experiments using our fourth order method. The results do demonstrate that greater accuracy can be achieved by using a higher order accurate approximation to the Laplacian. However in order to achieve this increased accuracy one must compute the integrals much more accurately.

TABLE 1  
Second order accurate solver.

Boundary data	$n$	$q$	Max. error in $U$ on $B_2$	Max. error in $U$ on $B_3$	Max. error in $D$	CPU time (sec.)
Region: Interior of ellipse centered at (.5, .5) with semiaxes .3 and .35						
$x^2 - y^2$	16	50	$.58 \cdot 10^{-4}$	$.17 \cdot 10^{-7}$	$.37 \cdot 10^{-5}$	.323
	16	60	$.13 \cdot 10^{-4}$	$.14 \cdot 10^{-10}$	$.12 \cdot 10^{-5}$	.495
	16	80	$.27 \cdot 10^{-6}$	$.44 \cdot 10^{-13}$	$.16 \cdot 10^{-6}$	1.019
	*32	60	$.12 \cdot 10^{-5}$	$.14 \cdot 10^{-6}$	$.17 \cdot 10^{-5}$	.590
	32	80	$.37 \cdot 10^{-4}$	$.41 \cdot 10^{-7}$	$.50 \cdot 10^{-4}$	1.126
$x^4 - 6x^2y^2 + y^4$	64	100	$.43 \cdot 10^{-3}$	$.53 \cdot 10^{-4}$	$.44 \cdot 10^{-3}$	2.76
	*16	60	$.93 \cdot 10^{-6}$	$.43 \cdot 10^{-7}$	$.58 \cdot 10^{-3}$	.897
	*32	60	$.90 \cdot 10^{-6}$	$.74 \cdot 10^{-6}$	$.13 \cdot 10^{-3}$	1.21
$\sin 7x \cosh 7y$	*64	70	$.93 \cdot 10^{-4}$	$.13 \cdot 10^{-4}$	$.34 \cdot 10^{-3}$	1.91
Region: Interior region bounded by ellipse with semiaxes .3 and .35, and circle of radius .2, both centered at (.5, .5)						
$x^2 - y^2$	32	120	$.33 \cdot 10^{-3}$	$.58 \cdot 10^{-6}$	$.58 \cdot 10^{-3}$	3.21
Region: $r = .2 \cos \theta + .25$ (nonconvex)						
$x^2 - y^2$	16	60	$.21 \cdot 10^{-4}$	$.18 \cdot 10^{-8}$	$.43 \cdot 10^{-5}$	.521
	32	60	$.28 \cdot 10^{-3}$	$.32 \cdot 10^{-5}$	$.14 \cdot 10^{-3}$	.610
Region: Exterior of circle of radius .2 centered at (.5, .5)						
$\frac{6(x-.5)}{(x-.5)^2 + (y-.5)^2}$	32	80	$.24 \cdot 10^{-3}$	$.59 \cdot 10^{-7}$	$.10 \cdot 10^{-3}$	1.31

TABLE 2  
Computation of harmonic function and conjugate. Test function,  $u = x^2 - y^2$ .

Region	$n$	$q$	Max. error in $u$ on $D$	Max. error in conjugate on $D$	CPU time (sec.)
.3 - .35 ellipse $r = .15 \cos \theta + .3$	32	60	$.12 \cdot 10^{-5}$	$.13 \cdot 10^{-5}$	.648
	16	60	$.21 \cdot 10^{-4}$	$.12 \cdot 10^{-4}$	.677

TABLE 3  
Modified Dirichlet problem.

Interior region bounded by ellipse with semiaxes .3 and .35 and circle of radius .2

Boundary data	$n$	$q$	Max. error in $u$ on $D$	CPU time (sec.)
$x^2 - y^2$	32	120	$.44 \cdot 10^{-3}$	2.78

TABLE 4  
Fourth order accurate solver.

Dirichlet data given on ellipse with semiaxes .3 and .35

Boundary data	$n$	$q$	Max. error on ring 1	Max. error on ring 2	Max. error on ring 3	Max. error on ring 4	Max. error in $D$	CPU time (sec.)
$x^4 - 6x^2y^2 + y^4$	*32	80	$.93 \cdot 10^{-6}$	$.43 \cdot 10^{-7}$	$.35 \cdot 10^{-11}$	$.50 \cdot 10^{-12}$	$.50 \cdot 10^{-6}$	.356
$\sin 7x \cosh 7y$	*32	60	$.93 \cdot 10^{-4}$	$.13 \cdot 10^{-4}$	$.17 \cdot 10^{-7}$	$.76 \cdot 10^{-12}$	$.41 \cdot 10^{-4}$	.380

**9. Other extensions.** In this section we report on possible extensions of this method.

First, we need not embed the entire irregular region in the same larger regular region. Although we must solve the integral equation on the entire boundary, we can afterwards embed different parts of the irregular region in different regular regions.

We also expect that our method could be applied to three-dimensional and inhomogeneous problems. Also, this method can be extended to other linear elliptic equations in other coordinates when a fundamental solution is known and a fast solver is possible.

Finally we mention that since we seem to be able to evaluate the density so accurately, then we should be able to evaluate the derivative of a potential function directly. By integrating by parts we obtain formulas for the derivatives in which the growth of the kernel is no greater than the growth of the kernel in the formula for the function itself. Indeed, since

$$u(t) = \operatorname{Re} \frac{1}{2\pi i} \int_{\partial\Omega} \frac{\mu(s)}{s-t} ds$$

(see [11, p. 138]) we have

$$\begin{aligned} u_x(t) &= \operatorname{Re} \frac{1}{2\pi i} \int_{\partial\Omega} \frac{d}{dt} \frac{\mu(s)}{(s-t)} ds \\ &= \operatorname{Re} \frac{1}{2\pi i} \int_{\partial\Omega} \frac{\mu'(s)}{(s-t)^2} dt \\ &= \frac{1}{2\pi} \int_{\partial\Omega} \frac{\mu'(s)}{x'(s)^2 + y'(s)^2} \left[ \frac{\partial \log r(s,t)}{\partial n_s} x'(s) - \frac{\partial \log r(s,t)}{\partial s} y'(s) \right] ds. \end{aligned}$$

**Appendix.** We now prove a result that indicates why in many cases we can expect little loss of accuracy due to forming the discrete five point Laplacian of  $U_{ij}$  and applying the fast solver, and show that in no case should more than one order of accuracy be lost. We also show that the loss of accuracy can be decreased by choosing the boundary of the embedding region  $R$  to be close to the boundary of the irregular region.

We first show that it is enough to consider the effect of the error in computing  $\Delta_h^5$  on  $B3$  alone.

**DEFINITION.** Let  $S_{ij}$  be a mesh function defined on  $R$ . We say  $S_{ij}^*$  is the discrete harmonic extension of  $S_{ij}$  on  $B2$  if

$$\left. \begin{matrix} S_{ij}^* \\ (\Delta_h^5 S^*)_{ij} \\ S_{ij}^* \end{matrix} \right\} = \begin{cases} S_{ij}, & (i, j) \in B2, \\ 0, & (i, j) \in D - B, \\ 0, & (i, j) \in R - D. \end{cases}$$

**CLAIM.** If  $U_{ij}^*$  is the discrete harmonic extension of  $U_{ij}$  on  $B2$  and we apply our procedure with  $U_{ij} = U_{ij}^*$  on  $B3$ , then  $v_{ij} = U_{ij}^*$  on  $R$ .

This follows because (see § 2)

$$(\Delta_h^5 U^*)_{ij} = \begin{cases} (\Delta_h^5 v)_{ij} = 0 & \text{on } D - B, \\ (\Delta_h^5 v)_{ij} & \text{on } B = B1 \cup B2 \text{ by construction,} \\ (\Delta_h^5 v)_{ij} = 0 & \text{on } R - (B \cup D). \end{cases}$$

Letting  $W_{ij} = U_{ij}^* - v_{ij}$  and using the discrete maximum principle for  $\Delta_h^5$  [9, p. 447] we see that  $U_{ij}^* = v_{ij}$ . This shows that if the values  $U_{ij}$  on  $B2 \cup B3$  are second order accurate, and if on  $B3$ ,  $U_{ij}$  is the discrete harmonic extension of  $U$  on  $B2$ , then  $v$  will be second order accurate throughout  $R$ . Therefore, if  $|U_{ij} - u_{ij}| = O(h^2)$  on  $B2 \cup B3$  then it suffices to consider the effect of an error of order  $h^2$  on  $B3$  alone. We now prove a result that indicates that the closer points of  $B3$  are to the boundary of  $R$ , then the smaller this effect is. We first note that the solution of  $(\Delta_h^5 W)_{ij} = f_{ij}$  with  $w_{ij} = 0$  on  $\partial R$  can be expressed as  $W_{ij} = -h^2 G_h(i, i', j, j')$  where  $0 \leq G_h(i, i', j, j')$  [2]. Therefore, in the worst case the errors have the same sign.

Now let  $R_k$  be the  $k$ th ring of mesh points in  $R$  where the rings are nested and start at the boundary. For example,  $R_0 = \partial R$ , and  $R_1$  consists of mesh points which have neighboring mesh points in  $\partial R$ .

Let  $E_k(i, j)$  be the solution of

$$\Delta_h^5 E_k(i, j) = \begin{cases} 1, & (i, j) \in R_k, \\ 0 & \text{elsewhere,} \end{cases}$$

$$E_k(i, j) = 0, \quad (i, j) \in \partial R.$$

**LEMMA.**  $-kh^2 \leq E_k(i, j) \leq 0$  for  $(i, j) \in R$ .

*Proof.* By the maximum principle  $E_k(i, j) \leq 0$ . Let  $R_k^*$  be the four corner points of  $R_k$ . Consider the mesh function  $F_k(i, j)$  defined as the solution of

$$\Delta_h^5 F_k(i, j) = \begin{cases} 1 & \text{on } R_k - R_k^*, \\ 2 & \text{on } R_\lambda^*, \quad 1 \leq \lambda \leq k, \\ 0 & \text{elsewhere,} \end{cases}$$

$$F_k(i, j) = 0 \quad \text{on } \partial R.$$

$F_k$  can be found exactly:

$$F_k = \begin{cases} -mh^2 & \text{on } R_m, \quad 1 \leq m \leq k-1, \\ -kh^2 & \text{elsewhere.} \end{cases}$$

We note  $\Delta_h^5 E_k \leq \Delta_h^5 F_k$  inside  $R$  and  $F_k(i, j) = E_k(i, j)$  on  $\partial R$ . It follows that  $F_k(i, j) \leq E_k(i, j)$  and so  $-kh^2 \leq E_k(i, j)$ .

This shows that if  $B3 = R_k$  and if the error made in computing the integral on  $R_k$  was in absolute value less than  $\varepsilon$ , then the effect of that error in the interior would be less than  $k\varepsilon h^2$ . Suppose now that  $m$  is the average distance of the points of  $B3$  from the boundary of  $R$ . One can see that as  $m$  increases the error due to forming the Laplacian and using the fast solver grows like  $mh^2$ . Of course, in the worst case  $m$  is close to  $1/h$ , and so one order of accuracy is lost. Therefore one must compute the solution to higher accuracy on both  $B2$  and  $B3$ . But we note that our estimate is very pessimistic since in general since the errors in the discrete Laplacian will have different signs and some of their effects will tend to cancel.

**Acknowledgments.** The author would like to thank Professor Charles Peskin for suggesting this problem and for his help during its completion. The author has also benefitted from discussions with Professor Olof Widlund.

#### REFERENCES

- [1] P. M. ANSELONE, *Collectively Compact Operator Approximation Theory*, Prentice-Hall, Englewood Cliffs, NJ, 1971.
- [2] J. H. BRAMBLE, *On the convergence of difference approximations to weak solutions of Dirichlet's problem*, Numer. Math., 13 (1969), pp. 101–111.
- [3] O. BUNEMAN, *A compact non-iterative Poisson solver*, Rep. SUIPR-294, Institute for Plasma Research, Stanford Univ., Stanford, CA, 1969.
- [4] B. L. BUZBEE, F. W. DORR, J. A. GEORGE AND G. H. GOLUB, *The direct solution of the discrete Poisson equation on irregular domains*, SIAM J. Numer. Anal., 8 (1971), pp. 722–736.
- [5] R. COURANT AND D. HILBERT, *Methods of Mathematical Physics*, Interscience, New York, 1953.
- [6] G. DAHLQUIST AND A. BJORCK, *Numerical Methods*, Prentice-Hall, Englewood Cliffs, NJ, 1974.
- [7] D. FISHER, G. GOLUB, O. HALD, C. LEIVA AND O. WIDLUND, *On Fourier-Toeplitz methods for separable elliptic problems*, Math. Comp., 28 (1974), pp. 349–368.
- [8] R. W. HOCKNEY, *The potential calculation and some applications*, Methods in Computational Physics, vol. 9, Academic Press, New York, 1970, pp. 135–211.
- [9] E. ISAACSON AND H. KELLER, *Analysis of Numerical Methods*, John Wiley, New York, 1966.
- [10] S. G. MIKHLIN, *Dirichlet's problem for regions with several closed boundaries*, Dokl. Akad. Nauk, SSSR, no. 7, 1934. (In Russian.)
- [11] ———, *Integral Equations and Their Applications*, Pergamon, New York, 1957.
- [12] D. O'LEARY AND O. WIDLUND, *Capacitance matrix methods for the Helmholtz equation on general three dimensional regions*, Math. Comp., 33 (1979), pp. 849–879.
- [13] 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.
- [14] G. SMITH, *Numerical Solution of Partial Differential Equations: Finite Difference Methods*, Oxford Univ. Press, Oxford, 1978.

## NUMERICAL ANALYSIS OF SPECTRAL PROPERTIES OF COUPLED OSCILLATOR SCHRÖDINGER OPERATORS III. THE DOUBLING ALGORITHM\*

D. ISAACSON†, E. L. ISAACSON‡, D. MARCHESIN§ AND P. J. PAES-LEME¶

**Abstract.** We describe a recursive numerical algorithm for the computation of the lowest eigenvalues of Schrödinger energy operators involving coupled anharmonic oscillators. Given eigenpairs for  $N$  oscillators, the algorithm computes eigenpairs for  $2N$  oscillators. We compute 48 of the eigenvalues of 4, 8 and 16 coupled oscillators. For large numbers of oscillators, iterative methods are used, such as block-Lanczos and inverse iteration via pre-conditioned SYMMLQ.

**Key words.** Schrödinger operators, eigenvalues of coupled anharmonic oscillators, doubling algorithm, block-Lanczos, pre-conditioned SYMMLQ

**1. Introduction.** We describe in detail a numerical method for approximating the lowest eigenvalues and eigenfunctions of coupled anharmonic oscillator Schrödinger operators. We have used it to compute up to 48 eigenvalues for 2, 4, 8 and 16 coupled oscillators. The method is uniformly accurate for all values of the parameters describing the anharmonic oscillators. It can be extended to compute eigenvalues, eigenfunctions and other quantities of physical interest for other coupled systems.

The use of numerical methods for computing spectral properties of coupled anharmonic oscillators goes back at least as far as [17]. Much of the recent literature has been devoted to approximating the lowest eigenvalue by constructing a trial function for the corresponding eigenfunction of the  $N$  oscillator operator, especially as  $N$  tends to infinity. Discussions of these methods may be found in [15], [2], [11]. For two coupled anharmonic oscillators the only computations of spectral properties of which we are aware are in [1] and [6]. The algorithm that we use is different from any that have appeared in the literature. It can also be used to give approximations to the infinite oscillator limit. However, this paper is devoted to a description of the algorithm and of the results obtained for a few oscillators.

First we describe briefly how these operators approximate  $\phi_2^4$  quantum field theory energy operators.

One of the simplest classical field equations that is relativistically covariant and has nontrivial scattering is  $\phi_{tt} - \phi_{xx} + a\phi + 2b\phi^3 = 0$ . The energy of a field satisfying this equation is given by its classical Hamiltonian

$$H_{cl}(\phi, \phi_t) = \frac{1}{2} \int_{-\infty}^{+\infty} (\phi_t^2 + \phi_x^2 + a\phi^2 + b\phi^4) dx.$$

Classically a measurement of the total energy of this field can yield any value in the range of  $H_{cl}$ . According to quantum mechanics, a measurement of the total energy of

\* Received by the editors July 28, 1982, and in final revised form July 4, 1983.

† Rensselaer Polytechnic Institute, Troy, New York 12181. The research of this author was supported in part by National Science Foundation grants MCS-80-02938 and INT-79207728.

‡ University of Wyoming, Laramie, Wyoming 82071. The research of this author was supported in part by National Science Foundation grant PHY-80-09179 and NSF Postdoctoral Fellowship SPI-8009169 while he was at Rockefeller University, New York, New York.

§ Pontifícia Universidade Católica do Rio de Janeiro, Brazil. The research of this author was supported in part by National Science Foundation grant PHY-80-09179.

¶ Pontifícia Universidade Católica do Rio de Janeiro, Gavea, Brazil CEP 22453. The research of this author was supported in part by CNPq/National Science Foundation grant 0310.1465/80.

a physical system can only yield a value in the spectrum of a certain self-adjoint operator, the quantum mechanical Hamiltonian associated with the system. For conservative mechanical systems with finitely many degrees of freedom there is a recipe (called quantization) for finding this quantum mechanical Hamiltonian. It consists of replacing the momentum variables in the classical Hamiltonian function by the operator  $-i \partial/\partial\phi$ . Thus, we approximate  $H_{cl}(\phi, \phi_t)$  by a Hamiltonian of a system with finitely many degrees of freedom. Then we apply the quantization recipe and arrive at the coupled anharmonic oscillator Schrödinger operators whose eigenvalues are of physical interest.

Specifically, choose  $l > 0$ ,  $N > 0$  and let  $\delta = l/N$ ,  $x_j = -l/2 + j\delta$ ,  $q_j \equiv \phi(x_j)$ ,  $p_j \equiv \phi_t(x_j)$  for  $j = 0, 1, 2, \dots, N$ . Then  $H_{cl}(\phi, \phi_t)$  may be approximated by the Riemann sum

$$\frac{\delta}{2} \sum_{j=0}^N \{p_j^2 + aq_j^2 + bq_j^4 + (q_{j+1} - q_j)^2/\delta^2\}.$$

Note that we use a one-sided difference as opposed to a central one. Our algorithm, described in § 3, is recursive precisely because the cross terms in the formula above involve only consecutive indices. We consider mainly two types of boundary conditions: Dirichlet ( $q_0 = q_{N+1} = 0$ ) and periodic ( $q_0 = q_{N+1}$ ). In either case we obtain a classical  $N$ -anharmonic oscillator Hamiltonian of the form

$$\frac{1}{2} \sum_{j=1}^N \left( \delta p_j^2 + V(q_j) \right) - d \sum_j q_j q_{j+1}$$

where

$$V(q_j) = a' q_j^2 + b' q_j^4, \\ a' = (a + 2/\delta^2)\delta, \quad b' = b\delta, \quad d = 1/\delta$$

and the last summation goes from 1 to  $N - 1$  in the Dirichlet case, and from 1 to  $N$  in the periodic case. For simplicity we write all of our formulas for the Dirichlet case.

Following the quantization recipe,  $p_j$  is replaced by  $-i \partial/\partial q_j$ , yielding the  $N$ -oscillator Schrödinger operator

$$(1.1) \quad H^{(N)} = H^{(N)}(q_1, \dots, q_N) \equiv \frac{1}{2} \sum_{j=1}^N \left( -\delta \frac{\partial^2}{\partial q_j^2} + V(q_j) \right) - d \sum_{j=1}^{N-1} q_j q_{j+1}.$$

The eigenvalues of  $H^{(N)}$  approximate the total energy of the system described classically by  $H_{cl}(\phi, \phi_t)$ . (A more detailed physical description and derivation are given in [4].)

Our algorithm reduces the eigenvalue problem for  $H^{(N)}$  to the diagonalization of a sparse operator. For a small number of oscillators, a direct method can be used to perform the diagonalization. Otherwise, iterative methods are used, e.g., block-Lanczos and inverse iteration via pre-conditioned SYMMLQ.

Some relevant properties of the Hamiltonian  $H^{(N)}$  are listed in § 2. In § 3 we describe the doubling algorithm we introduced in [4]–[6] for the purpose of numerically approximating the eigenpairs of  $H^{(N)}$  when  $N = 2^L$  and  $L = 0, 1, 2, 3, \dots$ . We discuss the implementation of the algorithm in § 4. Section 5 contains the results. Useful formulae for the Dirichlet and periodic Hamiltonians are found in the Appendices of [9].

**2. Properties of the  $N$ -oscillator Hamiltonian.** We describe the analytical properties of the Hamiltonian which motivate our method.

The closure of the restriction of  $H^{(N)}$  to the space of  $C^\infty$  rapidly decreasing functions is a self-adjoint operator on  $L_2(\mathbb{R}^N)$ .  $H^{(N)}$  is bounded from below and has

a compact resolvent.  $H^{(N)}$  has a complete orthonormal set of real eigenfunctions  $\Psi_j^{(N)}$  with eigenvalues  $E_j^{(N)}$ . Thus

$$H^{(N)}\Psi_j^{(N)} = E_j^{(N)}\Psi_j^{(N)}, \quad j = 1, 2, 3, \dots$$

and

$$E_1^{(N)} < E_2^{(N)} \leq E_3^{(N)} \leq E_4^{(N)} \leq \dots$$

All of the eigenfunctions are  $C^\infty$  and rapidly decreasing. We have

$$H^{(2N)} = A^{(2N)} + B^{(2N)}$$

where

$$A^{(2N)}(\underline{1}, \underline{2}) = H^{(N)}(\underline{1}) + H^{(N)}(\underline{2})$$

and

$$B^{(2N)}(\underline{1}, \underline{2}) = -dq_N q_{N+1}.$$

(Here  $\underline{1}$  denotes the set of variables  $q_1, \dots, q_N$  and  $\underline{2}$  denotes the set of variables  $q_{N+1}, \dots, q_{2N}$ .) The term  $B^{(2N)}$  is relatively bounded with respect to  $A^{(2N)}$  in the sense of Kato (see [10, Chap. 4, § 1, Section 1]). In fact a simple computation shows that

$$\pm B^{(2N)} \leq \mu A^{(2N)} + z(\mu)I.$$

When  $b' > 0$  in  $V(q_i)$ ,  $\mu > 0$  may be chosen as small as we like. This and higher order estimates involving powers of  $B^{(2N)}$  and  $A^{(2N)}$  yield a rapid convergence of the eigenfunctions of  $H^{(2N)}$  when expressed in terms of the eigenfunctions of  $A^{(2N)}$ . The convergence has been proven to be faster than any polynomial and uniform away from  $b = 0$  for two oscillators [6]. Thus, the eigenfunctions of  $A^{(2N)}$  are a good choice for a basis in which to represent  $H^{(2N)}$ .

The eigenfunctions of  $A^{(2N)}$  are  $\Psi_i^{(N)}(\underline{1})\Psi_j^{(N)}(\underline{2})$ ; i.e., they are the tensor products of the eigenfunctions of  $H^{(N)}(\underline{1})$  with those of  $H^{(N)}(\underline{2})$ . Moreover, any function  $\Psi \in L^2(\mathbb{R}^{2N})$  may be represented with respect to the basis  $\{\Psi_i^{(N)}(\underline{1})\Psi_j^{(N)}(\underline{2})\}$  by a matrix  $C \in l^2(\mathbb{N} \times \mathbb{N})$ ; i.e., the map

$$\Psi \rightarrow C = [C_{ij}]$$

is unitary where

$$(2.1) \quad \Psi(\underline{1}, \underline{2}) = \sum C_{ij} \Psi_i^{(N)}(\underline{1}) \Psi_j^{(N)}(\underline{2}).$$

(All indices range over  $1, 2, \dots$  unless stated otherwise.)

The matrix eigenvalue problem corresponding to

$$H^{(2N)}\Psi = E\Psi$$

is

$$E^{(N)}C + CE^{(N)} - dX^{(N)}CY^{(N)} = EC.$$

Here  $E^{(N)}$  is the diagonal matrix with entries  $E_j^{(N)}$ , and  $X^{(N)}$ ,  $Y^{(N)}$  are the matrices representing multiplication in  $L^2(\mathbb{R}^N)$  by  $q_N, q_1$  respectively. That is,

$$(2.2) \quad \begin{aligned} X_{ij}^{(N)} &\equiv \langle \Psi_i^{(N)}(\underline{1}), q_N \Psi_j^{(N)}(\underline{1}) \rangle, \\ Y_{ij}^{(N)} &\equiv \langle \Psi_i^{(N)}(\underline{1}), q_1 \Psi_j^{(N)}(\underline{1}) \rangle. \end{aligned}$$



(Here  $\langle \cdot, \cdot \rangle$  is the standard inner product on  $L^2(\mathbb{R}^N)$ .) The formula for the matrix eigenproblem results from the formula for applying the operator  $H^{(2N)}$  to a matrix  $C$ :

$$(2.3) \quad C \rightarrow E^{(N)}C + CE^{(N)} - dX^{(N)}CY^{(N)};$$

(A derivation of this formula is presented in [9].)

Formula (2.3) shows that it is easy to apply the  $2N$ -oscillator Hamiltonian given the  $N$ -oscillator information  $E^{(N)}$ ,  $X^{(N)}$ ,  $Y^{(N)}$ . This allows us to compute the eigenvalues  $E_j^{(2N)}$  and matrix representations  $C^j$  of the corresponding eigenfunctions  $\Psi_j^{(2N)}$  of  $H^{(2N)}$ . From these it is possible to compute  $X^{(2N)}$ ,  $Y^{(2N)}$  through the formulae

$$(2.4) \quad \begin{aligned} X_{ij}^{(2N)} &= (C^i X^{(N)}) \cdot C^j, \\ Y_{ij}^{(2N)} &= C^i \cdot (Y^{(N)} C^j) \end{aligned}$$

where  $A \cdot B \equiv \sum_{m,n} A_{mn} B_{mn}$ . The first equation follows from

$$\begin{aligned} X_{ij}^{(2N)} &= \langle \Psi_i^{(2N)}(\underline{1}, \underline{2}), q_{2N} \Psi_j^{(2N)}(\underline{1}, \underline{2}) \rangle \\ &= \sum_{m,n,r,s} C_{mn}^i \langle \Psi_m^{(N)}(\underline{1}) \Psi_n^{(N)}(\underline{2}), q_{2N} \Psi_r^{(N)}(\underline{1}) \Psi_s^{(N)}(\underline{2}) \rangle C_{rs}^j \\ &= \sum_{m,n,r,s} C_{mn}^i \langle \Psi_m^{(N)}(\underline{1}), \Psi_r^{(N)}(\underline{1}) \rangle \langle \Psi_n^{(N)}(\underline{2}), q_{2N} \Psi_s^{(N)}(\underline{2}) \rangle C_{rs}^j \\ &= \sum_{m,n,s} C_{mn}^i X_{ns}^{(N)} C_{ms}^j. \end{aligned}$$

The formula for  $Y^{(2N)}$  is proved similarly.

Formulae (2.3), (2.4) are the basic algebraic motivation of the doubling algorithm: information for the  $2N$ -oscillator Hamiltonian can be computed from similar information for the  $N$ -oscillator Hamiltonian.

To reduce computational costs, it is advantageous to diagonalize  $H^{(N)}$  in as many invariant subspaces as possible. Therefore we look for all of the symmetries in the problem which are compatible with the doubling procedure.

Define the operators  $R^{(N)}$  and  $S^{(N)}$  on  $L_2(\mathbb{R}^N)$  by

$$(2.5) \quad \begin{aligned} R^{(N)}\Psi(q_1, \dots, q_N) &\equiv \Psi(q_N, \dots, q_1) \quad \text{and} \\ S^{(N)}\Psi(q_1, \dots, q_N) &\equiv \Psi(-q_1, \dots, -q_N); \end{aligned}$$

these operators are both unitary and self-adjoint. Furthermore,  $H^{(N)}$ ,  $R^{(N)}$  and  $S^{(N)}$  all commute. Thus, the eigenfunctions of  $H^{(N)}$  may be chosen to be eigenfunctions of both  $R^{(N)}$  and  $S^{(N)}$ . With this choice

$$\begin{aligned} R^{(N)}\Psi_j^{(N)} &= r_j^{(N)}\Psi_j^{(N)}, \\ S^{(N)}\Psi_j^{(N)} &= s_j^{(N)}\Psi_j^{(N)} \end{aligned}$$

where  $r_j^{(N)}, s_j^{(N)} = \pm 1$ . Therefore, it suffices to compute the eigenfunctions of  $H^{(N)}$  in each of the four invariant subspaces:

$$(2.6) \quad \begin{aligned} \mathfrak{H}_{++}^{(N)} &\equiv \{\Psi \in L_2(\mathbb{R}^N): R^{(N)}\Psi = \Psi, S^{(N)}\Psi = \Psi\}, \\ \mathfrak{H}_{+-}^{(N)} &\equiv \{\Psi \in L_2(\mathbb{R}^N): R^{(N)}\Psi = \Psi, S^{(N)}\Psi = -\Psi\}, \\ \mathfrak{H}_{-+}^{(N)} &\equiv \{\Psi \in L_2(\mathbb{R}^N): R^{(N)}\Psi = -\Psi, S^{(N)}\Psi = \Psi\}, \\ \mathfrak{H}_{--}^{(N)} &\equiv \{\Psi \in L_2(\mathbb{R}^N): R^{(N)}\Psi = -\Psi, S^{(N)}\Psi = -\Psi\}. \end{aligned}$$

We remark that  $\Psi_1^{(N)} \in \mathfrak{H}_{++}^{(N)}$ ; i.e.,  $r_1^{(N)} = s_1^{(N)} = +1$ .

**3. The algorithm.** The algorithm begins with an initialization in which the 1-oscillator information is determined. This is followed repeatedly by a sequence of two steps in which the  $2N$ -oscillator information is computed from the  $N$ -oscillator information. At the beginning of each two-step sequence, a truncation number  $K$  is chosen to determine the size of the matrix eigenproblem to be solved.

*Initialization.* Set  $N = 1$ . Find the eigenvalues and eigenfunctions of the one oscillator Hamiltonian, namely

$$\frac{1}{2} \left( -\frac{\partial^2}{\partial q_1^2} + V(q_1) \right) \Psi_j^{(1)} = E_j^{(1)} \Psi_j^{(1)}, \quad j = 1, 2, 3, \dots$$

Also compute the matrices  $X^{(1)}$ ,  $Y^{(1)}$  given by

$$X_{ij}^{(1)} = Y_{ij}^{(1)} = \int_{-\infty}^{+\infty} \Psi_i^{(1)}(q_1) q_1 \Psi_j^{(1)}(q_1) dq_1.$$

In the present paper we are not concerned with the method for carrying out this initialization. Instead, we refer the reader to [5].

*Step 1.* Choose the truncation size  $K$ . Given  $K$  eigenvalues  $E_j^{(N)}$  and the  $K \times K$  matrices  $X^{(N)}$ ,  $Y^{(N)}$  for  $N$ -oscillators, compute the eigenvalues and the eigenvectors of the Hamiltonian of  $2N$  coupled oscillators. The eigenvectors are represented by matrices (2.1). The Hamiltonian acts on matrices by

$$C \rightarrow E^{(N)} C + C E^{(N)} - d X^{(N)} C Y^{(N)}.$$

(See (2.3).) The lowest eigendata in each of the four invariant subspaces (2.6) is computed by one of two methods.

a) Compute the matrix representation of  $H^{(2N)}$  in each subspace through formula (2.3) and diagonalize it directly.

b) Use an iterative method in each subspace based on repeated applications of  $H^{(2N)}$  through formula (2.3). It is shown in [9] that formula (2.3) has a simple counterpart in each subspace.

The direct and iterative methods that we have used are described in § 4.

*Step 2.* Given the matrix representation  $C^j$ ,  $j = 1, 2, \dots$  of the eigenfunctions of  $H^{(2N)}$  computed in Step 1, as well as the matrices  $X^{(N)}$  and  $Y^{(N)}$ , compute the matrices  $X^{(2N)}$  and  $Y^{(2N)}$  defined by (2.2) using the formulae (2.4):

$$X_{ij}^{(2N)} = (C^i X^{(N)}) \cdot C^j, \quad Y_{ij}^{(2N)} = C^i \cdot (Y^{(N)} C^j).$$

Formulae which generalize (2.4) are proven in [9]. To achieve a reduction in computing cost, we replace the matrix  $Y$  by the matrix  $X$  through a simple change of variable. Before performing the computations in Step 2 the eigendata from Step 1 is reordered. This is accomplished by interlacing the eigenpairs from the four subspaces (2.6).

*Step 3.* Double  $N$  and go back to Step 1.

The truncation size  $K$  is chosen in such a way that the error in the lowest eigendata for  $H^{(2N)}$  is small enough. Obviously,  $K$  cannot be larger than the number of eigenpairs computed in the previous doubling.

We remark that the eigenfunctions  $\Psi_i^{(2N)}(1, 2)$  of  $H^{(2N)}$  are not calculated. Instead, their matrix representations with respect to the basis  $\{\Psi_i^{(N)}(1) \Psi_j^{(N)}(2)\}$  are computed. To recover the true eigenfunction it is necessary to perform a recursive procedure using all of the intermediate matrix representations starting with the 1-oscillator eigenfunctions. Nevertheless, our experience indicates that to compute quantities of physical interest it is not necessary to recover the true eigenfunctions.

Now we summarize the properties of the coupled system which are important for the doubling algorithm.

First, an inductive formula holds such as

$$H^{(2N)}(\underline{1}, \underline{2}) = H^{(N)}(\underline{1}) + H^{(N)}(\underline{2}) - dQ^{(N)}(\underline{1})Q^{(N)}(\underline{2}).$$

In general  $Q^{(N)}(\underline{1})$  is an operator in the variables  $(q_1, \dots, q_N)$  and  $Q^{(N)}(\underline{2})$  the same operator in the variables  $(q_{2N}, \dots, q_{N+1})$ . Moreover, the matrix representation of  $Q^{(2N)}(\underline{1}, \underline{2})$  should be easily computable from that of  $Q^{(N)}(\underline{1})$  and the eigendata of  $H^{(2N)}$ . (In our case,  $Q^{(N)}(\underline{1}) = q_N$ , and the formula for  $Q^{(2N)}(\underline{1}, \underline{2})$  is given in (2.4).) These are the essential features of the problem for our algorithm.

Second, the operators  $R^{(2N)}$ ,  $S^{(2N)}$  and  $H^{(2N)}$  commute. In addition, if  $\Phi, \Psi$  are two functions in any of the invariant subspaces (not necessarily the same one), then

$$\begin{aligned} R^{(2N)}\Phi(\underline{1})\Psi(\underline{2}) &= (R^{(N)}\Psi(\underline{1}))(R^{(N)}\Phi(\underline{2})), \\ S^{(2N)}\Phi(\underline{1})\Psi(\underline{2}) &= (S^{(N)}\Phi(\underline{1}))(S^{(N)}\Psi(\underline{2})). \end{aligned}$$

These facts let us work in each invariant subspace with representations of  $H^{(2N)}$  truncated to finite size, as proved in [9].

Each eigenvalue and eigenvector obtained for 2 oscillators converges faster than any inverse power of  $K$ , as proven in [6]. A similar result holds for  $N$  oscillators.

**4. Implementation of the algorithm.** A simple-minded version of the algorithm is based on the matrix representation of the operator (2.3) acting on the space spanned by the truncated tensor product basis

$$(4.1) \quad \{\Psi_i^{(N)}(\underline{1})\Psi_j^{(N)}(\underline{2})\}_{i,j=1}^K.$$

This is a  $K^2$ -dimensional basis leading to a  $K^2 \times K^2$  matrix to be diagonalized.

Taking advantage of the symmetries (2.5), the operator can be represented in each of the four subspaces (2.6). Since each truncated subspace has dimension  $K^2/4$  (approximately), and since the cost of diagonalizing an  $n \times n$  matrix via standard methods is proportional to  $n^3$ , this restriction reduces the computing time by a factor of sixteen. Thus, it is more efficient to compute the lowest  $K'$  eigenpairs in each subspace than to compute the lowest  $4K'$  eigenpairs overall.

The formula (2.3) has a simple counterpart in each subspace. These formulae are employed to generate matrices which are diagonalized using EISPACK routines [19]. On the CDC 6600 and the IBM 370/165, this procedure becomes impractical when  $K$  is greater than 20 because of both computer time and memory limitations.

Recalling (2.3), we note that the cost of applying this formula to a vector in the subspace (4.1) is  $2(K^2 + K^3)$  multiplications, even though the operator is represented in this subspace by a  $K^2 \times K^2$  matrix. For  $K$  large this yields an effective sparseness of  $2/K$ , which can be exploited through the usage of iterative diagonalization methods based directly on (2.3).

We have employed two methods to diagonalize the operator  $H^{(2N)}$  which are based on repeated applications of (2.3).

The Lanczos method finds tridiagonal matrices which are unitarily equivalent to the operator  $H^{(2N)}$  [22], [13], [16]. For our applications the main disadvantage of this method is that the convergence is slow for eigenvalues in clusters and is not guaranteed for degenerate eigenvalues. There are other disadvantages as well, most of which are solved by the block-Lanczos method [21], [14]. We use Underwood's version.

We note that the spectrum of  $H^{(2N)}$  is sparse at the lower end and concentrated everywhere else (see the figures). Thus the block-Lanczos procedure is adequate to

compute the lower part of the spectrum, which is exactly our aim. However, a severe limitation of the block-Lanczos method did arise. Even though the accuracy of the extreme eigenpairs was high, the accuracy of the interior eigenpairs could not be made satisfactory by increasing the number of iterations. This problem was obviated through the use of the second method which we now describe.

Suppose we are given a sufficiently good approximate value  $\lambda$  of a nondegenerate eigenvalue of the operator  $H^{(2N)}$ . The corresponding eigenpair can be computed using first the inverse power method and then inverse iteration [23], [22]. Each step in this procedure involves solving the linear system

$$Pu_n = u_{n-1}$$

where  $P = H^{(2N)} - \lambda I$ .

To solve these systems we take advantage of the effective sparseness of  $H^{(2N)}$  by using a method which requires only a procedure that applies the operator to a vector. For an indefinite symmetric system  $Pu = d$  a good iterative scheme is SYMMLQ [12], [20]. This algorithm is very economical in its memory requirements since it needs storage for only a few vectors. The convergence is faster the closer  $P$  is to a multiple of the identity. Often there is an approximation  $Q$  of  $P$  which is symmetric, positive definite and computationally inexpensive to invert. In this case the original system is replaced by  $(Q^{-1/2}PQ^{-1/2})v = Q^{-1/2}d$  leading to a preconditioned SYMMLQ algorithm, each step of which applies  $P$  and  $Q^{-1}$  once.

We determine an initial approximate eigenvalue  $\lambda$  by the block-Lanczos procedure. As a preconditioning for SYMMLQ we use the absolute value of  $\text{diag}(H^{(2N)} - \lambda I)$ , namely the operator

$$(4.2) \quad \psi_i(1)\psi_j(2) \rightarrow |E_i^{(N)} + E_j^{(N)} - \lambda| \psi_i(1)\psi_j(2).$$

This preconditioning is about 50% more effective than the one used in [20] where  $\lambda$  is replaced by zero in (4.2). Moreover, it makes SYMMLQ several times faster. The importance of the preconditioning is described in [18], [20].

The spectra for 2 and 4 oscillators were computed using EISPACK. For 8 and 16 oscillators, the block-Lanczos method was employed. For 16 oscillators, the data from some intermediate steps were used to provide initial approximations for the preconditioned SYMMLQ algorithm. We used an 8-byte version of Underwood's code with 16-byte accumulators. The block tridiagonal matrix found by the procedure was  $60 \times 60$  with block size 4. The tolerance was set to  $10^{-6}$  for the lowest eigenvalue. The CPU time required to find the spectrum in the doubling from 2 to 4 oscillators was 46 seconds using EISPACK. In the doubling from 4 to 8 the time was 309 seconds using block-Lanczos and 50 seconds using preconditioned SYMMLQ. The times mentioned above refer to an IBM370/165.

The entire procedure gives satisfactory results as long as the eigenvalues do not become almost degenerate. The spectrum becomes continuous as the number of doublings is increased [4]. Our computations yield physically interesting results before breaking down [8].

**5. Results.** Our main results are presented in the figures. The one oscillator Hamiltonian is

$$(5.1) \quad H^{(1)}(q_1) = \frac{1}{2} \left[ -\frac{\partial^2}{\partial q_1^2} + (3 - \omega g)q_1^2 + gq_1^4 \right]$$

where  $\omega$  is a constant which depends on the computation under consideration. We

use  $g$  as a parameter in all our figures. Plots for the eigenvalues of  $H^{(1)}$  and  $H^{(2)}$  may be found in [5] and [6].

The lowest 12 eigenvalues of  $H^{(4)}$  in each invariant subspace are all shown in Fig. 5.1. Similar quantities for  $H^{(8)}$  and  $H^{(16)}$  are shown in Figs. 5.2, 5.3.

In each figure all eigenvalues are shifted so that the lowest eigenvalue of the corresponding Hamiltonian is zero (vacuum renormalization). The constant  $\omega$  in (5.1) is chosen in such a way that the slope of the second lowest eigenvalue is zero at  $g=0$  (Wick ordering).

We note that the gap between the lowest two eigenvalues approximates the mass of the lightest particle in the  $g:\phi^4_2$  quantum field theory. The other eigenvalues approximate the energies of higher momentum or many particle states.

We are unaware of any other method which is accurate for  $g$  in the intermediate region ( $0 \ll g \ll \infty$ ). This is especially true for the critical region in which the mass (i.e., the difference between the second and the first eigenvalues) tends to zero as  $N$  increases.

Plots for the periodic Hamiltonian as well as a discussion of the physical significance of these computations may be found in [8].

For  $g=0$  (free-field) the exact eigenvalues for any number of oscillators may be computed explicitly. They are

$$\sum_{j=1}^N \left( n_j + \frac{1}{2} \right) \mu_j, \quad n_j = 0, 1, 2, \dots$$

where

$$\mu_j = \left( 1 + 4 \sin^2 \frac{j\pi}{2(N+1)} \right)^{1/2}.$$

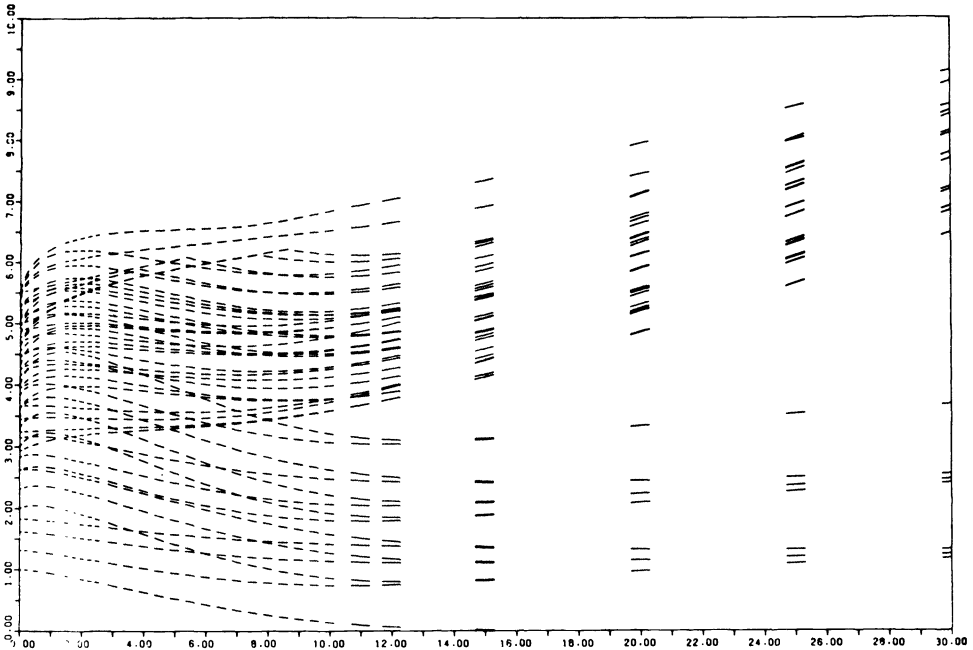


FIG. 5.1. Spectrum of  $H^{(4)}$  as a function of  $g$ . We show the 12 lowest eigenvalues in each of the four invariant subspace.

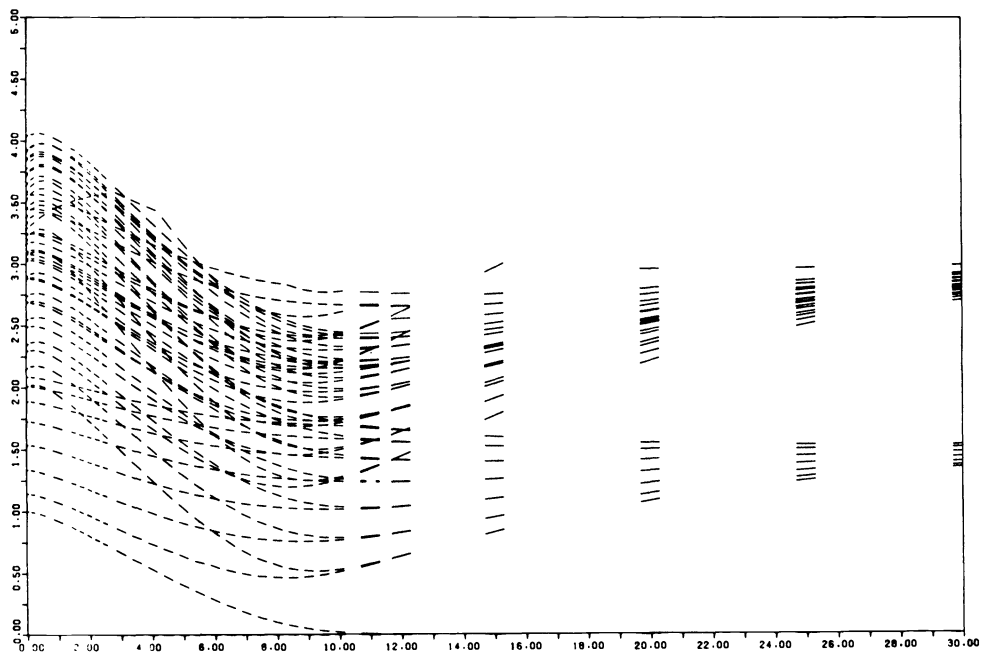


FIG. 5.2. Spectrum of  $H^{(8)}$  shown as in Fig. 5.1.

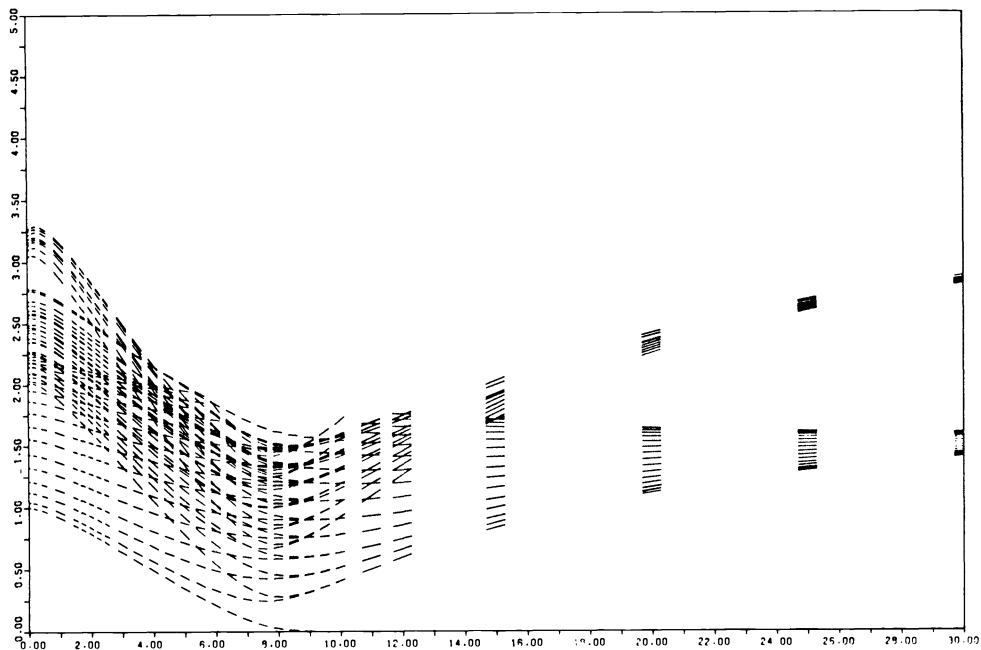


FIG. 5.3. Spectrum of  $H^{(16)}$  shown as in Fig. 5.1.

This fact provides an excellent check for our computations. The relative errors are shown in Table 5.1 for the subspace  $\mathfrak{S}_{++}^{(N)}$  in (2.6).

TABLE 5.1  
Relative errors in the eigenvalues in the  $\mathfrak{S}_{++}^{(N)}$  subspace as a function of  $N$  (number of oscillators) for  $g=0$ .

eigenvalue \ $N$	2	4	8	16
1st	$1.1 \times 10^{-7}$	$2.0 \times 10^{-7}$	$3.6 \times 10^{-6}$	$7.9 \times 10^{-5}$
5th	$1.1 \times 10^{-7}$	$1.3 \times 10^{-6}$	$5.5 \times 10^{-5}$	$4.0 \times 10^{-4}$
10th	$2.3 \times 10^{-7}$	$1.9 \times 10^{-5}$	$1.6 \times 10^{-4}$	$1.8 \times 10^{-3}$

In all figures we plot the eigenvalues with their slopes. It is easy to compute the derivatives of the eigenvalues with respect to the parameter  $g$  [9]. To ensure graphical accuracy of the plots, much better accuracy is required in the initial doublings. Table 5.2 presents the total number  $K$  of eigenpairs used in the doubling procedure as a function of the number  $N$  of coupled oscillators.

TABLE 5.2  
Total number  $K(N)$  of eigenpairs used in our computations in the doubling procedure from  $N/2$  to  $N$  coupled oscillators. These numbers ensure at least 12 accurate eigenpairs per subspace.

$N$	2	4	8	16
$K(N)$	14	24	32	32

A detailed error analysis, employed to determine these numbers, is found in [9].

REFERENCES

[1] R. BLANKENBECLER, T. DE GRAND AND R. L. SUGAR, *Moment method for eigenvalues and expectation values*, Phys. Rev. D, 21 (1980), p. 1055.  
 [2] J. B. BRONZAN AND R. L. SUGAR, *Thinning degrees of freedom in lattice field theories*, Phys. Rev. D, 21 (1980), p. 1567.  
 [3] M. R. HESTENES AND E. STIEFEL, *Methods of conjugate gradients for solving linear systems*, J. Res. Nat. Bur. Standards, 49 (1952), pp. 409–436.  
 [4] D. ISAACSON, D. MARCHESIN AND P. J. PAES-LEME, *Numerical methods for studying anharmonic oscillator approximations to the  $\phi_2^4$  quantum field theory*, Int. J. Engng. Sci., 18 (1980), pp. 341–349.  
 [5] D. ISAACSON, E. L. ISAACSON, D. MARCHESIN AND P. J. PAES-LEME, *Numerical analysis of spectral properties of coupled oscillator Schrödinger operators I—Single and double well anharmonic oscillators*, Math. Comp., 37 (1981), pp. 273–292.  
 [6] ———, *Numerical analysis of spectral properties of coupled oscillator Schrödinger operators II—Two coupled anharmonic oscillators*, SIAM J. Numer. Anal., 19 (1982), pp. 126–141.  
 [7] ———, *Critical behavior of the two-state doubling algorithm*, J. Math. Phys., 24 (1983), pp. 41–45.  
 [8] ———, *Eigenvalues of 4, 8, 16 coupled anharmonic oscillators*, Phys. Rev. D, 27 (1983), pp. 3036 ff.  
 [9] ———, *Numerical analysis of spectral properties of coupled oscillator Schrödinger operators III. The doubling algorithm*, Tech. Rep. MAT 014, Pontificia Universidade Católica do Rio de Janeiro, Brazil, April 1982.  
 [10] T. KATO, *Perturbation Theory for Linear Operators*, Springer, New York, 1972.  
 [11] A. F. PACHECO, *Block-spin method for the  $\phi^4$  theory on a lattice*, Phys. Rev. D, 23 (1981), p. 1845.  
 [12] C. C. PAIGE AND M. A. SAUNDERS, *Solution of sparse indefinite systems of linear equations*, SIAM J. Numer. Anal., 12 (1975), pp. 617–629.

- [13] B. N. PARLETT, *The Symmetric Eigenvalue Problem*, Prentice-Hall, Englewood Cliffs, NJ, 1980.
- [14] B. N. PARLETT AND D. S. SCOTT, *The Lanczos algorithm with selective orthogonalization*, *Math. Comp.*, 33 (1979), pp. 217–238.
- [15] J. L. RICHARDSON AND R. BLANKENBECLER, *Anharmonic analysis of lattice field theories*, *Phys. Rev. D*, 20 (1979), p. 1351.
- [16] Y. SAAD, *On the rates of convergence of the Lanczos and the block-Lanczos methods*, *SIAM J. Numer. Anal.*, 17 (1980), pp. 687–706.
- [17] L. I. SCHIFF, *Lattice-space quantization of a non-linear field theory*, *Phys. Rev.*, 92 (1953), pp. 766–779.
- [18] D. S. SCOTT, *Solving sparse symmetric generalized eigenvalue problems without factorization*, *SIAM J. Numer. Anal.*, 18 (1981), pp. 102–110.
- [19] B. T. SMITH et al., *Matrix Eigensystem Routines*, Eispack Guide, Springer, New York, 1974.
- [20] D. B. SZYLD AND O. B. WIDLUND, *Application of conjugate gradient type methods to eigenvalue calculations*, in *Advances in Computer Methods for Partial Differential Equations III*, R. Vichnevetsky and P. Stepleman, eds., IMACS, Rutgers Univ., New Brunswick, NJ, 1979, pp. 167–173.
- [21] R. UNDERWOOD, *An iterative block-Lanczos method for the solution of large sparse symmetric eigenproblems*, Computer Science Dept. Rept. 496, Stanford Univ., Stanford, 1975.
- [22] J. H. WILKINSON, *The Algebraic Eigenvalue Problem*, Clarendon Press, Oxford, 1965.
- [23] ———, *Inverse iteration in theory and in practice*, *Symposia Mathematica*, Vol. X, (1972), pp. 361ff.



## UNBIASED MONTE CARLO INTEGRATION METHODS WITH EXACTNESS FOR LOW ORDER POLYNOMIALS\*

ANDREW F. SIEGEL† AND FANNY O'BRIEN‡

**Abstract.** We consider methods for estimation of the integral of a given function that combine the unbiasedness of Monte Carlo integration (which permits a simple statistical assessment of the error) with the higher precision often attained by deterministic methods. We propose symmetric random designs with  $2k + 1$  points that achieve exactness for polynomials of degree up to  $2k + 1$ . The distribution for the three-point method is unique, although for higher order methods there are multiple choices for the sampling distribution. For two-dimensional multiple integration over a rectangle, we propose an unbiased five-point method that achieves exactness for polynomials of degree up to three in both variables. Some bounds on error variances are given.

**Key words.** numerical integration, random quadrature

**1. Introduction and summary.** Computing the integral of a given function is an important activity of many mathematical and statistical investigations. For integrals that are not amenable to direct calculation, either because the functional form of the integrand is not completely specified or because of mathematical intractability, useful approximations are provided by the many methods of numerical quadrature. We will consider estimation of the integral

$$(1.1) \quad I_f = \int_{-1}^1 f(x) dx$$

based on weighted sums of the functional values at  $n$  points:

$$(1.2) \quad I_f(\xi_1, \dots, \xi_n) = \sum_{i=1}^n w_i(\xi_1, \dots, \xi_n) f(\xi_i)$$

and also by the averages of values of (1.2) obtained from independent replications.

Because different numerical methods choose different weightings and different points at which to evaluate the integrand, the development and selection of good general quadrature formulae can be viewed as a problem in the statistical field of the design of experiments. Even with the availability of powerful computers, there remains a need for increased accuracy with less computational effort for the routine task of evaluating integrals. Part of our motivation for developing these hybrid methods arose from efficiency considerations in a large-scale Monte Carlo study of the properties of robust estimators (Tukey and Pregibon (1981), Bell and Pregibon (1981), Bell (1981)).

Many general methods fall within the weighted-sum framework of (1.2) with various choices for the weighting functions  $w_i$  and for the joint distribution of the

---

\* Received by the editors May 16, 1983, and in revised form November 7, 1983. This work was supported in part by U.S. Army Research Office under contracts DAAG29-79-C-0205 and DAAG29-82-K-0178, and by the U.S. Department of Energy under contract DE-AC02-81ER10841. Some revisions were made while A. F. Siegel was on leave at the Departments of Biostatistics and Statistics, University of Washington at Seattle. We are grateful to J. W. Tukey, S. Morgenthaler, C. Nachtsheim, and P. Bloomfield for helpful suggestions during the development of these methods. F. O'Brien was formerly named F. Zambuto.

† Department of Statistics and Department of Finance, Business Economics and Quantitative Methods, University of Washington, Seattle, Washington 98195.

‡ Department of Market Analysis and Forecasting, AT&T Communications, Bedminster, New Jersey 07921.

design points

$$(1.3) \quad (\xi_1, \dots, \xi_n) \sim F.$$

This formulation includes deterministic methods, such as Newton–Cotes and Gaussian integration, as well as random methods such as Monte Carlo. Deterministic methods can be represented by choosing the distribution  $F$  to be degenerate at the fixed design points and choosing the weights to be the appropriate constants. The simple Monte Carlo method can be represented by choosing  $n = 1$ ,  $w_1 = 1$ , and  $\xi_1$  uniformly distributed in  $(-1, 1)$ .

Some advantages and disadvantages are inherent in any method. Many deterministic methods are exact for low order polynomials, thereby achieving high accuracy for smooth functions. However, the natural error bounds associated with deterministic methods often require extensive analytical computations involving bounds on a higher derivative of the integrand. On the other hand, the standard error or confidence interval for the mean computed from independent replications of unbiased Monte Carlo methods provide a simple probabilistic error estimate requiring essentially no analytic effort. The primary drawback of simple Monte Carlo is that while the precision is easily assessed, this precision may not be very high. For example, simple Monte Carlo integration of the very smooth function  $f(\xi) = \xi^2$  with a sample of  $n = 1000$  replications does not even achieve two digits of accuracy 90% of the time.

One approach that seems promising is to combine the good properties of each method: the high accuracy for smooth functions of deterministic methods with the simply-assessed error estimates of Monte Carlo. We will investigate random methods that provide unbiased estimates of the integral of any integrable function, yet still provide the exact answer should the function happen to be a polynomial of low order; the variance in this case would be zero and the confidence interval would be degenerate.

Section 2 provides a review of past work in this area. In § 3 we show that there exists a unique three-point symmetric randomized design that achieves unbiasedness in general together with exactness for polynomials up to degree three, and we provide a variance bound for the integral estimate of a smooth function. For some nonsmooth functions and for functions observed with additive random errors, finite variance with exactness for cubics can only be achieved by considering designs of more than three points. Five-point designs with exactness for quintics are considered in § 4 where we show that the design is not unique. We exhibit several design choices that provide general unbiasedness together with exactness up to degree 5 and prove that this nonuniqueness cannot be exploited in order to gain exactness for polynomials of even higher degree.

In § 5, for the general case of designs of  $2k + 1$  points which provide exactness up to degree  $2k + 1$  we provide a class of symmetric random designs in which the joint distribution  $F$  in (1.3) is a multivariate polynomial expressed using Vandermonde determinants. Two-dimensional numerical integration over a rectangle is considered in § 6. In addition to the natural nine-point cross-design which is exact for polynomials of degree up to three jointly in both variables, we exhibit an economical five-point design which achieves polynomial accuracy of the same degree at almost half the computational effort. In a numerical example, simple Monte Carlo is shown to require about twenty times as many function evaluations to attain comparable accuracy.

While the methods presented here can be used directly, they can also be used effectively in conjunction with adaptive partitioning algorithms, as in Friedman and Wright (1981). In this case, the unbiased variance estimate could be used to provide their required measure of “badness” of the integrand within a subregion.

**2. Literature review.** Several investigators have provided examples of methods that achieve exactness for integrating polynomials while being unbiased for almost any function. This approach represents one of several techniques available for variance reduction in Monte Carlo investigations, which are reviewed by Hammersley and Handscomb (1964) and by Haber (1970). These techniques are related to the method of antithetic variables, introduced by Hammersley and Morton (1956), which exploit negative correlation in order to reduce variance. In § 5.8 of Hammersley and Handscomb, a general method of orthogonal functions due to Ermakov and Zolotukhin (1960) was used to find several integration formulae, including an unbiased four-point method with exactness up to cubics. Because it allows a choice of basis functions, this general approach is very flexible and adaptable. Other extensions and examples may be found in Ermakov (1964) and in Handscomb (1964). Granovskii (1968) provides some examples along similar lines with exactness for nonpolynomials. A sampling procedure for the general method of Ermakov and Zolotukhin may be found in Bogues, Morrow, and Patterson (1981).

Haber (1969) proved the existence of unbiased methods with exactness for any given degree of polynomial using a very simple partially random design with a fixed set of points together with one single additional point uniformly distributed over the range of integration. Some general results including variance bounds were also provided.

While the methods of Quackenbush (1969) require more than the minimal number of function evaluations for exactness for polynomials of a given order, they provide more stable estimates for nonsmooth functions and also in the presence of random errors. This is important in statistical experimental design, where an experiment must be performed in order to evaluate the function at each point and randomness is necessarily introduced. However, for routine numerical integration this may not be an important consideration. Cranley and Patterson (1970), (1976) also describe methods that require more function evaluations than those to be described here.

**3. Exactness for cubics.** Every symmetric three-point quadrature design samples the integrand at 0,  $\xi$ , and  $-\xi$ , for some  $\xi$  in  $(0, 1)$ . In order to achieve exactness for quadratics, the integral estimate based on these points must be the integral of the interpolating quadratic, which is

$$(3.1) \quad I_f(\xi) = \frac{f(-\xi) - 2f(0) + f(\xi)}{3\xi^2} + 2f(0).$$

This formula is of the form (1.2) and, due to symmetry, is exact for polynomials up to degree three. Simpson's rule results from the (degenerately random) choice  $\xi = 1$  and is exact up to degree three. By choosing  $\xi = .7746 \dots$ , again degenerately random, Gaussian integration is able to achieve exactness for polynomials of degree up to five. The following theorem shows that if  $\xi$  is chosen randomly from the proper distribution, then the estimate in (3.1) can be made unbiased for the integral of any function. One appealing interpretation of this proposed formula is as a randomized version of Simpson's rule, in which the spacing is chosen from a probability distribution.

**THEOREM 3.1.** *If  $\xi$  is a random variable on  $(0, 1)$  with density  $3\xi^2$ , then  $I_f(\xi)$  given by (3.1) is both exact for cubic polynomials and also unbiased in general. That is,*

$$(3.2) \quad E\{I_f(\xi)\} = \int_{-1}^1 f(x) dx \quad \text{for any integrable } f$$

and

$$(3.3) \quad I_f(\xi) = \int_{-1}^1 f(x) dx$$

if  $f$  is a polynomial of degree three or less, provided only that  $\xi \neq 0$ . Moreover,  $3\xi^2$  is the unique density with these properties.

*Proof.* The potential singularity at  $\xi = 0$  cannot cause any problem with computation because  $P\{\xi = 0\} = 0$ . Exactness for polynomials of degree at most three has already been noted for (3.1) with any choice of  $\xi \neq 0$ . Unbiasedness follows because

$$\begin{aligned} E\{I_f(\xi)\} &= \int_0^1 \left[ \frac{f(-\xi) - 2f(0) + f(\xi)}{3\xi^2} + 2f(0) \right] 3\xi^2 d\xi \\ &= \int_0^1 f(-\xi) d\xi + \int_0^1 f(\xi) d\xi - 2f(0) + 2f(0) \\ &= \int_{-1}^1 f(\xi) d\xi = I_f. \end{aligned}$$

To establish uniqueness, suppose  $\mu$  is a distribution on  $(0, 1)$  such that  $\xi \sim \mu$  implies  $E\{I_f(\xi)\} = I_f$  for all integrable  $f$ . With the choice  $f_m(x) = 3|x|^{m+2}/2$ , it follows that

$$(3.4) \quad I_{[f_m]}(\xi) = \frac{(3/2)|-\xi|^{m+2} + (3/2)|\xi|^{m+2}}{3\xi^2} = \xi^m.$$

Hence a consequence of unbiasedness (3.2) is that

$$(3.5) \quad \int_0^1 \xi^m d\mu(\xi) = \frac{3}{m+3} \quad \text{for } m = 1, 2, \dots$$

Because these are the moments of  $\mu$ , and because distributions concentrated on a finite interval are determined by their moments (Feller (1971, § VII.3)), uniqueness follows.  $\square$

If the integrand is smooth, then the variance of the error can be bounded above in a way analogous to Simpson's rule.

**THEOREM 3.2.** *If the fourth derivative of  $f$  is bounded, and if  $\xi$  is distributed as in Theorem 3.1, then*

$$(3.6) \quad \text{St. Dev. } \{I_f(\xi)\} \leq \frac{M}{54.991}, \quad \text{where } M = \sup_{(-1, 1)} |f^{(4)}(x)|.$$

*This is about 1.6 times as large as the upper bound on the error of Simpson's rule, which is  $M/90$ .*

*Proof.* Using Taylor's theorem with remainder, expanding the terms of (3.1), we find that

$$(3.7) \quad I_f(\xi) = 2f(0) + \frac{f''(0)}{3} + \frac{\xi^2}{72} [f^{(4)}(\varepsilon_1) + f^{(4)}(-\varepsilon_2)]$$

where  $\varepsilon_1$  and  $\varepsilon_2$  are in  $[0, \xi]$  and are functions of  $\xi$ . The variance is

$$\begin{aligned}
 \text{Var} \{I_f(\xi)\} &= \text{Var} \{I_f(\xi) - 2f(0) - f''(0)/3\} \\
 &= \text{Var} \left\{ \frac{\xi^2}{72} [f^{(4)}(\varepsilon_1) + f^{(4)}(-\varepsilon_2)] \right\} \\
 (3.8) \quad &\leq E \left\{ \frac{\xi^2}{72} [f^{(4)}(\varepsilon_1) + f^{(4)}(-\varepsilon_2)] \right\}^2 \\
 &\leq \frac{M^2}{1296} E\{\xi^4\} = \frac{M^2}{3024}.
 \end{aligned}$$

The square root of this yields the bound in (3.6) for the standard error.  $\square$

If the integrand is bounded but not smooth at zero, then the integral estimate of Theorem 3.1 can have infinite variance. This happens, for example, with the discontinuous function  $f(x)$  which is 1 when  $x > 0$  and is 0 when  $x \leq 0$ . However, many functions with discontinuous derivatives at zero (such as the absolute value function) still have integral estimates with bounded variance provided they satisfy the condition of the following theorem.

**THEOREM 3.3.** *With the integral estimate of Theorem (3.1), if*

$$(3.9) \quad \left| \frac{f(x) - f(0)}{x} \right| \leq M \quad \text{for all } x \neq 0,$$

then

$$(3.10) \quad \text{St. Dev.} \{I_f(\xi)\} \leq \frac{M}{.8660}.$$

*Proof.* By subtracting the constant  $2f(0)$  and rearranging, the variance may be written as

$$\begin{aligned}
 \text{Var} \{I_f(\xi)\} &= \text{Var} \left[ \frac{\frac{f(\xi) - f(0)}{\xi} - \frac{f(-\xi) - f(0)}{(-\xi)}}{3\xi} \right] \\
 (3.11) \quad &\leq E \left[ \frac{\frac{f(\xi) - f(0)}{\xi} - \frac{f(-\xi) - f(0)}{(-\xi)}}{3\xi} \right]^2 \\
 &\leq \frac{4M^2}{9} E \left[ \frac{1}{\xi^2} \right] = \frac{4M^2}{3}. \quad \square
 \end{aligned}$$

Hammersley and Handscomb (1964, p. 74) also note problems with infinite variance and observe that in some cases a Lipschitz condition is sufficient to guarantee finiteness. If an unbiased method exact for cubics is needed that has finite variance for very nonsmooth functions (for example a function that is observed with independent random additive errors), then a design with more than three points must be used because of the uniqueness in Theorem (3.1). A six-point method was proposed in Quackenbush (1969) that achieves this, and Haber's (1969) method described in § 1 will also achieve this with three fixed and one random point. Quackenbush's formula is

$$(3.12) \quad J_f(\xi) = -\frac{1}{3} [f(\xi) + f(-\xi)] + \frac{2}{3} \left[ f\left(\frac{-1+\xi}{2}\right) + f\left(\frac{-1-\xi}{2}\right) + f\left(\frac{1+\xi}{2}\right) + f\left(\frac{1-\xi}{2}\right) \right]$$

where  $\xi$  is uniformly distributed on (0, 1).

In the presence of random independent additive errors with mean zero and variance  $\sigma^2$  added to each function evaluation, the variance of Quackenbush's integral estimate (3.12) will be exactly  $2\sigma^2$  larger than it would have been had the function been evaluated without error. In particular, this will be bounded.

**4. Exactness for quintics.** Every symmetric five-point quadrature design samples the integrand at  $0, \xi, -\xi, \eta,$  and  $-\eta,$  for some  $\xi$  and  $\eta$  in  $(0, 1)$ . In order to achieve exactness for polynomials of degree up to five, the integral estimate must be the integral of the interpolating quintic, which is

$$(4.1) \quad I_f(\xi, \eta) = \frac{3 - 5\eta^2}{15\xi^2(\xi^2 - \eta^2)} F(\xi) + \frac{3 - 5\xi^2}{15\eta^2(\eta^2 - \xi^2)} F(\eta) + 2f(0)$$

where, for simplicity, we have introduced the auxiliary function

$$(4.2) \quad F(x) = f(-x) - 2f(0) + f(x).$$

The five-point closed Newton-Cotes formula will result if we choose  $\xi = .5$  and  $\eta = 1$ . Choosing  $\xi = .538 \dots$  and  $\eta = .936 \dots,$  we obtain the Gaussian integration formula, which is exact for polynomials up to degree nine, the maximum possible precision for a five-point method.

It is possible to choose the pair  $(\xi, \eta)$  randomly from a probability distribution on the square  $(0, 1) \times (0, 1)$  so that (4.1) is an unbiased estimate of the integral of any function. Hammersley and Handscomb (1964) provide an example where  $\xi$  is fixed at 1 and only  $\eta$  is random. Another generalization of the results of § 3 is given in the following theorem where the distribution has a symmetric polynomial density.

**THEOREM 4.1.** *If  $(\xi, \eta)$  is a random vector distributed with density*

$$(4.3) \quad \phi(\xi, \eta) = 90\xi^2\eta^2(\xi + \eta)(\xi - \eta)^2 \quad \text{on } (0, 1) \times (0, 1),$$

*then  $I_f(\xi, \eta)$  given by (4.1) is both exact for quintic polynomials and also unbiased in general. That is,*

$$(4.4) \quad E\{I_f(\xi, \eta)\} = \int_{-1}^1 f(x) dx \quad \text{for any integrable } f$$

and

$$(4.5) \quad I_f(\xi, \eta) = \int_{-1}^1 f(x) dx$$

*if  $f$  is a polynomial of degree five or less, provided  $\xi \neq 0, \eta \neq 0,$  and  $\xi \neq \eta.$*

Because this result is a special case of Theorem 5.1 of the next section, a proof will not be given here. The variance bound will also be deferred to § 5.

A contour plot of the density (4.3) is shown in Fig. 4.1. Note that  $\xi$  and  $\eta$  tend to be chosen to be far from one another and also far from zero.

It is clear that the distribution given in Theorem 4.1 is not unique, due to the previously mentioned results in Hammersley and Handscomb. In fact, there are many other symmetric polynomial densities (but with higher degree than (4.3)) given by

$$(4.6) \quad \phi_\epsilon(\xi, \eta) = 90\xi^2\eta^2(\xi + \eta)(\xi - \eta)^2\{1 + \epsilon[32(\xi^2 + \xi\eta + \eta^2) - 35(\xi + \eta)]\}$$

where  $\epsilon$  is any positive constant such that (4.6) is nonnegative ( $0 < \epsilon < .005$  is sufficient). It can be established directly that (4.6) is a density on the unit square and that (4.4) and (4.5) also hold with this distribution.

The possibility of exploiting this nonuniqueness naturally arises, and one might try to find a distribution that minimizes the variance for a particular integrand. The

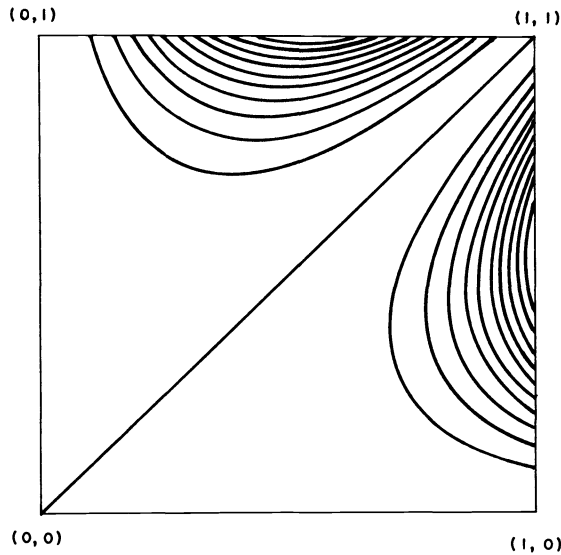


FIG. 4.1. A contour plot of the symmetric polynomial density of (4.3) that leads to exactness for polynomials of degree up to 5. The density is zero on the 45° line, and increases in equal steps of about .78.

following theorem shows that the nonuniqueness cannot be used, however, to achieve exactness for polynomials of the next higher degree.

**THEOREM 4.2.** *There is no distribution of  $(\xi, \eta)$  that yields an unbiased integral estimate (4.1) that is also exact for all polynomials of degree six or less.*

*Proof.* In order for (4.1) to give the exact integral of  $f(x) = x^6/2$ , we must have

$$(4.7) \quad I_f(\xi, \eta) = \frac{3 - 5\eta^2}{15\xi^2(\xi^2 - \eta^2)} \xi^6 + \frac{3 - 5\xi^2}{15\eta^2(\eta^2 - \xi^2)} \eta^6 = \frac{1}{7}$$

which simplifies to

$$(4.8) \quad 21(\xi^2 + \eta^2) - 35\xi^2\eta^2 - 15 = 0.$$

From this it can be verified that neither  $\xi$  nor  $\eta$  can take on a value between .655 and .845 (square roots of  $\frac{3}{7}$  and  $\frac{5}{7}$  respectively) because the solution of (4.8) would force the other to be outside the interval  $(0, 1)$ . A method that never samples the integrand within such an interval of positive measure cannot be unbiased, say, for the integral of a function that is zero outside that interval and positive within it.  $\square$

**5. A general formula for degree  $2k + 1$ .** Consider symmetric quadrature designs of  $2k + 1$  points consisting of distinct points  $\xi_1, \dots, \xi_k$  together with their negatives and zero. In order to achieve exactness for polynomials of degree up to  $2k + 1$ , the integral estimate must be the integral of the corresponding interpolating polynomial which can be written as follows:

$$(5.1) \quad I_f(\xi) = \sum_{i=1}^k w_i(\xi)F(\xi_i) + 2f(0) = \begin{bmatrix} \frac{1}{3} & \frac{1}{5} & \dots & \frac{1}{2k+1} \end{bmatrix} \begin{bmatrix} \xi_1^2 & \xi_1^4 & \dots & \xi_1^{2k} \\ \xi_2^2 & \xi_2^4 & \dots & \xi_2^{2k} \\ \vdots & \vdots & \ddots & \vdots \\ \xi_k^2 & \xi_k^4 & \dots & \xi_k^{2k} \end{bmatrix}^{-1} \begin{bmatrix} F(\xi_1) \\ F(\xi_2) \\ \vdots \\ F(\xi_k) \end{bmatrix} + 2f(0),$$

where  $F$  was defined in (4.2). The following theorem shows that there is a symmetric polynomial density for  $(\xi_1, \dots, \xi_k)$  such that (5.1) will be unbiased for the integral of any function.

**THEOREM 5.1.** *If  $(\xi_1, \dots, \xi_k)$  is distributed according to the density function*

$$\begin{aligned}
 (5.2) \quad \phi(\xi) &= \frac{\begin{vmatrix} \xi_1^2 & \xi_1^4 & \dots & \xi_1^{2k} \\ \xi_2^2 & \xi_2^4 & \dots & \xi_2^{2k} \\ \vdots & \vdots & \ddots & \vdots \\ \xi_k^2 & \xi_k^4 & \dots & \xi_k^{2k} \end{vmatrix} \begin{vmatrix} 1 & \xi_1 & \dots & \xi_1^{k-1} \\ 1 & \xi_2 & \dots & \xi_2^{k-1} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & \xi_k & \dots & \xi_k^{k-1} \end{vmatrix}}{k! \begin{vmatrix} \frac{1}{3} & \frac{1}{5} & \dots & \frac{1}{2k+1} \\ \frac{1}{4} & \frac{1}{6} & \dots & \frac{1}{2k+2} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{1}{k+2} & \frac{1}{k+4} & \dots & \frac{1}{3k} \end{vmatrix}} \\
 &= 2^{-\binom{k}{2}} \left[ \prod_{i=1}^k \binom{k+2i}{i \ 2i \ k-i} \xi_i^2 \right] \left[ \prod_{i < j} [(\xi_i + \xi_j)(\xi_i - \xi_j)^2] \right],
 \end{aligned}$$

then

$$(5.3) \quad E\{I_f(\xi)\} = \int_{-1}^1 f(x) \, dx \quad \text{for any integrable } f.$$

Moreover, if  $0, \xi_1, \dots, \xi_k$  are distinct, then

$$(5.4) \quad I_f(\xi) = \int_{-1}^1 f(x) \, dx \quad \text{if } f \text{ is a polynomial of degree } 2k+1 \text{ or less.}$$

*Proof.* The two forms of the function in (5.2) can be shown to be equivalent with the help of formulae for the determinant of the Vandermonde and Cauchy matrices (e.g. Knuth, (1973, Vol. 1, p. 36, Probs. 37 and 38)). First we show that this function is a density. From the second form in (5.2) it is clearly nonnegative. Let  $V(x_1, \dots, x_k)$  denote the determinant of a general Vandermonde matrix, so that the integral of the numerator in (5.2) is

$$(5.5) \quad \int_0^1 \dots \int_0^1 V(\xi_1^2, \dots, \xi_k^2) \frac{V(\xi_1, \dots, \xi_k)}{\xi_1 \dots \xi_k} d\xi_1 \dots d\xi_k.$$

Let  $\theta$  denote a permutation of  $\{1, \dots, k\}$  and let  $(-1)^\theta$  be 1 or  $-1$  according to whether  $\theta$  is even or odd. Expanding the determinant on the left, (5.5) becomes

$$(5.6) \quad \sum_{\theta} \int_0^1 \dots \int_0^1 (-1)^\theta \left[ \prod_{i=1}^k \xi_{\theta(i)}^{2i-1} \right] V(\xi_1, \dots, \xi_k) d\xi_1 \dots d\xi_k;$$

changing to these permuted variables in the rest of the integrand, we find that all summands are equal and (5.6) simplifies to

$$(5.7) \quad k! \int_0^1 \dots \int_0^1 (\xi_1^1 \cdot \xi_2^3 \cdot \dots \cdot \xi_k^{2k-1}) V(\xi_1, \dots, \xi_k) d\xi_1 \dots d\xi_k.$$

Expanding the remaining determinant, combining terms and integrating the resulting monomials, (5.7) becomes

$$(5.8) \quad k! \sum_{\theta} \frac{(-1)^\theta}{[\theta(1)+2] \cdot [\theta(2)+4] \cdot \dots \cdot [\theta(k)+2k]}$$



which we recognize as the expansion of the denominator of (5.2), establishing (5.2) as a density because it integrates to 1.

To verify unbiasedness, using symmetry in (5.1) we have

$$(5.9) \quad E\{I_f(\xi)\} - 2f(0) = kE\{w_1(\xi)F(\xi_1)\}.$$

Using Cramér’s rule to obtain  $w_1(\xi)$  as a ratio of determinants, one of which cancels a term in the density when integrated, then rearranging the order of integration, (5.9) may be written as

$$(5.10) \quad k \int_0^1 F(\xi_1) \left\{ \int_0^1 \cdots \int_0^1 C \begin{vmatrix} \frac{1}{3} & \xi_2^2 & \cdots & \xi_k^2 \\ \vdots & \vdots & & \vdots \\ \frac{1}{2k+1} & \xi_2^{2k} & \cdots & \xi_k^{2k} \end{vmatrix} \cdot \begin{vmatrix} 1 & \xi_1 & \cdots & \xi_1^{k-1} \\ \vdots & \vdots & & \vdots \\ 1 & \xi_k & \cdots & \xi_k^{k-1} \end{vmatrix} d\xi_2 \cdots d\xi_k \right\} d\xi_1$$

where  $C$  is the constant term of the density. We now show that the bracketed  $(k - 1)$ -fold integral in (5.10), which is a polynomial in  $\xi_1$ , is in fact a constant. Consider a term in the expansion of the determinant on the right in which  $\xi_1$  occurs raised to a positive power. Because one entry from the leftmost column must also appear, some  $\xi_m$  is missing from each such term. Integrating this term first with respect to  $\xi_m$ , we see that

$$(5.11) \quad \int_0^1 \begin{vmatrix} \frac{1}{3} & \xi_2^2 & \cdots & \xi_k^2 \\ \vdots & \vdots & & \vdots \\ \frac{1}{2k+1} & \xi_2^{2k} & \cdots & \xi_k^{2k} \end{vmatrix} d\xi_m = \begin{vmatrix} \frac{1}{3} & \xi_2^2 & \cdots & \int_0^1 \xi_m^2 d\xi_m & \cdots & \xi_k^2 \\ \vdots & \vdots & & \vdots & & \vdots \\ \frac{1}{2k+1} & \xi_2^{2k} & \cdots & \int_0^1 \xi_m^{2k} d\xi_m & \cdots & \xi_k^{2k} \end{vmatrix}$$

is zero because columns 1 and  $m$  are identical. Thus, for some constant  $C'$ ,

$$(5.12) \quad E\{I_f(\xi)\} = C' \int_0^1 F(x) dx + 2f(0) = C' \int_{-1}^1 f(x) dx + 2(1 - C')f(0).$$

If we substitute the special case  $f(x) = x^2$  in (5.12), we find that  $C' = 1$ , which completes the proof of unbiasedness. □

**THEOREM 5.2.** *Let  $(\xi_1, \dots, \xi_k)$  have any distribution for which  $I_f(\xi)$  from (5.1) is an unbiased estimate of the integral of any integrable function  $f$ . If  $f$  is  $2k + 2$ -times differentiable, and  $M$  is the supremum of the magnitude of this derivative, then an upper bound on the variance of the estimate is given by*

$$(5.13) \quad \text{Var}\{I_f(\xi)\} \leq \frac{4M^2}{[(2k + 2)!]^2} E \left[ \sum_{i=1}^k w_i(\xi) \xi_i^{2k+2} \right]^2.$$

*Proof.* From Taylor’s theorem there exist  $\varepsilon_1$  and  $\varepsilon_2$  in  $[0, \xi]$  such that

$$(5.14) \quad I_f(\xi) = 2 \left[ f(0) + \frac{f''(0)}{3!} + \cdots + \frac{f^{(2k+2)}(0)}{(2k+1)!} \right] + \sum_{i=1}^k \frac{w_i(\xi) \xi_i^{2k+2}}{(2k+2)!} [f^{(2k+2)}(\varepsilon_1) + f^{(2k+2)}(-\varepsilon_2)].$$

Omitting the leading constants, which cannot affect a variance calculation, we see that

$$(5.15) \quad \text{Var} \{I_f(\xi)\} \leq E \left\{ \sum_{i=1}^k \frac{w_i(\xi) \xi_i^{2k+2}}{(2k+2)!} [f^{(2k+2)}(\varepsilon_1) + f^{(2k+2)}(-\varepsilon_2)] \right\}^2$$

from which (5.13) follows.  $\square$

**6. Two-dimensional integrals.** A real need for quadrature methods arises in the estimation of integrals in more than one variable. The simple solution of representing a multiple integral as a succession of iterated one-dimensional integrals can be too time-consuming for some practical problems.

In this section we present two designs for two-dimensional integration over a rectangle that are exact for bi-cubics (polynomials of joint degree at most three in two variables) and also unbiased for the integral of any function. One is the simple iterated solution based on the method of § 3, which requires nine function evaluations. The second design requires only five function evaluations, a substantial savings, noticeably reducing the amount of effort while retaining the same basic properties. For dimensionality greater than two, this five-point design might be applied to disjoint pairs of coordinates, with the one-dimensional method used on the remaining axis if the dimensionality is odd.

Because any integral over a rectangular region can be linearly transformed to an integral over a square, without loss of generality we will consider estimation of the integral

$$(6.1) \quad \int_{-1}^1 \int_{-1}^1 f(\xi, \eta) d\xi d\eta.$$

The iterated solution, generalizing the method of Theorem 3.1, is based on the design shown in Fig. 6.1, and may be written as follows:

$$(6.2) \quad \begin{aligned} I_f(\xi, \eta) = & \frac{1}{9\xi^2\eta^2} [f(\xi, \eta) + f(\xi, -\eta) + f(-\xi, \eta) + f(-\xi, -\eta) \\ & - 2(1 - 3\eta^2)[f(\xi, 0) + f(-\xi, 0)] \\ & - 2(1 - 3\xi^2)[f(0, \eta) + f(0, -\eta)] + 4(1 - 3\xi^2)(1 - 3\eta^2)f(0, 0)]. \end{aligned}$$

It is straightforward to verify by substitution and calculation that if  $\xi$  and  $\eta$  are independently and identically distributed according to the density  $3x^2$  on  $(0, 1)$ , then

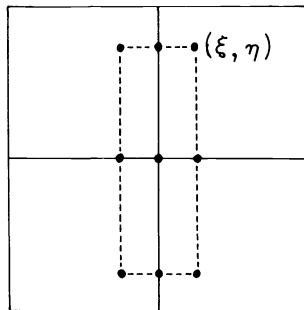


FIG. 6.1. The sampling pattern for the nine-point iterated solution of (6.2) that leads to exactness for polynomials of degree up to 3 jointly in two variables.

the estimate (6.2) gives the exact integral for all bicubic polynomials and also is unbiased for any integrable function. Generation of these random numbers is not difficult, as this is the density of the cube root of a uniformly distributed random variable.

The five-point design, which reduces the number of function evaluations nearly to half of those required above, is shown in Fig. 6.2. The integral estimate is given by

$$(6.3) \quad J_f(\xi, \eta) = \frac{f(\xi, \eta) + f(-\eta, \xi) + f(\eta, -\xi) + f(-\xi, -\eta) - 4f(0, 0)}{3(\xi^2 + \eta^2)/2} + 4f(0, 0)$$

and the density function for  $(\xi, \eta)$  is

$$(6.4) \quad \phi(\xi, \eta) = \frac{3}{2}(\xi^2 + \eta^2), \quad \xi \text{ and } \eta \text{ in } (0, 1).$$

It is again straightforward to verify exactness for bicubic polynomials and unbiasedness for any integrable function.

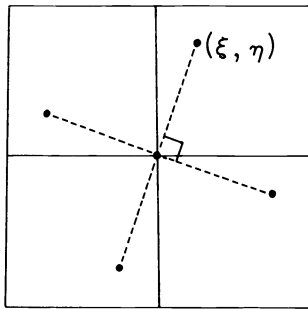


FIG. 6.2. The sampling pattern for the five-point solution of (6.3) that leads to exactness for polynomials of degree up to 3 jointly in two variables.

Moreover, the density (6.4) is not difficult to sample from. First choose  $\xi$  from its marginal density

$$(6.5) \quad \frac{1}{2} + \frac{3\xi^2}{2}$$

which we recognize as the mixture distribution which is half the time a uniform random variable, and the other half of the time the cube root of a uniform random variable. Conditionally on  $\xi$ , the density of  $\eta$  is

$$(6.6) \quad \frac{3\xi^2}{1+3\xi^2} + \frac{3\eta^2}{1+3\xi^2}$$

which we again recognize as a mixture. Thus with probability  $(3\xi^2)/(1+3\xi^2)$  we choose  $\eta$  from the uniform distribution, and with probability  $1/(1+3\xi^2)$  we choose  $\eta$  as the cube root of a uniform random variable.

Variance bounds for this five-point method are given in the following theorem. It should be emphasized that these bounds need not be calculated as part of the routine use of these methods; an assessment of the precision comes from the standard error obtained from repeated independent calculations. These bounds are presented instead in order to provide some intuition about the class of functions for which these methods work well.

**THEOREM 6.1.** *Let  $f(\xi, \eta)$  be a real integrable function defined on the square  $(-1, 1) \times (-1, 1)$ , and suppose that all the partial derivatives of  $f$  exist up to fourth order. Then the variance of the integral estimate in (6.3) when  $(\xi, \eta)$  is sampled from the density*

(6.4) cannot exceed

$$(6.7) \quad \begin{aligned} &.00112 M_1^2 + .01979 M_2^2 + .00988 M_3^2 + .00822 M_1 M_2 \\ &\quad + .02778 M_2 M_3 + .00617 M_1 M_3 \end{aligned}$$

where the constants are given by

$$(6.8) \quad \begin{aligned} M_1 &= \sup \left| \frac{\partial^4 f}{\partial \xi^4} \right| + \sup \left| \frac{\partial^4 f}{\partial \eta^4} \right|, \\ M_2 &= \sup \left| \frac{\partial^4 f}{\partial \xi^2 \partial \eta^2} \right|, \\ M_3 &= \sup \left| \frac{\partial^4 f}{\partial \xi \partial \eta^3} \right| + \sup \left| \frac{\partial^4 f}{\partial \xi^3 \partial \eta} \right|. \end{aligned}$$

*Proof.* Begin by considering  $(\xi, \eta)$  to be fixed and expand

$$(6.9) \quad g(\lambda) = f(\lambda\xi, \lambda\eta) + f(-\lambda\eta, \lambda\xi) + f(\lambda\eta, -\lambda\xi) + f(-\lambda\xi, -\lambda\eta)$$

in a Taylor series about zero with remainder term of fourth degree. From this we can show that

$$(6.10) \quad \begin{aligned} &\left| J_f(\xi, \eta) - 4f(0, 0) - \frac{2}{3} \left[ \frac{\partial^2 f}{\partial \xi^2}(0, 0) + \frac{\partial^2 f}{\partial \eta^2}(0, 0) \right] \right| \\ &\leq \frac{1}{18(\xi^2 + \eta^2)} [M_1(\xi^4 + \eta^4) + 12M_2\xi^2\eta^2 + 4M_3(\xi^3\eta + \xi\eta^3)]. \end{aligned}$$

The variance of the estimate is bounded above by the expected square of (6.10). Expanding the square of the right-hand side, multiplying by the density (6.4), integrating over the unit square  $(0, 1) \times (0, 1)$ , and simplifying, we find

$$(6.11) \quad \begin{aligned} \text{Var} \{J_f(\xi, \eta)\} &\leq \frac{1}{324} \left[ M_1^2 \iint \left[ \xi^6 + \eta^6 - \xi^4\eta^2 - \xi^2\eta^4 + \frac{4\xi^4\eta^4}{\xi^2 + \eta^2} \right] d\xi d\eta \right. \\ &\quad + 144M_2^2 \iint \frac{\xi^4\eta^4}{\xi^2 + \eta^2} d\xi d\eta \\ &\quad + 16M_3^2 \iint \xi^2\eta^2(\xi^2 + \eta^2) d\xi d\eta \\ &\quad + 24M_1M_2 \iint \left[ \xi^4\eta^2 + \xi^2\eta^4 - \frac{2\xi^4\eta^4}{\xi^2 + \eta^2} \right] d\xi d\eta \\ &\quad \left. + 32M_1M_3 \iint (\xi^5\eta + \xi\eta^5) d\xi d\eta + 96M_2M_3 \iint \xi^3\eta^3 d\xi d\eta \right]. \end{aligned}$$

Using a transformation to polar coordinates, the integral of the rational function is

$$(6.12) \quad \int_0^1 \int_0^1 \frac{\xi^4\eta^4}{\xi^2 + \eta^2} d\xi d\eta = \frac{1}{4} \int_0^{\pi/4} \tan^4(\theta) d\theta = \frac{\pi}{16} - \frac{1}{6}.$$

Using this in (6.11) and integrating the polynomials directly, (6.7) follows.

We now provide an example. The double integral on the left in (6.12) can be estimated using the very method (6.3 and 6.4) for which it provides variance bounds.

With 10,000 replications, we computed an estimate of .02965 with an estimated standard error of .000072 and an actual error (compared to  $\pi/16 - 1/6 = .02968 \dots$ ) of .00003. For comparison, simple Monte Carlo with 10,000 replications of a single point in the square provided an estimate of .03050 with an estimated standard error of .00064 and an actual error of  $-.00081$ .

These simulations were performed on an Intersystems Z80 based S100 microcomputer in Microsoft FORTRAN using a linear congruential random number generator with a 32-bit seed and the multiplier 16807. The ratio of estimated variances is  $(.00064)^2/ (.000072)^2$ , which is about 79. Discounting for the fact that simple Monte Carlo requires only one-fourth as many function evaluations, the estimated advantage is a factor of nearly 20. Thus to obtain comparable precision for this integral (6.12), simple Monte Carlo requires about 20 times as many function evaluations.

## REFERENCES

- K. L. BELL (1981), *Data modifications based on order: pushback; a configural polysampling approach*, Ph.D. thesis, Dept. Statistics, Princeton Univ., Princeton, NJ.
- K. L. BELL AND D. PREGIBON (1981), *Some computational details of configural sampling methods*, Tech. Rep. 191, Series 2, Dept. Statistics, Princeton Univ., Princeton, NJ.
- K. BOGUES, R. M. CORBETT AND T. N. L. PATTERSON (1981), *An implementation of the method of Ermakov and Zolotukhin for multidimensional integration and interpolation*, Numer. Math., 37, pp. 49–60.
- R. CRANLEY AND T. N. L. PATTERSON (1970), *A regression method for the Monte Carlo evaluation of multi-dimensional integrals*, Numer. Math., 16, pp. 58–72.
- (1976), *Randomization of number theoretic methods for multiple integration*, SIAM J. Numer. Anal., 13, pp. 904–914.
- S. M. ERMAKOV AND V. G. ZOLOTUKHIN (1960), *Polynomial approximations and the Monte Carlo method*, Teor. Veroyatnost. i Primenen 5, pp. 473–476; Theory Prob. Appl., 5, pp. 428–431.
- S. M. ERMAKOV (1964), *Random quadratures of improved accuracy*. U.S.S.R. Computational Mathematics and Mathematical Physics, 4 (No. 3), pp. 213–219.
- W. FELLER (1971), *An Introduction to Probability Theory and Its Applications*, second edition, volume II, John Wiley, New York.
- J. FRIEDMAN AND M. WRIGHT (1981), *A nested partitioning procedure for numerical multiple integration*, ACM Trans. Math. Software, 7, pp. 76–92.
- B. L. GRANOVSKII (1968), *Random quadratures of the Gaussian type*, U.S.S.R. Computational Mathematics and Mathematical Physics 8 (No. 4), pp. 244–252.
- S. HABER (1969), *Stochastic quadrature formulas*, Math. Comp., 23, pp. 751–764.
- (1970), *Numerical evaluation of multiple integrals*, SIAM Rev., 12, pp. 481–526.
- J. M. HAMMERSLEY AND D. C. HANDSCOMB (1964), *Monte Carlo Methods*, Methuen, London.
- J. M. HAMMERSLEY AND K. W. MORTON (1956), *A new Monte Carlo technique: antithetic variables*, Proc. Cambridge Philos. Soc., 52, pp. 449–475.
- D. C. HANDSCOMB (1964), *Remarks on a Monte Carlo method*, Numer. Math., 6, pp. 261–268.
- D. E. KNUTH (1973), *The Art of Computer Programming*, Volume 1, second edition, Addison-Wesley, Reading, MA.
- R. W. QUACKENBUSH (1969), *Monte Carlo quadrature with exactness for polynomials*, Ph.D. Thesis, Stevens Institute of Technology, Hoboken, NJ.
- J. W. TUKEY AND D. PREGIBON (1981), *Assessing the behavior of robust estimates of location in small samples 1: introduction to configural polysampling*, Technical Report No. 185, Series 2, Dept. Statistics, Princeton Univ., Princeton, NJ.

## ON THE FACTORIZATION OF BLOCK-TRIDIAGONALS WITHOUT STORAGE CONSTRAINTS\*

MARSHAL L. MERRIAM†

**Abstract.** In many programs solving difference equations, problem size is restricted by the number of available memory cells. A strategy has been developed to permit trade-offs between the number of floating point operations required and storage requirements for the solution of certain problems such as block tridiagonal systems of equations. This is done by recomputing some intermediate results instead of storing them. Reducing the storage to the square root of the current requirement will roughly double the number of computations. In theory, if  $m$  is the order of each sub-matrix in the block tridiagonal matrix, one can solve any linear system with only  $5m^2 + 1$  temporary storage cells. This method lends itself to efficient use on computers with parallel processing or vector processing architectures. On these computers the larger number of floating point operations is more than offset by the decrease in I/O and the increased percentage of vector operations made possible by this algorithm.

**Key words.** block tridiagonals, storage, Thomas algorithm, decomposition

**1. Introduction.** The most widely used algorithm for solving general systems of linear equations is Gaussian elimination. Other methods have appeared which take advantage of the structure of certain problems, i.e. cyclic reduction for banded matrices with constant coefficients. Most methods thus far devised have had the objective of reducing the total number of floating point operations. The method described here does not.

What one really wishes to minimize is the overall time and cost of solving a given problem. When computers were slow and problems were small, the way to do this was to minimize arithmetic. With the advent of supercomputers however, other considerations have become important.

One such consideration is the ability to vectorize an algorithm. For a given computer the speed of the vector hardware may exceed that of the scalar hardware by a factor of ten or more. A common way of finding long vectors in a tridiagonal solver is to solve many tridiagonals at once. The vector length then becomes the number of simultaneously solved systems. The storage requirements of such an approach exceed those of the scalar approach by a factor of the vector length.

Another consideration is the time spent in communication with secondary memory. The speed of the arithmetic units makes it possible to solve, in a reasonable amount of time, problems whose storage requirements exceed the capacity of primary memory. On most computers it is difficult to overlap the transfer time between primary and secondary memory. This becomes the dominant cost in some cases. In addition, programs which use secondary memory are often significantly more complex than those which do not. Furthermore transfers between memory levels are hardware dependent. Programs which do explicit transfers between memory levels are not portable for this reason. Realizing this, we turn our efforts towards an algorithm that requires less memory, even if it requires more arithmetic.

Recomputation is such an algorithm. It offers the user a trade-off between the number of arithmetic operations, time spent in scalar computations and time spent on data transfers to secondary memories. A Fortran subroutine has been written which utilizes recomputations in the solution of block tridiagonal systems. The user can

---

\* Received by the editors August 31, 1982, and in revised form October 11, 1983.

† NASA Ames Research Center, Moffett Field, California 94035.



The storage problems with this method stem from the fact that one must compute all of the elements  $c'$  and  $q'$  before any of the elements  $q$  can be computed. The right-hand side is usually overwritten with the solution so that the same storage cell is occupied at various times by  $r_i$ ,  $q'_i$ , and finally  $q_i$ . Traditionally, both the  $c'$  matrices and the right-hand side are stored for a total of  $(n-1)m^2 + nm$  storage cells.

Notice that to compute  $c'_{i+1}$  we only need  $b'_i$  and  $c_{i+1}$ . Also, to compute  $b'_{i+1}$  we need only  $c'_{i+1}$  and the matrix elements  $a_i$  and  $b_{i+1}$ . Schematically this is shown in Fig. 1. The dashed boxes contain the original matrix entries. In many applications these

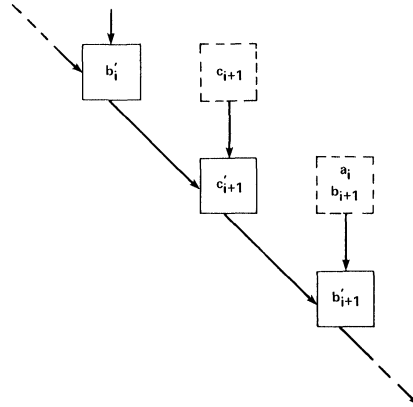


FIG. 1. Data dependencies in the Thomas algorithm.

require no storage since they are either analytically known or can be recovered from other information contained in memory. We consider such applications here. The main consequence of this simplification is that if any element of the decomposition is known (i.e.  $b'$  or  $c'$ ), then the forward elimination can be reinitiated at that point to get any subsequent decomposition element. This is the basis for the whole scheme. We save a few, selected, elements  $c'$  on the forward elimination and then execute the following sequence:

- a. Execute the backward substitution in a conventional manner as far as possible.
- b. When an element  $c'_i$  is needed and not available, recompute it by re-initiating the forward elimination starting with a stored element  $c'$ . The best choice is that element whose index is highest without exceeding  $i$ . All elements  $c'$  with higher indexes are no longer useful and may be overwritten to save other indexes of  $c'$ . If no stored data remains, recompute from index 1. The element  $c'_1$  is zero. When the needed element has been recomputed resume step a.

Notice that the forward elimination in step b is analogous to the original forward elimination. The starting and ending indexes are different as is the amount of available storage but the form is the same. Thus we may use the same selection process to decide which elements to save in step b that we did on the original forward elimination. What follows is a description of and rationale for one such selection process.

**3. Selection algorithm—rationale.** Let  $t$  be the number of unknowns. That is we wish to solve for  $c'_{t_0+t} \cdots c'_i \cdots c'_{t_0+1}$  in that order, starting from  $c'_{t_0}$ .

Let  $s$  be the number of storage cells. That is we can store in locations  $C'_{s_0+1} \cdots C'_k \cdots C'_{s_0+s}$ .

Let  $p_i$  be the number of times  $c'_i$  is computed.

Let  $P(t, s)$  be the largest value of  $p_i$  for  $t_0 < i \leq t_0 + t$ .  $P$  is said to be the degree of the problem.



Let  $L(P, s)$  be the largest value of  $t$  possible given  $P$  and  $s$ . A problem is said to be full and of degree  $P$  if  $t = L(P, s)$ .

It can be shown that a full problem can be split into  $s$  full subproblems of degree  $P-1$  using only  $s-1$  storage locations. The last storage location is used to solve for  $c'_{t_0+t}$ . The process of solving for  $c'_{t_0+t}$  is considered to be part of the forward sweep and is not counted as part of a subproblem.

The strategy for choosing which elements to save is as follows:

1. Find  $P(t, s)$ . This is the smallest value of  $P$  for which  $L(P, s) \geq t$ .
2. Split the problem into  $s$  subproblems. We refer to the one containing  $c'_{t_0+t-1}$  as the highest subproblem and the one containing  $c'_{t_0+1}$  as the lowest. Since  $s-1$  storage locations are required to split the problem into subproblems, there will only be one location left to use in solving the highest subproblem. Consequently the highest subproblem will have at most  $L(P-1, 1)$  unknowns. The second highest will have at most  $L(P-1, 2)$  unknowns and so forth until the lowest subproblem contains at most  $L(P-1, s)$  unknowns.
3. A lower bound on the number of unknowns in each subproblem can be given by decrementing the degree of each by one. That is, the highest subproblem must have at least  $L(P-2, 1)$  unknowns, the second highest must have at least  $L(P-2, 2)$ , and so forth. (Note that  $L(0, s) = 0$ .) When the degree of each subproblem is decremented to  $P-2$ , what is left is a full problem of degree  $P-1$ .
4. As described here the selection algorithm uses only subproblems of degree  $P-1$  or  $P-2$ . At most one nonfull subproblem will be used. The following rules are used in choosing the degree and size of the subproblems.
  - a. All subproblems of degree  $P-2$  are higher than any subproblems of degree  $P-1$ .
  - b. The highest subproblem of degree  $P-1$  need not necessarily be full. If the subproblems are numbered from lowest to highest, it is numbered  $k'$ .

**4. Selection algorithm—construction.** Now that we know what we want the selection algorithm to do, we have to show how to construct it. The following recipe will work. Let  $H_k$  be the collection of saved indexes. That is  $c'_{H_k}$  is saved in  $C'_k$ .

1. Initially  $H_k = t_0$  for all  $s_0 \leq k \leq s_0 + s$ . This is a full subproblem for  $P = 0$ .
2. We now guess  $P$  by guessing all possible values in order; that is  $P = 1, 2, \dots, t$ .
3. For each  $P$  we guess  $k'$  by guessing all possible values in order; that is  $k' = 1, 2, \dots, s$ .
4. At this point we add more unknowns to one of the subproblems. There are two cases.
  - a. If  $t_0 + t - H_s \leq H_s - H_{k'} + 1$ , then we have arrived at the correct guess. We wish to put more unknowns in subproblem  $k'$  so that it is of degree  $P-1$  instead of  $P-2$ . The correct action is to add  $t_0 + t - H_s$  to  $H_k$  for  $k' \leq k \leq s$  and stop.
  - b. If  $t_0 + t - H_s > H_s - H_{k'} + 1$ , then we have not yet arrived at the correct guess. We wish to increase the number of unknowns in subproblem  $k'$  so that it is a full subproblem of degree  $P-1$ . The correct action is to add  $H_s - H_{k'} + 1$  to  $H_k$  for  $k' \leq k \leq s$ . In programming make sure to evaluate  $H_s - H_{k'}$  before performing any additions since  $H_s$  and  $H_{k'}$  are affected.
5. Go back to step 3 unless all of the  $k'$  values have been tried.
6. Go back to step 2 unless all of the  $P$  values have been tried.
7. If you get here something is wrong. All possible choices have been tried and one of them must be right.

The flowchart in Fig. 2 illustrates the above process.

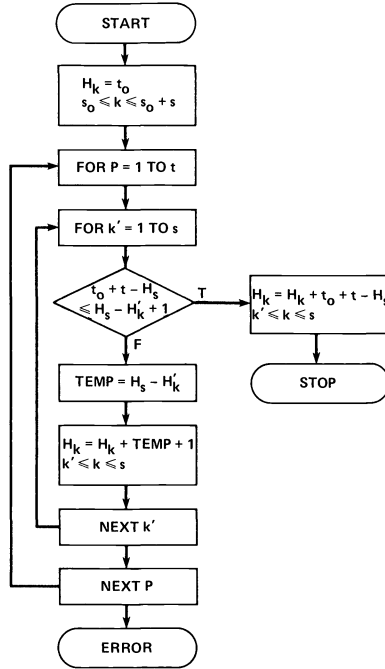


FIG. 2. Detail of the selection algorithm.

*Example.* Suppose  $n = 11$  and there is only room to store 3 temporaries. The conventional Thomas algorithm requires 10 temporaries. The selection algorithm would proceed as follows using  $t_0 = 1, t = 10, s = 3$ :

1. Initially  $H_k = 1$  for  $1 \leq k \leq 3$ . At this point

$$H_1 = 1, \quad H_2 = 1, \quad H_3 = 1.$$

2. Guess  $P = 1, k' = 1$  and check on the inequality  $t_0 + t - H_s \leq H_s - H_{k'} + 1$ . Since  $11 - 1 > 1 - 1 + 1$ , we have not yet arrived at the correct guess. Since  $H_s - H_{k'} + 1 = 1$ , we add 1 to each  $H_k$  for  $1 \leq k \leq 3$ .

$$H_1 = 2, \quad H_2 = 2, \quad H_3 = 2.$$

3. Guess  $P = 1, k' = 2$  and check on the inequality  $t_0 + t - H_s \leq H_s - H_{k'} + 1$ . Since  $11 - 2 > 2 - 2 + 1$ , we have not yet arrived at the correct guess. Since  $H_s - H_{k'} + 1 = 1$ , we add 1 to each  $H_k$  for  $2 \leq k \leq 3$ .

$$H_1 = 2, \quad H_2 = 3, \quad H_3 = 3.$$

4. Guess  $P = 1, k' = 3$  and check on the inequality  $t_0 + t - H_s \leq H_s - H_{k'} + 1$ . Since  $11 - 3 > 3 - 3 + 1$ , we have not yet arrived at the correct guess. Since  $H_s - H_{k'} + 1 = 1$ , we add 1 to each  $H_k$  for  $3 \leq k \leq 3$ . At this point

$$H_1 = 2, \quad H_2 = 3, \quad H_3 = 4.$$

5. Guess  $P = 2, k' = 1$  and check on the inequality  $t_0 + t - H_s \leq H_s - H_{k'} + 1$ . Since  $11 - 4 > 4 - 2 + 1$ , we have not yet arrived at the correct guess. Since  $H_s - H_{k'} + 1 = 3$ , we add 3 to each  $H_k$  for  $1 \leq k \leq 3$ . At this point

$$H_1 = 5, \quad H_2 = 6, \quad H_3 = 7.$$

6. Guess  $P=2, k'=2$  and check on the inequality  $t_0+t-H_s \leq H_s-H_{k'}+1$ . Since  $11-7 > 7-6+1$ , we have not yet arrived at the correct guess. Since  $H_s-H_{k'}+1=2$ , we add 2 to each  $H_k$  for  $2 \leq k \leq 3$ . At this point

$$H_1=5, \quad H_2=8, \quad H_3=9.$$

7. Guess  $P=2, k'=3$  and check on the inequality  $t_0+t-H_s \leq H_s-H_{k'}+1$ . Since  $11-9 > 9-9+1$ , we have not yet arrived at the correct guess. Since  $H_s-H_{k'}+1=1$ , we add 1 to each  $H_k$  for  $3 \leq k \leq 3$ . At this point

$$H_1=5, \quad H_2=8, \quad H_3=10.$$

8. Guess  $P=3, k'=1$  and check on the inequality  $t_0+t-H_s \leq H_s-H_{k'}+1$ . Since  $11-10 < 10-5+1$ , we have arrived at the correct guess. Since  $t_0+t-H_s=1$ , we add 1 to each  $H_k$  for  $1 \leq k \leq 3$ . The final selections are

$$H_1=6, \quad H_2=9, \quad H_3=11.$$

This says we should save the elements  $c'_6, c'_9$ , and  $c'_{11}$  on the forward elimination. The computing algorithm would proceed as follows:

1. Initial forward sweep. Save  $c'_6, c'_9$ , and  $c'_{11}$ .
2. Backward sweep. Compute  $q_{11}$  and  $q_{10}$ . To compute  $q_9$  we require  $c'_{10}$ .
3. Since  $q_{10}$  is already computed  $c'_{11}$  is not needed. Resume forward sweep using  $c'_9$  and overwrite  $c'_{11}$  with  $c'_{10}$ .
4. Continue the backward sweep, using  $c'_{10}$  and  $c'_9$  to compute  $q_9$  and  $q_8$ .
5. Resume forward sweep with  $c'_6$ , overwriting  $c'_9$  and  $c'_{10}$  with  $c'_7$  and  $c'_8$ .
6. Continue the backward sweep, computing  $q_7, q_6$ , and  $q_5$ .
7. Resume forward sweep from index 1, overwriting  $c'_6, c'_7$ , and  $c'_8$  with  $c'_3, c'_4$ , and  $c'_5$ .
8. Continue backward sweep by computing  $q_4, q_3$ , and  $q_2$ .
9. Again resume forward sweep from index 1, overwriting  $c'_3$  with  $c'_2$ .
10. Conclude the backward sweep by computing  $q_1$ .

Each forward sweep except the last used all the storage that contained elements that were no longer needed. Temporaries  $c'_6, c'_9$ , and  $c'_{11}$  were computed only once. Temporary  $c'_2$  was computed three times. All the rest were computed twice. The total cost was almost twice that of the conventional Thomas algorithm yet the required storage was less than the square root of that required by the conventional algorithm. In larger problems it is often possible to reduce the storage requirements by a factor of ten while only doubling the arithmetic, a paying proposition if I/O is expensive. It is interesting to note that the minimum required storage space is five blocks, each an  $m \times m$  matrix, plus one word. This is extremely expensive, however, since the computational effort is higher than the nonrecomputing case by a factor of roughly  $n^2/2$ . Three of the blocks are needed for the elements  $a, b$ , and  $c$ . One is needed for the intermediate  $b'$  and the last is needed for the temporary  $c'$ . One additional word is needed for the pointer  $H_1$  to keep track of the one temporary. The flowchart in Fig. 3 illustrates how the selection process and the recomputation algorithm fit into the Thomas algorithm.

**5. Optimality.** One can always raise the question of whether the selection algorithm is optimum, i.e. does it always choose all of the  $H_k$  such that work is minimized given  $t$  and  $s$ ? In answering this I am indebted to my anonymous reviewer in providing the following optimal strategy:

Let  $F'(t, s)$  be the minimum number of computations necessary to compute  $c'_{t_0+t} \cdots c'_{t_0+1}$  from  $c'_{t_0}$  with  $s$  storage locations. Here, one computation is the amount of work necessary to advance one index of the forward sweep. Work done on the right hand side does not count. Then we can write down the recursive relation:

$$(5a) \quad F'(t, s) = \min_{1 \leq i \leq t} [i + F'(t-i, s-1) + F'(i-1, s)].$$

The first term,  $i$  is the cost of computing  $c'_{t_0+1} \cdots c'_{t_0+i}$ . The second term is the minimum cost of computing  $c'_{t_0+t} \cdots c'_{t_0+i+1}$  in the remaining space. The third index is the minimum cost of recomputing  $c'_{t_0+i-1} \cdots c'_{t_0+1}$ . Since all possible choices are tried for  $i$  and implicitly all choices are tried for saved indices in the second and third terms, the equality holds. The recursion is saved from being infinite by the relations

$$(5b) \quad F'(1, s) = 1,$$

$$(5c) \quad F'(0, s) = 0,$$

$$(5d) \quad F'(t, 1) = t(t+1)/2.$$

While this scheme is optimal, it is expensive to implement, costing  $O(st^2)$  operations. The selection algorithm suggested in the previous section costs  $O(t)$  operations and makes an optimal selection in all cases where  $s \leq t \leq 100$ . We conjecture that it always makes an optimal selection.

**6. Discussion.** The standard Thomas algorithm has the following operation count.

$$(6) \quad \begin{array}{ll} \text{multiplications} & n(\frac{7}{3}m^3 + 3m^2 - \frac{1}{3}m) - 2m^2(m+1), \\ \text{additions} & n(\frac{7}{3}m^3 + \frac{3}{2}m^2 - \frac{5}{6}m) - 2m^2(m+1), \\ \text{divisions} & nm. \end{array}$$

Here  $n$  is the number of block unknowns and  $m$  is the dimension of each block. In recomputing, only the factorization of the matrix is done more than once. If  $n \gg 1$  the operation count just for factoring the matrix is:

$$(6a) \quad \begin{array}{ll} \text{multiplications} & n(\frac{7}{3}m^3 - \frac{1}{3}m), \\ \text{additions} & n(\frac{7}{3}m^3 - \frac{3}{2}m^2 + \frac{1}{6}m), \\ \text{divisions} & nm. \end{array}$$

Using as a measure of relative cpu time the equivalency formula

$$(7) \quad 1 \text{ add} = 1 \text{ multiply} = \frac{1}{4} \text{ divide},$$

the total number of operations becomes

$$(8) \quad n(\frac{14}{3}m^3 - \frac{3}{2}m^2 + \frac{23}{6}m).$$

The variable  $F(t, s)$  is defined as the total number of operations involved in factorization divided by the quantity in brackets. In this way the dependence on  $m$  of the results is largely removed. The variable  $s$ , defined above, is a measure of the available storage. Finally  $n$ , also defined above, is a measure of the problem size. Figure 3 plots  $F$  vs.  $t$  for various values of the parameter  $s$ . This figure describes the common situation in which a computer has a limited total memory. This occurs when secondary memory is very much slower than primary memory, making its use impractical for solving linear systems. Often the secondary memory is a disk or a standard tape drive. In the case

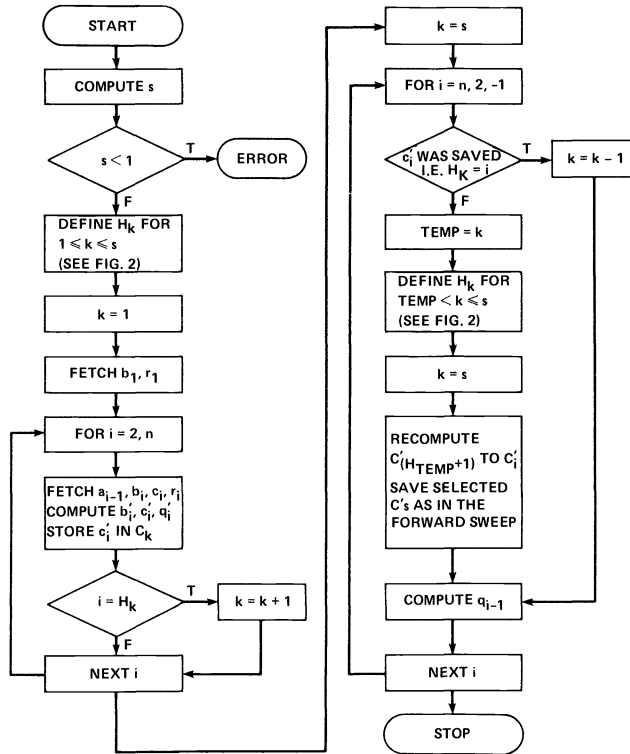


FIG. 3. The Thomas algorithm modified for recomputing.

of the current generation of micro-computers it may even be a cassette. In such a case recomputing may allow solution of problems which otherwise could not be solved at all. Figure 4 gives, at a glance, the cost of solving block tridiagonal systems as function of problem size given a fixed amount of memory.

Another situation for which recomputing can be helpful is where a program has been written, the problem size is fixed, and the user wishes to modify the program in

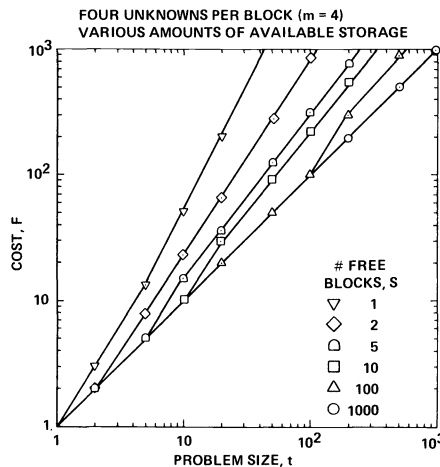


FIG. 4. Tradeoffs between CPU time and storage.

some way that requires more memory than the computer has. One way to get more memory is to reduce the amount of space allocated for temporaries used in solving block tridiagonals. Depending on the problem this may free a significant amount of storage. In this way the user may avoid a complete rewrite which might otherwise be necessary to incorporate transfers to secondary memory. Depending on the accounting algorithm for the computer in question, recomputing may even be cheaper than transfers to secondary memory. Experience has shown, however, that recomputing rarely pays on a cost/run basis if any element  $c'$  is computed more than twice.

A situation sometimes arises in which the total computer time is fixed. From this constraint one may estimate the maximum number of times each element may be computed. We call this number  $P$ . Given  $P$  and the storage constraint  $s$  there is a limit to the number of block unknowns we can solve for. We call this number  $L$ . The question arises: What is the relationship between  $P$ ,  $s$ , and  $L$ ? Such information could be useful in deciding on a vector length or deciding if this algorithm would pay at all. It can be shown that the recursion relation for finding  $L(P, s)$  is

$$(9a) \quad L(P, 1) = P,$$

$$(9b) \quad L(1, s) = s,$$

$$(9c) \quad L(P, s) = L(P, s-1) + L(P-1, s) + 1.$$

We notice immediately that  $L(P, s) = L(s, P)$ , that is the function  $L$  is symmetric about the line  $P = s$ . Also, along a line where  $P$  (or  $s$ ) is a constant the values of  $L$  may be exactly fitted by a polynomial of degree  $P$  (or  $s$ ). The first few and the general case are given here.

$$(10a) \quad P = 1 \quad L = s,$$

$$(10b) \quad P = 2 \quad L = \frac{3}{2}s + \frac{1}{2}s^2,$$

$$(10c) \quad P = 3 \quad L = \frac{11}{6}s + s^2 + \frac{1}{6}s^3,$$

$$(10d) \quad P = z \quad L = \sum_{j=1}^{z+1} (|S_{z+1}^{(j)}| s^{j-1} / z!) - 1.$$

In the general case,  $S_{z+1}^{(j)}$  are Stirling numbers of the first kind.

Thus we see that the highest order term is always  $s^z/z!$ . Equation (10d) is given without proof. In principle, however, one could substitute (10d) into (9c) to prove the equality.

The recomputation algorithm is arithmetically the same as the Thomas algorithm; hence it has exactly the same stability properties and gives exactly the same answer. Although the storage overhead is usually negligible, it does require  $s$  scalar temporaries. This should be compared with  $sm^2$  temporaries used per tridiagonal in the rest of the computation or with  $(n-1)m^2$  temporaries needed per tridiagonal for the Thomas algorithm. The computational overhead is equally negligible, involving only a small amount of integer arithmetic. If no recomputation is done, the recomputation algorithm costs virtually the same to use as a conventional Thomas algorithm.

**7. Conclusions.** It has proved useful to program the entire block tridiagonal solver as a subroutine which has, as an argument, the amount of available space. This substantially reduces the consequences of programming at the limit of primary memory. This alone helps increase productivity through reducing the number of times programs are rewritten to free a tiny amount of storage.

Using recomputation, problem size can be substantially increased on computers where memory size is poorly matched to processor speed for this type of problem. Recomputation can free enough memory to allow effective use of vector processing capabilities on machines like the Cray-1 and the CDC 205. Furthermore, the extra computation required is largely made up of dot products, at which these machines are very efficient. This algorithm was used on Illiac IV codes at Ames Research Center [2] from 1977 until the Illiac was replaced in 1981. The required vector length of 64 and the small size of primary memory made recomputation a virtual necessity on the Illiac, allowing the solution of problems which otherwise could not have been solved.

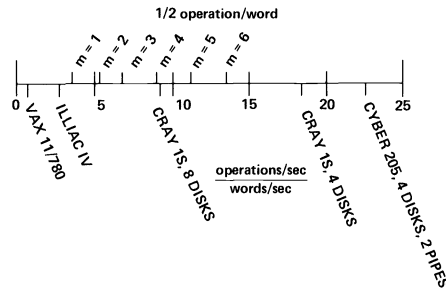
Surprisingly, execution speed can actually be increased through the use of recomputation. This can occur when disk latency and data transfer become a substantial portion of the code's running time. It can also occur when recomputation is used to increase vector lengths. In both cases costs can be reduced by doing more arithmetic, keeping the job in core using recomputation. Any time a situation arises where it costs more to bring a problem in and out of core than it does to perform the arithmetic, recomputation will pay. To quantify this, recomputation pays if

$$(11a) \quad \frac{\text{number of operations/block}}{\text{number of operations/second}} \stackrel{!}{=} \frac{\text{number of words/block}}{\text{number of words/second} \times 2}$$

or equivalently if

$$(11b) \quad \frac{\text{number of operations/block}}{\text{number of words/block}} \stackrel{!}{=} \frac{\text{number of operations/second}}{\text{number of words/second} \times 2}$$

The factor of two comes from the fact that each word must be written once and read once. The per block quantities are dependent upon  $m$  while the per second quantities are machine dependent. In Fig. 5 this inequality is depicted graphically. The I/O and CPU capabilities of several common computers are given for perspective [3], [4], [5].



IF THE DATA POINT FOR THE COMPUTER IS TO THE RIGHT OF THE DATA POINT FOR THE PROBLEM, RECOMPUTATION PAYS

FIG. 5. Feasibility of recomputing.

The arithmetic is assumed to be done at one fourth the maximum rate for single precision arithmetic. The I/O is assumed to take place at the maximum rate. A significant disk access time has the effect of lowering the effective transfer rate. Longer vector lengths made possible by recomputing will raise the effective CPU speed. Both of these make recomputing look more attractive. Neither is included in Fig. 5.

Of course, this general approach is not limited to tridiagonal matrices. It can easily be extended to cover periodic and pentadiagonal matrices. For that matter it can be

used to solve dense or wide banded matrices. It is not limited to Gaussian elimination but is applicable to any method with a forward and backward sweep which saves intermediate results. There is an interesting variant of this algorithm due to Eisenstat et al. [6] in which, by computing from both ends at once, it is possible to solve the system using only  $6m^2$  storage locations and approximately  $(n+2)m^3 \log_2 n$  operations. By comparison the method advocated here would need about  $nm^3\sqrt{n}$  operations for the same amount of storage. In practice the crossover point is at about  $n = 50$ . Eisenstat points out that if the matrix is further specialized so that the blocks  $a$ ,  $b$ , and  $c$  are diagonal, tridiagonal and diagonal respectively, then reordering the unknowns simplifies the problem further so that the work is only  $\frac{5}{3}$  that of normal Gaussian elimination. A combination of the two algorithms may produce an algorithm better than either of them. This is being studied.

Bigger and faster memories may temporarily reduce the need for recomputation but cannot remove its advantages. As long as there are substantial differences in speed between memory hierarchies, computers which do not match memory to processor speeds, computers with vector speeds significantly quicker than scalar speeds, or problems which are computationally light relative to their size, there will be a need for recomputation schemes.

**Acknowledgments.** The author wishes to thank the reviewers for their helpful comments. In particular the dynamic programming technique in the section on optimality and the algorithm of Eisenstat et al. were brought to my attention through them.

#### REFERENCES

- [1] W. F. AMES, *Mathematics in Science and Engineering*, 18, Academic Press, New York, 1965, pp. 341–342.
- [2] J. KIM AND P. MOIN, *Large eddy simulation of turbulent channel flow—Illiac IV calculation*, Proc. AGARD Symposium on Turbulent Boundary Layers, Experiment, Theory, and Modelling, The Hague, The Netherlands, Sept. 24–26, 1979; also, NASA Technical Memorandum 78619, Sept. 1979.
- [3] *Cray-OS Version 1 Reference Manual*, SR-0011, Cray Research Inc., Mendota Heights, MN 1980.
- [4] *Software and Hardware Product Descriptions, Product Resource Manual*, CDC, Control Data Corporation, St. Paul, MN, 1982.
- [5] *Peripherals Handbook*, Digital Equipment Corporation, Maynard, MA, 1981, p. 79.
- [6] S. C. EISENSTAT, M. H. SCHULTZ AND A. H. SHERMAN, *Minimal storage band elimination*, in *High Speed Computer and Algorithm Organization*, Kuck, Lawrie and Sameh, eds., Academic Press, New York, 1977, pp. 273–286.



## THREE NEW RAPIDLY CONVERGENT ALGORITHMS FOR FINDING A ZERO OF A FUNCTION\*

D. LE†

**Abstract.** Three new algorithms using only function evaluations for numerical solution of nonlinear equations are described. These methods feature the combination of bisection with interpolation. They guarantee convergence in a small number of function evaluations (1.7 or three times the number required by bisection), and their rates of convergence are comparable with those of existing methods.

**Key words.** zero finding, quadratic interpolation, linear interpolation, nonlinear equation

**1. Introduction.** One of the most basic problems in scientific work is to find a zero  $\alpha$  of a nonlinear function  $f(x)$  of a single variable, i.e., solving a nonlinear equation  $f(x) = 0$ . Since  $\alpha$  cannot in general be expressed in closed form, iterative methods are usually sought to produce an approximation solution to the problem. Many classical algorithms are known: Newton–Raphson, bisection, *regula falsi*, etc. For a good survey, see Traub (1964) or Ralston and Rabinowitz (1978). These methods are either too slow or unsafe to use without requiring certain conditions to be satisfied. Recently, many algorithms have been proposed which guarantee global convergence as well as having high asymptotic order of convergence. One such pioneer algorithm, published by Dekker (1969), uses a mixture of linear interpolation and bisection. Another algorithm is that of Dowell and Jarratt (1971) which is the modification of the *regula falsi* method; their method was again modified by the same authors in 1972 to give better order of convergence. Another higher order version of the *regula falsi* method is the one published by Anderson and Bjorck (1973). Although those above algorithms, namely Dekker’s, Dowell and Jarratt’s and Anderson and Bjorck’s, can guarantee convergence, they may require a prohibitively large number of function evaluations for certain classes of functions. In 1971 Brent proposed an algorithm similar to that of Dekker, but forcing a bisection if successive secant or inverse quadratic interpolation iterations were converging too slowly. Brent’s algorithm has an upper bound on the number of function evaluations of  $(n_b + 1)^2 - 2$ , where  $n_b$  is the number of function evaluations needed by bisection. Dekker’s algorithm is also the starting point for two other algorithms published by Bus and Dekker (1975) which have good asymptotic behaviour and require only  $4n_b$  or  $5n_b$  evaluations at most. In 1977, Gonnet proposed a hybrid method that uses quadratic interpolation during the first 30 iterations or so and then switches to the bisection method. A distinct disadvantage of Gonnet’s algorithm is its inability to realise difficulties before exhausting 30 iterations and, after that point of no return, a very slow convergent method is used regardless of how easy the situation might become. During early iterations, the interval is often very large and thus may contain irregularities that make a superlinear convergent method alone (like quadratic interpolation) very inefficient; the bisection method should be used in this case to bring the interval to the root’s vicinity so that a superlinear convergent method can become useful. Function 2 with a simple root, as listed in § 5, would cause

---

\* Received by the editors May 11, 1982, and in revised form April 15, 1983.

† Mechanical and Industrial Engineering Department, University of New South Wales, Kensington, New South Wales, Australia. Present address, Energy Systems Analysis Group, CSIRO Division of Energy Technology, Lucas Heights Research Laboratories, Private Mail Bag 7, Sutherland, 2237, New South Wales, Australia.

such a typical failure for Gonnet's algorithm. While the new algorithm LZ1 (described in § 2) needs only 12 function evaluations to solve this problem, Gonnet's algorithm would move extremely slowly during the first 30 iterations, leaving bisection to finalise the search and, if the search interval is large enough, the total limit of 80 iterations, as fixed by Gonnet, may be insufficient for convergence. Although Gonnet's algorithm has produced some notable performances, as presented in his paper, it will not be considered any further here.

In §§ 2 and 3, two new algorithms (algorithms LZ1 and LZ2) are presented which use bisection and quadratic approximation and which have upper bounds on the total number of function evaluations needed of  $1.7n_b$  and  $3n_b$  respectively. Section 4 describes the third algorithm (algorithm LZ3) which utilises only linear approximation and bisection, but which also has an upper bound of  $3n_b$ . Section 5 shows some numerical results of these algorithms.

**2. Algorithm LZ1.** For a real function  $f$ , defined on the interval  $[a, b]$  with  $f(a) \cdot f(b) \leq 0$ , the three new algorithms LZ1, LZ2 and LZ3 will locate an approximation  $\hat{\alpha}$  to a zero  $\alpha$  of  $f$  to within the required precision by using only function evaluations. It should be noted that  $f$  need not be continuous on  $[a, b]$  and thus there may be no zero in  $[a, b]$ , but in this case LZ1, LZ2 and LZ3 still produce a small interval of  $2\delta_0$  ( $\delta_0$  is defined later), within which  $f$  takes both negative and positive values. All three algorithms require as their input the function  $f$ , the interval  $[a, b]$  and  $\epsilon_y$  which is the minimum required accuracy of the objective function values. Depending on the context as well as on the working precision, the user must carefully choose the limiting residual  $\epsilon_y$ , especially if  $|f'(x)|$  is very small at the root so that  $|f(x)|$  increases slowly as  $x$  moves away from the root. For most practical purposes, it is felt that the function value is more important than the position of the argument and thus the objective function value accuracy is sought by LZ1, LZ2 and LZ3 rather than the accuracy of the independent variable. One would usually accept the root estimate returned by a root solver routine if the value of the objective function at the estimate is near zero (as specified by the user's  $\epsilon_y$ ), although the final argument interval is still quite large. But the reverse can be unacceptable and, furthermore, the accuracy of the argument required for an acceptable objective function value is not usually known beforehand. Consequently, an interval estimate must always be close to machine accuracy to avoid the problem of stopping prematurely and producing an unreasonably large objective function value. Furthermore, because of the essential part played by  $\epsilon_y$  in association with cushion interpolation (for example, see step 1.4 of the algorithm LZ1), it is not advisable to use the argument interval as a stopping criterion in our algorithms, particularly LZ1. The problem of having to base the stopping rule on the objective function value also exists for many other well-known methods, notably Newton-Raphson and *regula falsi*. Therefore, LZ1, LZ2 and LZ3 only use the bracketing interval mainly as a mechanism to protect against divergence and not as a stopping criterion.

LZ1 combines bisection with a technique which is termed cushion interpolation to reduce the search interval and, whenever  $f$  can be reasonably well approximated by a quadratic function, then direct interpolation is used instead to estimate  $\alpha$ . It should be noted that the use of interpolation estimate as the new point for function evaluation is termed direct interpolation to distinguish from cushion interpolation, which will be defined later. In LZ1 and LZ2, quadratic interpolation is preferred over the use of inverse quadratic interpolation or rational interpolation since the two latter methods can produce estimates outside bounds and sometimes can even be inapplicable,

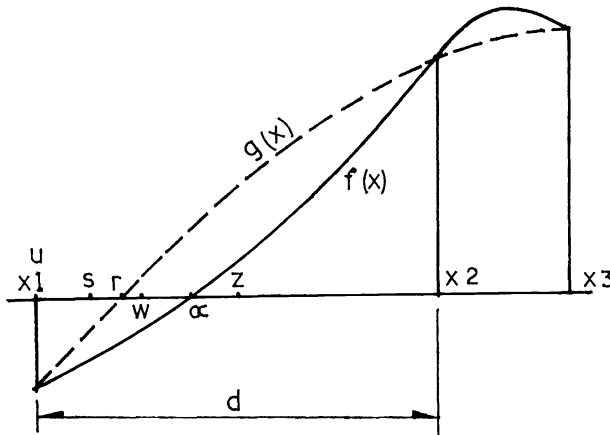


FIG. 1. Typical configuration of LZ1.

while quadratic interpolation will always produce an estimate within bounds assuming no round-off errors.

The main features of the algorithm are described below.

At the beginning of a typical iteration of the algorithm LZ1 (Fig. 1), three distinct points  $x_1$ ,  $x_2$  and  $x_3$  are available such that

$$(1) \quad f(x_1) \cdot f(x_2) \leq 0, \quad x_2 \in [x_1, x_3], \quad f(x_2) \cdot f(x_3) \geq 0, \quad f(x_3) \neq 0.$$

The first condition ensures that there exists a zero  $\alpha$  of  $f$  in the closed interval  $[x_1, x_2]$  while the other conditions state that  $x_2$  and  $x_3$  lie on the same side of  $\alpha$ .

A typical iteration of LZ1 consists of the following steps:

1.1. Let  $z = x_1 + (x_2 - x_1)/2$  and  $s$  be the last value of  $r$  ( $r$  is defined in 1.2); also let  $d = |x_1 - x_2|$  and  $e$  be the last value of  $d$ . If  $|f(x_2)| < |f(x_1)|$ , let  $u = x_2$ , otherwise  $u = x_1$  (this condition yields that  $u$  is the current best approximation of  $\alpha$ ). Let  $\delta_0 = \epsilon(1 + |u|)$  and  $\delta = \max(d/10, \delta_0)$  where  $\epsilon$  is a machine-dependent number and chosen to be  $10^{-14}$  in the numerical study shown in § 5. (In fact, the tolerance function  $\delta_0$  should ideally be considered as a combination of a relative tolerance depending on  $|u|$  and an absolute tolerance. However, for convenience, a single combined tolerance is used throughout and it is the user's responsibility to consider the proper scaling for the problem). The performance of the last iteration is then assessed as follows: if  $d \leq \frac{2}{3}e$ , then the last iteration is considered a success and the indicator  $k$  is set to 0; otherwise  $k = 1$  (the term  $\frac{2}{3}$  used in the above condition to decide a failure will be explained later on).

Test for convergence: if  $|f(u)| \leq \min(\epsilon_y, \epsilon)$  or  $d \leq 2\delta_0$  or  $l = 3$  ( $l$  is defined in 1.3), then convergence is assumed and  $\hat{\alpha} = u$ ; otherwise go to step 1.2.

1.2. Interpolation step to determine  $r$ .

The 3 points  $x_1$ ,  $x_2$  and  $x_3$  are used to fit a quadratic approximation  $g(x) = ax^2 + bx + c$  to  $f(x)$ . If  $a$  is too small, then linear approximation is used instead. It has been experienced that for a  $< 10^{-10}$  the calculation of the roots of the quadratic model is not reliable. This is machine-dependent and should be changed accordingly for other systems; its choice should also be problem-dependent, but this kind of dependence is not particularly crucial for the operability of the algorithm, although the speed of convergence might be impaired because of the premature switch from quadratic to linear approximation. The interpolation estimate  $r$  is then obtained by solving the

approximation function  $g(x) = 0$  (of the two real roots of  $g$ , the one closer to  $z$  will be chosen). If a round-off error causes  $g(x)$  to have no real roots or  $r \notin [x_1, x_2]$ , then let  $r = z$ .

1.3. This step calculates the new point  $w$  where the objective function will be evaluated.

If  $l = 2$  ( $l$  is the number of consecutive times that  $g$  can closely approximate  $f$  and will be defined more clearly in step 1.4),  $g$  is considered to be a reasonably good approximation of  $f$  and a direct interpolation is used to estimate  $\alpha$ , thus

$$w = \begin{cases} u + \delta_0 \cdot \text{sign}(z - u) & \text{if } |r - u| < \delta_0, \\ r & \text{otherwise.} \end{cases}$$

If  $l < 2$ ,  $w$  is determined by bisection or cushion interpolation; if  $|r - u| \geq d/2$  or  $k = 1$ , then use bisection, i.e. let  $w = z$ ; otherwise a cushion interpolation step is applied which is defined as

$$w = r + \left| \frac{r - s}{2} \right| \cdot \text{sign}(r - u)$$

but this step must not be too small. That is, if  $|w - u| < \delta$ , then let  $w = u + \delta \cdot \text{sign}(z - u)$  and if  $|w - u| \geq d/2$ , then  $w = z$ . This step is called cushion interpolation since the new point  $w$  is displaced from the actual interpolation estimate by the use of a cushion  $|r - s|/2 \cdot \text{sign}(r - u)$  to increase the probability of  $\alpha$  being bracketed within the small interval. It should be emphasised that a step of  $\delta$  is used for cushion interpolation, but a step of  $\delta_0$  is used for direct interpolation since there is no need to apply a very fine step like  $\delta_0$  to a crude cushion. Moreover, a larger step like  $\delta$  would give much more information than a smaller step.

1.4. Compare  $f(w)$  and  $g(w)$  to decide whether  $g$  can adequately represent  $f$  in the neighbourhood of  $\alpha$ ; i.e. let

$$l = \begin{cases} 0 & \text{if } |f(w) - g(w)| > \varepsilon_y, \\ l + 1 & \text{otherwise.} \end{cases}$$

(The use of  $|r - s|$  instead of  $|f(w) - g(w)|$  as an indicator for switching to direct interpolation can be dangerous due to rounding errors, particularly when  $|f(x_1)|$  is very large compared to  $|f(x_2)|$  or *vice versa*.)

1.5. Reduce the search interval.

If  $f(w) \cdot \text{sign}(f(x_1)) < 0$ , then let  $x_3 = x_2$ ,  $x_2 = w$  and go back to step 1.1; otherwise two choices of interval for the next iteration exist, namely  $[x_1, w, x_2]$  and  $[w, x_2, x_3]$ , since both can satisfy condition (1); the choice is made by using the smallest interval of the two. That is, if  $d \leq |x_3 - w|$ , then let  $x_3 = x_1$ ,  $x_1 = x_2$ ,  $x_2 = w$  and go to step 1.1; otherwise let  $x_1 = w$  and go to step 1.1.

It is obvious that in each iteration the argument value  $w$  as determined in step 1.3 is distinct from any existing point with the mutual distance of at least  $d/10$ . Let  $I_i$ , for iteration  $i = 1, 2, \dots, n$ , denote the closed interval whose endpoints are  $x_{1i}$  and  $x_{2i}$ . Then from the relation  $f(x_{1i}) \cdot f(x_{2i}) \leq 0$  and the operations of steps 1.3 and 1.5, it follows immediately that  $I_i$  contains a zero  $\alpha$  of  $f$  and  $I_1 \supset I_2 \supset I_3 \supset \dots \supset I_i$ . This proves that LZ1 will converge to a zero  $\alpha$  of  $f$  as  $i \rightarrow \infty$ .

Following from the definitions of the algorithm, in particular from step 1.3, a bisection is always performed whenever the previous iteration is a failure according to the concept of success and failure defined in step 1.1. The worst case would be a

step of  $\delta$  followed by a bisection which corresponds to  $I_i$  smaller than  $(\frac{9}{20})I_{i-2}$ . Hence two steps of LZ1 always reduce the length of the search interval faster than  $\ln(9/20)/(-\ln 2) \approx 1.15$  steps of bisection, or, equivalently, about 1.7 steps of LZ1 is faster than one step of bisection. Thus, the number of function evaluations needed by algorithm LZ1 is bounded above by  $1.7n_b$ . The above discussion neglects the effect of direct interpolation steps which usually occur only in the final iteration.

Now, in order for two consecutive steps of success to guarantee roughly the same rate of reduction, it is required that  $I_i = t^2 I_{i-2} \leq (9/20)I_{i-2}$  where  $t$  is the desired minimum reduction rate. This results in  $t \approx \frac{2}{3}$  which is used in the definition for failure in step 1.1.

The small bound on the number of function evaluations needed by LZ1 is made possible by the fact that only one failure is required to necessitate a bisection step due to the use of cushion interpolation. However, in LZ2 and LZ3 (described in §§ 3 and 4) since  $w$  estimates  $\alpha$  directly, at least two failures must be registered before bisection is sought to prevent bisection from being used too frequently.

The cushion interpolation used in LZ1 and LZ2 is motivated by the assumption that  $r$  is a better estimate of the root  $\alpha$  than  $s$  and that  $\alpha$  is likely to lie between  $r + (r-s)/2$  and  $r - (r-s)/2$ . Thus the use of cushion interpolation would increase the possibility of  $\alpha$  being bracketed within the smaller interval. It is seen that  $r$  cannot be expected to be reliable if  $|r-s|$  is large which then requires a large cushion or a bisection. If  $r$  approaches  $\alpha$ ,  $(r-s)$  and hence the cushion approaches 0 and thus cushion interpolation becomes direct interpolation.

The next algorithm to be described is algorithm LZ2 which has been designed to relax some degree of complexity of LZ1 but, unfortunately, at the cost of higher bound on number of function evaluations.

**3. Algorithm LZ2.** Given the input mentioned in § 2, LZ2 produces a sequence of  $w$  converging to a zero  $\alpha$  of  $f$  by using a mixture of bisection, cushion interpolation and direct interpolation. Basically, direct interpolation is used to estimate  $\alpha$  but whenever it is considered a failure, then cushion interpolation is sought instead. The algorithm will switch to bisection if a second consecutive failure occurs.

As in LZ1, the algorithm LZ2 starts a typical iteration with 3 distinct points,  $x_1$ ,  $x_2$ ,  $x_3$ , which satisfy the conditions (1) and proceed according to the following four steps.

2.1. This step is similar to step 1.1 of algorithm LZ1 except that:

- (i)  $\delta$  is defined as  $\delta = \max(d/100, \delta_0)$ ;
- (ii)  $k = 0$  if  $d \leq 0.6e$  and  $k = k + 1$  otherwise, and
- (iii) convergence is assumed if  $|f(u)| \leq \epsilon$ , or  $d \leq 2\delta_0$ .

The term  $d/100$  is used in the definition of  $\delta$  instead of  $d/10$  since, unlike in LZ1, the advantage of smaller bound on the number of function evaluations resulted from the use of  $d/10$  is nullified by the use of steps of  $\delta_0$  in direct interpolation and, furthermore, numerical study shows that  $d/100$  sometimes yields slightly better results.

2.2. This step calculates the interpolation estimate  $r$  using quadratic or linear approximation and is the same as step 1.2 of algorithm LZ1.

2.3. The new point  $w$  is chosen as follows:

If  $|r-u| \geq d/2$  or  $k = 2$ , then bisection is used:  $w = z$ .

If  $k = 0$ , direct interpolation is used and

$$w = \begin{cases} u + \delta_0 \cdot \text{sign}(z - u) & \text{if } |r - u| < \delta_0, \\ r & \text{otherwise.} \end{cases}$$

If  $k = 1$ , cushion interpolation is applied:

$$w = r + \left| \frac{r-s}{2} \right| \cdot \text{sign}(r-u) = \begin{cases} u + \delta \cdot \text{sign}(z-u) & \text{if } |w-u| < \delta, \\ z & \text{if } |w-u| \geq d/2. \end{cases}$$

2.4. This step is again exactly the same as step 1.5 of algorithm LZ1.

We see that the whole algorithm LZ2 is essentially the same as LZ1 with the major differences being in step 2.3. Following from the choice of  $w$ , as described in step 2.3, it is obvious that the three argument values  $x_1$ ,  $w$  and  $x_2$  are distinct and have a mutual distance which is bounded below by  $\delta_0$ . Thus the convergence proof for algorithm LZ2 can be constructed similar to that for LZ1 and hence will not be repeated here.

The number of function evaluations needed by algorithm LZ2 is at most  $3n_b$ . This follows immediately from the definition of the algorithm: a bisection step is always performed whenever the last two steps are considered as failures which implies the length of  $I_i$  is smaller than half the length of  $I_{i-3}$ .

The desired minimum reduction ratio  $t$  used in the definition of success (S) and failure (F) should be chosen as large as possible to increase the number of direct interpolations and to reduce the chance of performing bisection too frequently, but it must be carefully designed to still guarantee the above bound of  $3n_b$ . It can be shown that the cycle FS (one failure and one success) can be the slowest and thus the choice of  $t$  depends on the worst case of this cycle. Thus the total reduction rate of  $t^3$  of three cycles FS must be smaller than that of two cycles FFB (failure-failure-bisection), which yields  $t \leq 0.63$ . This explains the use of  $0.6e$  in the definition of failure in step 2.1.

**4. Algorithm LZ3.** The algorithm to be described in this section features the combination of bisection with linear interpolation and extrapolation instead of quadratic interpolation, and is much simpler than the previous two.

A typical iteration of algorithm LZ3 with 3 distinct points  $x_1$ ,  $x_2$ ,  $x_3$  satisfying conditions (1) consists of the following steps (Fig. 2):

3.1. This step is similar to step 2.1 of algorithm LZ2 except that both  $s$  and  $\delta$  are not defined since cushion interpolation is not used in this algorithm.

3.2. Linear interpolation and extrapolation.

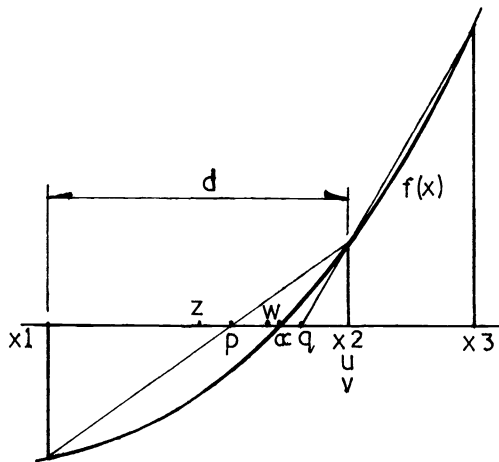


FIG. 2. Typical configuration of LZ3.

Let  $p$  be the linear interpolation point using  $x_1$  and  $x_2$

$$p = x_2 - f(x_2) \cdot (x_1 - x_2) / (f(x_1) - f(x_2))$$

and  $q$  be the linear extrapolation point using  $x_2$  and  $x_3$ , which is defined as:

$$q = \begin{cases} x_2 - f(x_2) \cdot (x_3 - x_2) / (f(x_3) - f(x_2)) & \text{if } |f(x_3)/(x_3 - x_1)| > |f(x_2)/d|, \\ x_1 & \text{otherwise.} \end{cases}$$

3.3. The new point  $w$  is determined as follows:

If  $k=2$  or  $j=1$  ( $j$  is defined in step 3.4), then bisection is used, that is  $w = z$ ; otherwise let

$$w = p + (q - p) / 2 = \begin{cases} z & \text{if } w \notin [x_1, x_2], \\ v + \delta_0 \cdot \text{sign}(z - v) & \text{if } |w - v| < \delta_0, \end{cases}$$

where  $v$  is defined as either the point  $x_1$  or  $x_2$  whichever is closer to  $w$ .

3.4. This step reduces the search interval and checks if there is any inflection point in the vicinity of the root.

Let the inflection indicator  $j = 0$ .

If  $f(w) \cdot \text{sign}(f(x_1)) < 0$ , then let  $x_3 = x_2$ ,  $x_2 = w$  and go back to step 3.1; otherwise let  $j = 1$  if  $(h - p) \cdot \text{sign}(h - q) > 0$  (i.e., there exists an inflection point), where  $h = x_2 - f(x_2) \cdot (w - x_2) / (f(w) - f(x_2))$ . Let  $x_3 = x_1$ ,  $x_1 = x_2$ ,  $x_2 = w$  and go back to step 3.1.

The convergence proof and bound on the number of function evaluations needed for algorithm LZ3 is exactly the same as for LZ2.

The motivation behind this algorithm is that for a convex (or concave) curve, the root  $\alpha$  of  $f$  must lie between  $p$  and  $q$  and thus bisection could be used for this reduced interval  $[p, q]$  instead of  $[x_1, x_2]$ .

Despite the fact that LZ3 uses only one function evaluation per iteration and utilises only linear approximation, its order of convergence, as proved below, can be comparable to that of Newton-Raphson's method and may be slightly better than quadratic or 3-point rational interpolation.

In the following proof, let us assume for all iterations  $n$  greater than certain  $n_0$  that  $|x_1 - x_3|$  is sufficiently small for certain conditions to hold and that  $\delta_0 < |w - v|$  (see step 3.3). The latter condition ensures that, for  $n \geq n_0$ , the tolerance function  $\delta_0$  does not influence the  $n$ th iteration step. Another important assumption necessary for the validity of the following order of convergence of the basic interpolation process is that the bisection step cannot occur asymptotically.

Let the function  $f$  have three continuous derivatives and the root  $\alpha$  be a unique simple root. Because  $\alpha$  is simple, then  $f'(\alpha) \neq 0$  and thus  $f'(x) \neq 0$  for all  $x$  in a certain neighbourhood of the root  $\alpha$ .

Let  $x_n$  be the most recent estimate of  $\alpha$  and  $\varepsilon_n$  be the error in the estimate  $x_n$ , i.e.  $\varepsilon_n = x_n - \alpha$ . Suppose that the method converges, i.e.  $\lim_{n \rightarrow \infty} x_n = \alpha$ .

Let  $a_{n+1}$  and  $b_{n+1}$  be defined as

$$(2) \quad a_{n+1} = x_n - \frac{f(x_n)}{f[x_{n-1}, x_n]}$$

$$(3) \quad b_{n+1} = x_n - \frac{f(x_n)}{f[x_{n-2}, x_n]}$$

and

$$(4) \quad x_{n+1} = \frac{a_{n+1} + b_{n+1}}{2},$$

where  $f[x_{n-1}, x_n]$  is the first divided difference of  $f$  at  $x_{n-1}$  and  $x_n$ . That is,

$$f[x_{n-1}, x_n] = \frac{f(x_n) - f(x_{n-1})}{x_n - x_{n-1}}.$$

It is seen that  $a_{n+1}$  and  $b_{n+1}$  and hence  $x_{n-1}$  and  $x_{n-2}$  are interchangeable in (4), thus  $x_{n-1}$  or  $x_{n-2}$  can correspond to either notation  $x_1$  or  $x_3$  in the above description of the algorithm.

Substituting (2) and (3) into (4),

$$x_{n+1} = x_n - \frac{f(x_n)}{2f[x_{n-1}, x_n]} - \frac{f(x_n)}{2f[x_{n-2}, x_n]}$$

or

$$(5) \quad f(x_n) + 2(x_{n+1} - x_n)f[x_{n-1}, x_n] + f(x_n) \frac{f[x_{n-1}, x_n]}{f[x_{n-2}, x_n]} = 0.$$

Now, using Newton's interpolation formula with error term for  $x = \alpha$

$$(6) \quad \begin{aligned} f(\alpha) = 0 = & f(x_n) + (\alpha - x_n)f[x_{n-1}, x_n] + (\alpha - x_n)(\alpha - x_{n-1}) \frac{(f[x_{n-2}, x_{n-1}] - f[x_{n-1}, x_n])}{x_{n-2} - x_n} \\ & + (\alpha - x_n)(\alpha - x_{n-1})(\alpha - x_{n-2}) \frac{f^{(3)}(\xi_1)}{6} \end{aligned}$$

where  $\xi_1 \in \text{int}[\alpha, x_{n-2}, x_{n-1}, x_n]$ .

Subtract (6) from (5) and use the mean-value theorem

$$\begin{aligned} f[x_{n-1}, x_n] &= f'(\xi_2), \quad \xi_2 \in \text{int}[x_{n-1}, x_n], \\ f[x_{n-2}, x_n] &= f'(\xi_3), \quad \xi_3 \in \text{int}[x_{n-2}, x_n], \\ f[x_{n-2}, x_{n-1}] &= f'(\xi_4), \quad \xi_4 \in \text{int}[x_{n-2}, x_{n-1}]; \end{aligned}$$

thus

$$(7) \quad \begin{aligned} \varepsilon_{n+1}f'(\xi_2) + (x_{n+1} - x_n)f'(\xi_2) + f(x_n) \frac{f'(\xi_2)}{f'(\xi_3)} \\ - \varepsilon_n \varepsilon_{n-1} \frac{f'(\xi_4) - f'(\xi_2)}{x_{n-2} - x_n} + \varepsilon_n \varepsilon_{n-1} \varepsilon_{n-2} \frac{f^{(3)}(\xi_1)}{6} = 0. \end{aligned}$$

Since  $x_{n+1} - x_n = x_{n+1} - x_n + \alpha - \alpha = \varepsilon_{n+1} - \varepsilon_n$  and similarly  $x_{n-2} - x_n = \varepsilon_{n-2} - \varepsilon_n$ , (7) becomes

$$\begin{aligned} 2\varepsilon_{n+1}f'(\xi_2) - \varepsilon_n f'(\xi_2) + f(x_n) \frac{f'(\xi_2)}{f'(\xi_3)} - \varepsilon_n \varepsilon_{n-1} \frac{(f'(\xi_4) - f'(\xi_2))}{x_{n-2} - x_n} \\ + \varepsilon_n \varepsilon_{n-1} \varepsilon_{n-2} \frac{f^{(3)}(\xi_1)}{6} = 0 \end{aligned}$$

or

$$(8) \quad \varepsilon_{n+1} = \frac{\varepsilon_n}{2} \frac{f(x_n)}{2f'(\xi_3)} + \frac{\varepsilon_n \varepsilon_{n-1}}{\varepsilon_{n-2} - \varepsilon_n} \left( \frac{f'(\xi_4)}{2f'(\xi_2)} - \frac{1}{2} \right) - \varepsilon_n \varepsilon_{n-1} \varepsilon_{n-2} \frac{f^{(3)}(\xi_1)}{12f'(\xi_2)}.$$



Expanding  $f$  in a Taylor series about  $x_n$  gives

$$f(\alpha) = 0 = f(x_n) + (\alpha - x_n)f'(x_n) + \frac{1}{2}(\alpha - x_n)^2 f''(\xi_5), \quad \xi_5 \in \text{int}[x_n, \alpha];$$

thus

$$(9) \quad f(x_n) = \varepsilon_n f'(x_n) - \frac{1}{2} \varepsilon_n^2 f''(\xi_5).$$

Substituting (9) into (8) and as  $x_n$  approaches  $\alpha$

$$\varepsilon_{n+1} \approx \frac{\varepsilon_n}{2} - \frac{\varepsilon_n f'(\alpha) - (1/2) \varepsilon_n^2 f''(\alpha)}{2f'(\alpha)} - \varepsilon_n \varepsilon_{n-1} \varepsilon_{n-2} \frac{f^{(3)}(\alpha)}{12f'(\alpha)}$$

or

$$(10) \quad \varepsilon_{n+1} \approx \frac{f''(\alpha)}{4f'(\alpha)} \varepsilon_n^2 - \frac{f^{(3)}(\alpha)}{12f'(\alpha)} \varepsilon_n \varepsilon_{n-1} \varepsilon_{n-2}.$$

One special case is when  $f^{(3)}(\alpha) = 0$ , then

$$\frac{|\varepsilon_{n+1}|}{\varepsilon_n^2} \approx \left| \frac{f''(\alpha)}{4f'(\alpha)} \right|$$

which means quadratic convergence with asymptotic error constant better than that of  $|f''(\alpha)/2f'(\alpha)|$  of the Newton–Raphson method. Let  $A = (f''(\alpha)/4f'(\alpha))\varepsilon_n^2$  and  $B = (f^{(3)}(\alpha)/12f'(\alpha))\varepsilon_n \varepsilon_{n-1} \varepsilon_{n-2}$ ; if  $A \cdot B > 0$  and  $|A| > |B|$ , then

$$|\varepsilon_{n+1}| \approx \left| \frac{f''(\alpha)}{4f'(\alpha)} \right| \varepsilon_n^2 - \left| \frac{f^{(3)}(\alpha)}{12f'(\alpha)} \right| \cdot |\varepsilon_n| \cdot |\varepsilon_{n-1}| \cdot |\varepsilon_{n-2}|$$

or  $|\varepsilon_{n+1}| < |f''(\alpha)/4f'(\alpha)|\varepsilon_n^2$ , which is again faster than the Newton–Raphson method. This is also a special case and can only occur occasionally if  $|f''(\alpha)|$  is very large. In general, however, the second term of (10) dominates asymptotically, thus

$$(11) \quad |\varepsilon_{n+1}| \approx \left| \frac{f^{(3)}(\alpha)}{12f'(\alpha)} \right| \cdot |\varepsilon_n| \cdot |\varepsilon_{n-1}| \cdot |\varepsilon_{n-2}|.$$

By writing  $|\varepsilon_{n+1}| \approx K|\varepsilon_n|^p$ ,  $|\varepsilon_n| \approx K|\varepsilon_{n-1}|^p$  and  $|\varepsilon_{n-1}| \approx K|\varepsilon_{n-2}|^p$ , then (11) becomes

$$K|\varepsilon_n|^p \approx \left| \frac{f^{(3)}(\alpha)}{12f'(\alpha)} \right| \cdot |\varepsilon_n|^{(p^2+p+1)/p^2} K^{(-2p-1)/p^2}.$$

It follows that the effective asymptotic order of convergence is 1.839 which is the largest root of the equation  $p^3 - p^2 - p - 1 = 0$ . This is similar to the order of convergence of quadratic interpolation.

**5. Numerical studies.** The performance of LZ1, LZ2 and LZ3 has been obtained for the following test functions, which were supplied with an interval containing the zero.

*Function 1.* This function is described in Brent (1973):

$$f(x) = -2 \sum_{i=1}^{20} \frac{(2i-5)^2}{(x-i^2)^3}$$

in the interval  $[n^2 + 10^{-9}, (n+1)^2 - 10^{-9}]$  for  $n = 1(1)19$ .

*Function 2.*  $f(x) = ax e^{bx}$  in the interval  $[-9, 31]$ , where

1.  $a = -40$  and  $b = -1$ ,
2.  $a = -100$  and  $b = -2$ ,
3.  $a = -200$  and  $b = -3$ .

This function has a stationary point near its simple zero and takes the  $x$ -axis as its asymptote for large  $x$ .

*Function 3.*  $f(x) = x^n - a$ , where

1.  $a = 0.2$  and  $n = 4, 6, 8, 10, 12$  in the interval  $[0, 5]$ ,

2.  $a = 1$  and  $n = 4, 6, 8, 10, 12$  in the interval  $[0, 5]$ ,

3.  $a = 1$  and  $n = 8, 10, 12, 14$  in the interval  $[-0.95, 4.05]$ .

Functions 4 to 10 were taken from Dowell and Jarratt (1971), each with a simple zero in the interval considered.

*Function 4.*  $f(x) = \sin(x) - 0.5$  in the interval  $[0, 1.5]$ .

*Function 5.*  $f(x) = 2x e^{-n} - 2 e^{-nx} + 1$  in the interval  $[0, 1]$  and  $n = 1, 2, 3, 4, 5, 15, 20$ .

*Function 6.*  $f(x) = (1 + (1 - n)^2)x - (1 - nx)^2$  in the interval  $[0, 1]$  and  $n = 1, 2, 5, 10, 15, 20$ .

*Function 7.*  $f(x) = x^2 - (1 - x)^n$  in the interval  $[0, 1]$  and  $n = 1, 2, 5, 10, 15, 20$ .

*Function 8.*  $f(x) = (1 + (1 - n)^4)x - (1 - nx)^4$  in the interval  $[0, 1]$  and  $n = 1, 2, 4, 5, 8, 15, 20$ .

*Function 9.*  $f(x) = (x - 1) e^{-nx} + x^n$  in the interval  $[0, 1]$  and  $n = 1, 5, 10, 15, 20$ .

*Function 10.*  $f(x) = (nx - 1)/((n - 1)x)$  in the interval  $[0.01, 1]$  and  $n = 2, 5, 15, 20$ .

*Function 11.* This function is a simple polynomial having a zero of multiplicity  $n$  and was tested by Bus and Dekker (1975):

$$f(x) = x^n \text{ in the interval } [-1, 10] \text{ and } n = 3, 5, 7, 9, 19, 25.$$

*Function 12.* This is a function given by Brent (1973) with a zero of multiplicity  $\infty$  and defined by:

$$f(x) = \begin{cases} 0 & \text{if } x = 0, \\ x \cdot \exp(-x^{-2}) & \text{otherwise.} \end{cases}$$

The interval  $[-1, 4]$  was considered for this problem.

The test results of LZ1, LZ2 and LZ3 under various stopping criteria are tabulated in Tables 1 to 6 compared against the performance of the following algorithms: algorithms *A*, *M* and *R* described in Bus and Dekker (1975), algorithm *B* published by Brent (1971) and algorithm *C* by Anderson and Bjorck (1973). Except for functions 2 and 3 where the results of algorithm *B* have actually been obtained by the author, all performance data of those competing algorithms have been quoted from other authors' works. Brent's algorithm has been used for comparison with the three new algorithms in solving functions 2 and 3 due to the availability of its FORTRAN version (see Brent (1973)) and since tests by other authors (e.g. Bus and Dekker (1975) and Swift and Lindfield (1978)) indicate that Brent's algorithm is one of the most efficient over a wide range of problems. Since LZ1, LZ2 and LZ3 are designed to work on the objective function value rather than on the argument interval (see the discussion in § 2), their results as presented in the tables were chosen such that the following condition is satisfied:  $|\hat{\alpha} - \alpha| \leq \varepsilon_1 |\alpha| + \varepsilon_2$ , where  $\hat{\alpha}$  is the final estimate of the zero  $\alpha$  and  $\varepsilon_1, \varepsilon_2$  are assigned appropriate values so the results can be comparable to those of competing methods. Tables 1 to 6 show the number of function evaluations needed by the various algorithms to locate a root of the given function to within the required precision as dictated in the following list of stopping criteria:

- (a) Final argument interval is  $2\delta$  for algorithm *B*, where  $\delta = 2 \times 16^{-7} |x| + 10^{-10}$ ; see Brent (1973). The corresponding criterion for LZ1, LZ2, LZ3 is:  $|\hat{\alpha} - \alpha| \leq 10^{-8} |\alpha| + 10^{-10}$ .

TABLE 1  
Number of function evaluations for function 1.

n	B	LZ1	LZ2	LZ3
	Stopping criterion (a)			
1	14	11	11	14
2	8	9	10	12
3	14	9	10	11
4	12	8	8	10
5	12	8	8	11
6	11	10	9	12
7	11	10	9	11
8	11	9	9	9
9	10	8	9	8
10	10	9	9	8
11	10	8	9	8
12	10	10	9	8
13	10	10	9	8
14	10	9	9	8
15	10	7	7	8
16	10	9	9	8
17	10	9	9	8
18	9	8	7	8
19	9	9	9	8
Total	201	170	169	178

TABLE 2  
Number of function evaluations for function 2.

Interval	a	b	B	LZ1	LZ2	LZ3
			Stopping criterion (b)			
[-9, 31]	-40	-1	16	12	16	12
	-100	-2	18	13	19	13
	-200	-3	19	13	20	13
Total			53	38	55	38

- (b) Final interval is  $2\delta$  for algorithm  $B$ , where  $\delta = 10^{-8}(2|x| + 1)$ . For LZ1, LZ2, LZ3:  $|\hat{\alpha} - \alpha| \leq 10^{-8}(2|\alpha| + 1)$ .
- (c) Final interval is  $2\delta$  for algorithm  $B$ , where  $\delta = 10^{-7}(2|x| + 1)$ . For LZ1, LZ2, LZ3:  $|\hat{\alpha} - \alpha| \leq 10^{-7}(2|\alpha| + 1)$ .
- (d) For algorithms  $A, M, R, B, C$ : final argument interval is  $2\delta$ ,  $\delta = 10^{-14}(|x| + 1)$ ; see Bus and Dekker (1975). For LZ1, LZ2, LZ3:  $|\hat{\alpha} - \alpha| \leq 10^{-14}(|\alpha| + 1)$ .
- (e) For algorithms  $B$  and  $C$ : Final interval is  $\delta$ ,  $\delta = 6 \times 10^{-8}(|x| + 1)$ ; see Anderson and Bjorck (1973). For LZ1, LZ2, LZ3:  $|\hat{\alpha} - \alpha| \leq 3 \times 10^{-8}(|\alpha| + 1)$ .
- (f) Final interval is  $2\delta$ ,  $\delta = 0.5 \times 10^{-8}(|x| + 1)$ , for the modified Davidenko-Broyden method and the combined search-Brent method; see Swift and Lindfield (1978). For LZ1, LZ2, LZ3:  $|\hat{\alpha} - \alpha| < 0.5 \times 10^{-8}(|\alpha| + 1)$ .

TABLE 3  
Number of function evaluations for function 3.

Interval	$a$	$n$	$B$	LZ1	LZ2	LZ3
			Stopping criterion (c)			
[0, 5]	0.2	4	13	9	8	11
		6	15	10	8	10
		8	16	11	11	10
		10	16	11	13	11
		12	16	11	13	11
[0, 5]	1.0	4	14	9	9	11
		6	14	10	10	10
		8	13	10	10	10
		10	15	10	11	10
		12	16	11	12	12
[-0.95, 4.05]	1.0	8	14	10	11	10
		10	14	10	11	11
		12	15	10	10	11
		14	16	11	13	11
Total			207	143	150	149

The above list of stopping criteria also indicates the references from which the performance data of other algorithms originated.

The three new algorithms have also been used to solve test problems supplied with only one starting point. These sample problems were taken from Swift and Lindfield (1978) and are listed below:

*Function 13.*  $f(x) = 2x e^{-n} - 2 e^{-nx} + 1$  with starting value 0. and  $n = 5$ .

*Function 14.*  $f(x) = (1 + (1 - n)^4)x - (1 - nx)^4$  with starting value 0. and  $n = 5$ .

*Function 15.*  $f(x) = x^n$  with starting value  $-1$ . and  $n = 3, 5, 7$ .

*Function 16.*  $f(x) = (e^{-nx} - x - 0.5)/x^n$  with starting value 0.1185 and  $n = 5$ .

*Function 17.*  $f(x) = x^{-1/2} - 2 \cdot \log_{10}(nx^{1/2}) + 0.8$  with starting value 0.001 and  $n = 5 \times 10^3, 5 \times 10^7$ .

In order to supply the initial bracketing interval required by LZ1, LZ2 and LZ3, the interval locating procedure given by Swift and Lindfield referred to in their paper as *findb* has been used. It should be noted that this procedure cannot guarantee convergence as it can lead to a zero, a stationary point or an asymptotic constant and it may fail to locate an interval with end points having function values of opposite sign. This will be the case when the function is asymptotic to a constant in the direction of search and thus  $x$  will become unbounded, or when the stepsize becomes large enough to bracket two or an even number of roots within it (if the function has more than one root) and the procedure will bounce back and forth with unboundedly increasing stepsize. Although several simple courses can be taken in such a situation, for example restart with a new starting point or smaller stepsize, no attempts have been made to modify the procedure *findb*.

Table 7 shows the performances of the combined search procedure *findb* with LZ1, LZ2 and LZ3 with initial stepsize of 0.001 against those of the modified Davidenko-Broyden continuation method and of the combined search-Brent method as reported by Swift and Lindfield.

TABLE 4  
Number of function evaluations for functions 4-10.

Function	Interval	<i>n</i>	A	M	R	B	C	LZ1	LZ2	LZ3	B	C	LZ1	LZ2	LZ3
			Stopping criterion (d)								Stopping criterion (e)				
4	[0, 1.5]	—	10	10	9	8	9	9	7	8	8	7	7	6	7
5	[0, 1]	1	9	9	7	8	7	9	10	9	7	6	7	7	7
		2	10	10	8	9	8	10	9	9					
		3	11	11	9	10	9	10	9	9					
		4	12	12	10	10	10	10	9	10					
		5										10	10	8	7
		15									11	12	9	9	9
		20									11	12	10	9	11
6	[0, 1]	1	10	9	8	8	9	6	4	8					
		2									9	8	6	4	8
		5	10	10	9	9	8	6	4	10	8	8	6	4	8
		10	9	9	9	9	8	6	4	9					
		15									7	7	6	4	7
		20								7	6	6	4	7	
7	[0, 1]	1	9	10	8	9	9	5	4	9					
		2									3	3	3	3	3
		5	10	10	9	9	10	10	9	9	8	8	8	7	8
		10	11	11	11	10	11	10	9	13					
		15									10	11	9	7	12
		20								12	11	9	10	12	
8	[0, 1]	1	10	10	8	9	9	10	9	8					
		2									10	9	8	9	8
		4	9	9	9	8	8	9	9	9					
		5									7	8	6	7	7
		8	7	7	8	7	8	10	9	8					
		15								6	7	6	6	6	
		20								6	6	6	6	5	
9	[0, 1]	1	9	9	8	9	9	8	9	9	8	7	7	7	8
		5	9	9	9	9	9	10	9	13	8	8	8	7	11
		10	10	10	10	9	10	10	12	14	8	9	9	10	13
		15									11	9	8	13	13
		20									12	10	10	13	15
10	[0.01, 1]	2									5	6	8	8	9
		5									12	7	9	10	9
		15									11	6	11	10	9
		20									13	6	11	12	10
Total			165	165	149	150	151	148	135	164	228	207	201	199	232

All computations have been carried out on the CDC Cyber 72 with 48 bits accuracy in floating point at the University of New South Wales. It should be noted that some exponent underflows have occurred during evaluations of function 11 for  $n = 25$  and function 12. We would like to point out that comparisons based on data compiled from different works may not be totally satisfactory because of the differences in computer and compiler systems, precisions used, programmer's coding ability, etc.

Furthermore, although the results obtained by different algorithms are of similar accuracy, the number of function evaluations listed for LZ1, LZ2 and LZ3 are in fact not directly comparable to those of other algorithms. This is because the three new algorithms stop as soon as one approximation  $\hat{\alpha}$  to the zero  $\alpha$  satisfying the stopping

TABLE 5  
Number of function evaluations for function 11.

Interval	$n$	A	M	R	B	C	LZ1	LZ2	LZ3
		Stopping criterion (d)							
	3	117	151	91	147	118	40	92	64
	5	206	149	163	122	207	42	100	64
	7	293	161	206	138	294	53	122	59
	9	380	160	196	137	381	49	96	74
	19	802	179	206	141	759	56	127	72
	25	1320	159	174	123	961	63	115	71
Total		3118	959	1036	808	2720	303	652	404

TABLE 6  
Number of function evaluations for function 12.

Interval	A	M	R	B	C	LZ1	LZ2	LZ3	
		Stopping criterion (d)							
[-1, 4]	>5000	27	23	18	969	4	11	11	

TABLE 7  
Number of function evaluations for problems with 1 starting point.

Function	Starting value	$n$	Modified				
			Davidenko-Broyden	B	LZ1	LZ2	LZ3
			Stopping criterion (f)				
13	0.	5	12	14	14	14	13
14	0.	5	8	7	7	7	6
15	-1.	3	130	126	34	47	40
	-1.	5	141	86	33	63	44
	-1.	7	146	67	35	73	39
16	0.1185	5	20	13	11	10	11
17	0.001	$5 \times 10^3$	18	13	12	11	11
	0.001	$5 \times 10^7$	13	10	9	8	9
Total			488	336	155	233	173

criterion has been found, while other algorithms have to find two. It could be expected that on average one or two extra function evaluations would be required to locate the second approximation that brackets the zero. These extra function evaluations are certainly not necessary if the first approximation produces a real zero as was the case with algorithm LZ3 for 7 out of 17 functions in Table 4. Because of this difference, performance measures of other algorithms cannot be compared directly with those of LZ1, LZ2 and LZ3, although they do provide a strong basis for judgement.

Although limited resources coupled with the difficulty discussed in § 2 have prevented more complete comparisons, the large set of test problems combined with various stopping criteria certainly demonstrated the merits of the algorithms.

Although Tables 1 and 4 do not indicate conclusively about the relative performances of the new algorithms, bearing in mind the effects of different stopping criteria, the remaining tables clearly point to the superiority of the new algorithms compared to others. The differences in the number of function evaluations range from 4 or 5 for simple zero functions (Tables 2 and 3) up to more than one hundred for multiple zero functions (Tables 5, 6, 7). It should be mentioned that as Table 4 contains quadratic test problems (function 6 and function 7 with  $n = 1$ ), it slightly favours LZ1 and LZ2 which use quadratic interpolation as their basic process, just as it also favours algorithm *C* with function 10 because of its relation to rational approximation.

**6. Conclusion.** In this paper we have presented two new approaches for approximating a zero of an arbitrary function. The first involves the concept of cushion interpolation which forms the basic process in the algorithm LZ1. This algorithm has a very small bound on the number of function evaluations required, namely  $1.7n_b$ , while its performance is still very competitive to other existing algorithms. Our second algorithm LZ2 is a variant of LZ1. While it has a slightly simpler operation than LZ1 and still performs comparably with LZ1, an inevitable drawback with LZ2 is a higher bound on the number of function evaluations needed, namely  $3n_b$ . However, this bound is still lower than those of existing algorithms. The third algorithm LZ3, with a completely different approach, shows extremely good performance and, on many occasions, it can even outperform LZ1 and LZ2, despite the fact that it is simpler and uses only linear interpolation. Its maximum number of function evaluations is also bounded by  $3n_b$ . Although the number of function evaluations required by the three new algorithms can be as high as  $3n_b$ , in practice they only slightly exceed that needed by the bisection method and the most outstanding performance in this respect belongs to LZ1. On these above limited tests, our new algorithms have demonstrated their robustnesses and efficient performances on both well- and ill-behaved functions.

**7. Acknowledgments.** The author wishes to thank Dr. G. Smith for his constructive suggestions and his careful reading of the manuscript. Valuable comments by the referees are also greatly appreciated.

#### REFERENCES

- [1] N. ANDERSON AND A. BJORCK, *A new high order method of regula falsi type for computing a root of an equation*, BIT, 13 (1973), pp. 253–264.
- [2] R. P. BRENT, *An algorithm with guaranteed convergence for finding a zero of a function*, Computer J., 14 (1971), pp. 422–425.
- [3] ———, *Algorithms for Minimisation Without Derivatives*, Prentice Hall, Englewood Cliffs, NJ, 1973.
- [4] J. C. P. BUS AND T. J. DEKKER, *Two efficient algorithms with guaranteed convergence for finding a zero of a function*, ACM Trans. Math. Software, 1 (1975), pp. 330–345.
- [5] G. DAHLQUIST AND A. BJORCK, *Numerical Methods*, Prentice-Hall, Englewood Cliffs, NJ, 1974.
- [6] T. J. DEKKER, *Finding a zero by means of successive linear interpolation*, in Constructive Aspects of the Fundamental Theorem of Algebra, B. Dejon and P. Henrici, eds., Wiley Interscience, New York, 1969, pp. 37–48.
- [7] M. DOWELL AND P. JARRATT, *A modified regula falsi method for computing the root of an equation*, BIT, 11 (1971), pp. 168–174.
- [8] ———, *The “Pegasus” method for computing the root of an equation*, BIT, 12 (1972), pp. 503–508.
- [9] G. H. GONNET, *On the structure of zero finders*, BIT, 17 (1977), pp. 170–183.

- [10] A. RALSTON AND P. RABINOWITZ, *First Course in Numerical Analysis*, McGraw-Hill, New York, 1978.
- [11] A. SWIFT AND G. R. LINDFIELD, *Comparison of a continuation method with Brent's method for the numerical solution of a single nonlinear equation*, *Computer J.*, 21 (1978), pp. 359–362.
- [12] J. F. TRAUB, *Iterative Methods for the Solution of Equations*, Prentice-Hall, Englewood Cliffs, NJ, 1964.



## THE MATHEMATICAL STRUCTURE OF HUBER'S $M$ -ESTIMATOR\*

D. I. CLARK†

**Abstract.** The structure of the defining function of Huber's  $M$ -estimator is examined. The behaviour of the function is explored, together with the question of uniqueness and the connection with the  $L_1$  estimator. This analysis is necessary for the construction of a finite algorithm for the  $M$ -estimator, which will be published subsequently.

**Key words.** Huber's  $M$ -estimator, robustness,  $L_1$  estimator, uniqueness

**1. Introduction.** Classically, given a linear model

$$(1) \quad \mathbf{b} = \mathbf{A}\mathbf{x} + \mathbf{r},$$

where  $\mathbf{b}$  is an  $n$ -vector of dependent observations,  $\mathbf{A}$  an  $n \times m$  array ( $n > m$ ) of independent observations,  $\mathbf{x}$  an  $m$ -vector to be estimated and  $\mathbf{r}$  an  $n$ -vector of errors or residuals,  $\mathbf{x}^* = \mathbf{z}$  has been chosen so that

$$(2) \quad \sum_{i=1}^n r_i(\mathbf{z})^2 = \min.$$

The estimator  $\mathbf{z}$  is called the least squares or  $L_2$  estimator and was shown by Gauss in 1821 [4] to be the "most probable value" under the assumptions that the errors have independent identical normal distributions. However, as was illustrated by Tukey in 1960 [12], the  $L_2$  estimator is very sensitive to quite small deviations from these assumptions, and, in particular, a few gross errors can have a marked effect.

In an effort to find a more robust estimator, Huber [7] suggested replacing the squared term in (2) with the less rapidly increasing function

$$(3) \quad \rho(r_i) = \begin{cases} \frac{1}{2}r_i^2 & \text{for } |r_i| \leq \gamma, \\ \gamma|r_i| - \frac{1}{2}\gamma^2 & \text{for } |r_i| > \gamma, \end{cases}$$

where  $\gamma$  is a parameter to be estimated from the data. The resulting estimator was shown by Huber [10] to be a maximum likelihood estimator for a perturbed normal distribution and has become known as Huber's  $M$ -estimator.

The most popular approaches to calculating the  $M$ -estimator have been iterative schemes based on letting  $\psi = \rho'$  and solving the equivalent system

$$\sum_i \psi(r_i) = 0.$$

These are Huber's method [8]

$$\mathbf{x}^{j+1} = \mathbf{x}^j + [\mathbf{A}^T \mathbf{A}]^{-1} \mathbf{A}^T \psi(\mathbf{r}^j),$$

Huber and Dutter's application of Newton's method [11],

$$\mathbf{x}^{j+1} = \mathbf{x}^j + [\mathbf{A}^T \langle \psi'(\mathbf{r}^j) \rangle \mathbf{A}]^{-1} \mathbf{A}^T \psi(\mathbf{r}^j),$$

and the iteratively reweighted least squares (IRLS) method attributed to Beaton and Tukey [1]

$$\mathbf{x}^{j+1} = \mathbf{x}^j + [\mathbf{A}^T \langle \mathbf{w}(\mathbf{r}^j) \rangle \mathbf{A}]^{-1} \mathbf{A}^T \langle \mathbf{w}(\mathbf{r}^j) \rangle \mathbf{r}^j,$$

\* Received by the editors July 21, 1981, and in final revised form October 18, 1983.

† Department of Statistics, Institute of Advanced Studies, Australian National University, Canberra, ACT, 2600, Australia.

where  $\langle \mathbf{a} \rangle$  denotes the diagonal matrix with  $\langle \rangle_{ii} = a_i$ , and  $w$  is a weight function such as  $w(t) = \psi(t)/t$ . Holland and Welsch [6] give a detailed account of the IRLS method with some comments on the relative rates of convergence of the three methods.

A different approach is based on Huber's [8] observation that if the correct partitioning of the minimizing residuals  $r_i$  into  $\sigma = \{i \mid |r_i| \leq \gamma\}$ ,  $\sigma_+ = \{i \mid r_i > \gamma\}$ ,  $\sigma_- = \{i \mid r_i < -\gamma\}$  were known, the  $M$ -estimator could be calculated in a single step as a minimizer of

$$\frac{1}{2} \sum_{\sigma} r_i^2 + \sum_{\sigma_+} (\gamma r_i - \frac{1}{2} \gamma^2) + \sum_{\sigma_-} (-\gamma r_i - \frac{1}{2} \gamma^2).$$

The difficulty, of course, is that the minimizing partition is not known a priori. Indeed, as Huber has noted, "The search for the  $M$ -estimator is the search for the correct partition."

Our approach [3], to be published subsequently, finds the correct partition by regarding  $\mathbf{z}$  as a function of  $\gamma$ , starting with the  $L_2$  estimator,  $\mathbf{z}(\infty)$ . We use the results of this paper to prove that  $\mathbf{z}$  is a continuous piecewise linear function of  $\gamma$ , and to show how to find a new feasible partition at the end of a range of  $\gamma$ , when one or more residuals change status.

This paper is concerned with the function  $\rho$  defined in (3) (as applied to residuals  $r_i$  defined in (1)) in an effort to understand its underlying structure, as opposed to justifying its statistical virtues. We find it convenient to use a partitioning  $\sigma, \bar{\sigma} = \sigma_+ \cup \sigma_-$  and to use the signs of  $r_i, i \in \bar{\sigma}$ . The  $M$ -estimator then becomes the minimizer  $\mathbf{z}$  of

$$(4) \quad \frac{1}{2} \sum_{\sigma} r_i^2 + \sum_{\bar{\sigma}} (\gamma |r_i| - \frac{1}{2} \gamma^2),$$

which, as  $|r_i| > 0, i \in \bar{\sigma}$ , is differentiable, and so  $\mathbf{z}$  satisfies

$$(5) \quad \sum_{\sigma} (\mathbf{a}_i \mathbf{a}_i^T \mathbf{z} - b_i \mathbf{a}_i) + \sum_{\bar{\sigma}} \gamma \theta_i \mathbf{a}_i = \mathbf{0},$$

where  $\mathbf{a}_i$  is the  $i$ th column of  $A^T$  and  $\theta_i = \text{sgn}(r_i)$ . The approach taken will be to study the function (4), assuming a partition  $\sigma, \bar{\sigma}$  of the residuals and the signs of  $r_i, i \in \bar{\sigma}$ , and to examine the requirements of such a partition being feasible, i.e.  $i \in \sigma \Leftrightarrow |r_i| \leq \gamma$  and the residuals of  $\bar{\sigma}$  are of the assumed sign. This gives us a necessary background for our algorithm mentioned above. It also seems a natural one, focusing on one of the aims of finding the  $M$ -estimator, that of identifying a set of outlying observations for closer scrutiny.

Another estimator which has received a great deal of attention in the quest for robustness is the  $L_1$  estimator, the solution of

$$\sum |r_i| = \min,$$

described variously as the most robust estimator in some sense [5], [10], and as the only technically robust  $L_p$  estimator of location [9]. As the  $L_1$  estimator is characterized [13, p. 118] by a partitioning of  $N$  into  $\sigma = \{i \mid r_i = 0\}$ ,  $\bar{\sigma} = \{i \mid r_i \neq 0\}$ , so that

$$\sum_{\sigma} \lambda_i \mathbf{a}_i + \sum_{\bar{\sigma}} \theta_i \mathbf{a}_i = \mathbf{0}, \quad |\lambda_i| \leq 1,$$

it would seem that the partitioning approach should throw light onto the relationship between the  $L_1$  and  $M$ -estimators.

We shall therefore use this approach to investigate the uniqueness of the  $M$ -estimator, the behaviour of the function and its function values, and the connection with the  $M$ -estimator.

**2. Definitions and conventions.** A partition  $P$  is a splitting of the set  $N = \{1, \dots, n\}$  into disjoint complementary subsets  $\sigma$  and  $\bar{\sigma}$ . The function associated with  $P$  is

$$F(\mathbf{x}) = \frac{1}{2} \sum_{\sigma} r_i(\mathbf{x})^2 + \sum_{\bar{\sigma}} \{\gamma|r_i(\mathbf{x})| - \frac{1}{2}\gamma^2\}.$$

$\mathbf{z}$  will denote the minimizer of  $F(\mathbf{x})$ , and  $F(\mathbf{z})$  will be called the value of the partition. Residuals of a partition will be measured at  $\mathbf{z}$ .

A residual is tight if its absolute value equals  $\gamma$ .

A partition is tight if at least one of its residuals is tight. "Tightness" will, unless otherwise stated, refer to partitions.

A partition is  $\sigma$ -feasible if  $|r_i(\mathbf{z})| \leq \gamma, i \in \sigma$ , and  $\bar{\sigma}$ -feasible if for  $i \in \bar{\sigma}, |r_i(\mathbf{z})| > \gamma$ , and the residuals are of the assumed sign.

A partition is feasible if it is  $\sigma$ -feasible and  $\bar{\sigma}$ -feasible.

Uniqueness will, unless otherwise stated, refer to the number of feasible partitions, rather than to whether a particular partition has a unique minimum.

$\mathbf{a}_i$  will denote the  $i$ th column of  $A^T$ .

$\theta_i$  will denote the sign of the  $i$ th residual,  $\theta_i = \text{sgn}(r_i)$ .

Adjacent partitions  $P_a$  and  $P_b$  satisfy  $\sigma_a = \sigma_b \cup \{k\}$  or  $\sigma_a = \sigma_b / \{k\}$ .

The data for the examples will be displayed as a tableau having the form

$$\begin{matrix} A^T \\ \mathbf{b}^T \end{matrix}, \quad \text{or} \quad \begin{matrix} A^T \\ \mathbf{b}^T \end{matrix} \quad \gamma = .$$

**3. Theorems and examples.** We first prove several lemmata, building up to Lemma 7, the key result. As the proofs of this lemma and its corollary are substantially longer and more complex than the remainder of the paper, they appear, in the interest of the readability of the paper as a whole, in the Appendix.

Several examples are given. These were constructed using the optimality criteria (5). As they are in general designed to demonstrate unusual behaviour, they tend to be sensitive in that a slight perturbation causes them to no longer illustrate the point being made. Consequently they were often difficult to construct and may not be easy to check. Verification, however, may be made easier by using the results of this section. This is done for the first three examples.

LEMMA 1. *The minimum of*

$$F = \frac{1}{2} \sum_{\sigma} r_i^2 + \sum_{\bar{\sigma}} (\gamma|r_i| - \frac{1}{2}\gamma^2)$$

*is characterized by*

$$\sum_{\sigma} r_i \mathbf{a}_i + \sum_{\bar{\sigma}} \gamma \phi_i \mathbf{a}_i = \mathbf{0},$$

where

$$\phi_i = \theta_i, \quad r_i \neq 0, \quad \phi_i \in [-1, 1], \quad r_i = 0.$$

LEMMA 2. *If  $P_a$  and  $P_b$  are adjacent partitions with  $\sigma_a = \sigma_b \cup \{k\}$ , then for any  $\mathbf{x}$  satisfying  $|r_k(\mathbf{x})| = \gamma, F_a(\mathbf{x}) = F_b(\mathbf{x})$ .*

*Proof.* The proof follows directly from

$$\frac{1}{2} r_k(\mathbf{x})^2 = \gamma|r_k(\mathbf{x})| - \frac{1}{2}\gamma^2. \quad \square$$

LEMMA 3. If  $P_a$  and  $P_b$  are adjacent partitions with  $\sigma_a = \sigma_b \cup \{k\}$  for which a minimizer  $\mathbf{z}$  of either partition has  $|r_k(\mathbf{z})| = \gamma$ , then  $\mathbf{z}$  also minimizes the other partition.

*Proof.* The proof follows directly from Lemma 1.  $\square$

LEMMA 4. If a partition has nonunique minimizers  $\mathbf{z}_1$  and  $\mathbf{z}_2$ , then

- (i) for  $i \in \sigma$ ,  $r_i(\mathbf{z}_1) = r_i(\mathbf{z}_2)$ ;
- (ii) for  $i \in \bar{\sigma}$ ,

$$\text{either } \text{sgn } r_i(\mathbf{z}_1) = \text{sgn } r_i(\mathbf{z}_2),$$

or at least one of  $r_i(\mathbf{z}_1) = 0$  and  $r_i(\mathbf{z}_2) = 0$  holds.

*Proof.* Let  $\mathbf{z}_3 = \frac{1}{2}\mathbf{z}_1 + \frac{1}{2}\mathbf{z}_2$ . Then, from the convexity of  $F$ ,  $\mathbf{z}_3$  also minimizes  $F$ , so that

$$(6) \quad F(\mathbf{z}_3) = \frac{1}{2}F(\mathbf{z}_1) + \frac{1}{2}F(\mathbf{z}_2).$$

Now  $r_i(\mathbf{z}_3) = \frac{1}{2}r_i(\mathbf{z}_1) + \frac{1}{2}r_i(\mathbf{z}_2)$ , so (6) becomes

$$(7) \quad \begin{aligned} & \frac{1}{8} \sum_{\sigma} \{r_i(\mathbf{z}_1) + r_i(\mathbf{z}_2)\}^2 + \frac{1}{2} \gamma \sum_{\bar{\sigma}} |r_i(\mathbf{z}_1) + r_i(\mathbf{z}_2)| \\ &= \frac{1}{4} \sum_{\sigma} \{r_i(\mathbf{z}_1)^2 + r_i(\mathbf{z}_2)^2\} + \frac{1}{2} \gamma \sum_{\bar{\sigma}} \{|r_i(\mathbf{z}_1)| + |r_i(\mathbf{z}_2)|\}. \end{aligned}$$

But  $(p+q)^2 \leq 2(p^2+q^2)$  and  $|p+q| \leq |p|+|q|$ , so that equality in (7) can occur only when all the corresponding elements are equal.  $\square$

LEMMA 5. If  $\sigma_a = \sigma_b \cup S$ , then  $F_a(\mathbf{x}) \geq F_b(\mathbf{x})$ , with equality holding only if  $|r_i(\mathbf{x})| = \gamma$ ,  $i \in S$ .

*Proof.* The proof follows directly from  $\frac{1}{2}r_i^2 \geq \gamma|r_i| - \frac{1}{2}\gamma^2$ , the inequality being strict unless  $|r_i| = \gamma$ .  $\square$

LEMMA 6. A feasible partition has a unique minimum iff the vectors  $\mathbf{a}_i$ ,  $i \in \sigma$ , span  $\mathbb{R}^m$ .

*Proof.* (i) If  $\mathbf{z}_1$  and  $\mathbf{z}_2$  both minimize  $F$ , then from Lemma 4,  $r_i(\mathbf{z}_1) = r_i(\mathbf{z}_2)$ ,  $i \in \sigma$ , and so the vectors  $\mathbf{a}_i$ ,  $i \in \sigma$ , do not span  $\mathbb{R}^m$ .

(ii) If the vectors  $\mathbf{a}_i$ ,  $i \in \sigma$ , do not span  $\mathbb{R}^m$ , there exists a vector  $\mathbf{d}$  orthogonal to the  $\mathbf{a}_i$ ,  $i \in \sigma$ . Then for  $i \in \sigma$ ,  $r_i(\mathbf{z} + \alpha\mathbf{d}) = r_i(\mathbf{z})$ , and for  $i \in \bar{\sigma}$  and small enough  $\alpha$ ,  $\text{sgn } r_i(\mathbf{z} + \alpha\mathbf{d}) = \text{sgn } r_i(\mathbf{z})$ , so that  $\mathbf{z} + \alpha\mathbf{d}$  also satisfies the optimality criteria of Lemma 1.  $\square$

LEMMA 7. If  $S$  is any proper subset of  $\sigma_a$  such that  $|r_i(\mathbf{z}_a)| \leq \gamma$ ,  $i \in S$ , and  $\mathbf{x}'$  is a vector satisfying  $|r_i(\mathbf{x}')| > \gamma$ ,  $i \in S$ , then  $F_b(\mathbf{z}_a) \leq F_b(\mathbf{x}')$ , where  $\sigma_b = \sigma_a \cap \bar{S}$ .

Further, if  $|r_i(\mathbf{z}_a)| < \gamma$  for any  $i \in S$ ,  $F_b(\mathbf{z}_a) < F_b(\mathbf{x}')$ .

*Proof.* See Appendix.

COROLLARY.  $F_a(\mathbf{z}_a) \leq F_b(\mathbf{x}')$ .

*Proof.* See Appendix.

Our first three theorems are concerned with the uniqueness or otherwise of a feasible partition. As one of the aims of finding the  $M$ -estimator is to identify sets of observations as potential outliers, the existence of alternate sets under the same model is of interest. We are therefore concerned with recognizing when alternate feasible partitions may occur, and with their characteristics when they do.

THEOREM 1(a). Let  $P_a$  and  $P_b$  be feasible partitions with  $\sigma_a = \sigma_b \cup S$ . Then

$$|r_i(\mathbf{z}_a)| = \gamma, \quad \forall i \in S.$$

*Proof.* From the feasibility of  $\mathbf{z}_a$  and  $\mathbf{z}_b$ , if  $|r_i(\mathbf{z}_a)| < \gamma$  for any  $i \in S$ , then by Lemma 7,  $F_b(\mathbf{z}_a) < F_b(\mathbf{z}_b)$ , which contradicts the optimality of  $\mathbf{z}_b$  for  $F_b$ .  $\square$

Remark 1. Note that if  $P_a$  and  $P_b$  are feasible partitions with  $\sigma_a = \sigma_b \cup S$ , it is not necessary for any partition of the form  $\sigma_b \cup S'$ ,  $S' \subset S$ , to be feasible.

*Example 1.*

$$\begin{array}{cccc} 1 & 2 & 2 & 0 \\ 1 & 3 & 0 & 3 \\ 2 & 4 & 3 & 5 \end{array} \quad \gamma = 1.$$

The feasible partitions are  $\sigma_a = \{1, 2, 3\}$ ,  $\mathbf{z}_a^T = (1, 1)$ , and  $\sigma_b = \{1\}$ ,  $\mathbf{z}_b^T \in ((1, 1), (2/3, 4/3)]$ . But  $\mathbf{r}(\mathbf{z}_a)^T = (0, 1, -1, -2)$ , so from Lemmata 3 and 6(i) (which does not need the feasibility of the partition),  $\mathbf{z}_a$  uniquely minimizes  $\sigma_c = \{1, 2\}$  and  $\sigma_d = \{1, 3\}$ . Thus neither  $\sigma_c$  nor  $\sigma_d$  is feasible.

THEOREM 1(b). *Let  $P_a$  and  $P_b$  be feasible partitions, and let  $\sigma_c = \sigma_a \cap \sigma_b$ . Then*

$$\begin{aligned} |r_i(\mathbf{z}_a)| &= \gamma, & i \in \sigma_a \cap \bar{\sigma}_c & (= \sigma_a \cap \bar{\sigma}_b), \\ |r_i(\mathbf{z}_b)| &= \gamma, & i \in \sigma_b \cap \bar{\sigma}_c & (= \bar{\sigma}_a \cap \sigma_b). \end{aligned}$$

*Proof.*  $\mathbf{z}_a$  and  $\mathbf{z}_b$  are feasible for  $P_a$  and  $P_b$ , respectively, and as  $\sigma_a = \sigma_c \cup (\sigma_a \cap \bar{\sigma}_c)$ , from Lemma 7,

$$(8) \quad F_c(\mathbf{z}_a) \leq F_c(\mathbf{z}_b).$$

Again, as  $\sigma_b = \sigma_c \cup (\sigma_b \cap \bar{\sigma}_c)$ , from Lemma 7,

$$(9) \quad F_c(\mathbf{z}_b) \leq F_c(\mathbf{z}_a).$$

Now if, for some  $i \in \sigma_a \cap \bar{\sigma}_c$ ,  $|r_i(\mathbf{z}_a)| < \gamma$ , (8) becomes  $F_c(\mathbf{z}_a) < F_c(\mathbf{z}_b)$ ; and if, for some  $i \in \sigma_b \cap \bar{\sigma}_c$ ,  $|r_i(\mathbf{z}_b)| < \gamma$ , (9) becomes  $F_c(\mathbf{z}_b) < F_c(\mathbf{z}_a)$ . Either condition causes (8) and (9) to be inconsistent.  $\square$

*Remark 2.* Note that a partition may be tight, even for a range of  $\gamma$ , and still be the only feasible partition.

*Example 2.*

$$\begin{array}{cccccc} 1 & 0 & 1 & 2 & 3 & 0 \\ 0 & 1 & 2 & 2 & \frac{1}{6} & 4 \\ 1 & 1 & 1 & 2 & 8 & 12 \end{array}$$

For  $3 \leq \gamma < 4.17$ ,  $\sigma = \{1, 2, 3, 4\}$  with  $\mathbf{z} = \frac{2}{3} + \gamma/9, \frac{1}{3} + 7\gamma/18$ , so that  $r_4 = \gamma$ . But for the whole of this range of  $\gamma$ ,  $|r_1| < \gamma$  and  $|r_2| < \gamma$ , so that by the corollary below there are no other feasible partitions.

COROLLARY. *A necessary condition for a feasible partition not to be unique is that the nontight vectors of  $\sigma$  do not span  $\mathbb{R}^m$ .*

*Proof.* Let  $P_a$  and  $P_b$  be two feasible partitions, and let  $\sigma_c = \sigma_a \cap \sigma_b$ . Then from Theorem 1(b) and Lemma 3,  $\mathbf{z}_a$  and  $\mathbf{z}_b$  both minimize  $F_c(\mathbf{x})$ , and so from Lemma 6 the vectors  $\mathbf{a}_i, i \in \sigma_c$ , do not span  $\mathbb{R}^m$ . But from Theorem 1(b) the nontight vectors of  $\sigma_a$  are included in  $\sigma_c$ .

*Remark 3.* We now have two necessary conditions for nonuniqueness, namely tightness and the nontight vectors of  $\sigma$  not spanning  $\mathbb{R}^m$ . These two conditions, however, are not sufficient to ensure nonuniqueness.

*Example 3.*

$$\begin{array}{ccccc} 1 & 9.5 & 7 & 1 & 3 \\ 1 & 7 & 9 & 3 & 0.5 \\ 1 & -110 & 90 & 85.6 & 58 \end{array} \quad \gamma = 1.6.$$

For  $\sigma = \{1, 2, 4\}$ ,  $\mathbf{z} = (-43.3, 43.5)^T$  and  $\mathbf{r} = (-.8, 3.15, -1.6, 1.6, -166.15)^T$ . Now if there were another feasible partition  $P_a$ , from Theorem 1(b), then  $1 \in \sigma_a$  and  $|r_i(\mathbf{z}_a)| = \gamma$ ,  $i \in \sigma_a \setminus \{1\}$ . But from Theorem 3, below,  $r_1(\mathbf{z}_a) = r_1(\mathbf{z})$ , so that  $\mathbf{z}_a = \mathbf{z} + \alpha(-1, 1)^T$ . But as  $\alpha$  increases,  $r_2$  and  $r_5$  decrease while  $r_3$  and  $r_4$  increase. Now for  $0 < \alpha < 1.6$ ,  $|r_3| < \gamma$ , and for  $0.62 < \alpha < 1.9$ ,  $|r_2| < \gamma$ . So the only possibilities for a feasible partition with  $\alpha > 0$  are

$$\alpha = 1.9, \quad \sigma = \{1, 2\}, \quad r_2 = -\gamma, \quad r_3 > \gamma, \quad r_4 > \gamma, \quad r_5 < -\gamma,$$

and

$$\alpha > 1.9, \quad \sigma = \{1\}, \quad r_2 < -\gamma, \quad r_3 > \gamma, \quad r_4 > \gamma, \quad r_5 < -\gamma,$$

and it is easily verified that in neither case are the optimality criteria of Lemma 1 satisfied. Similar reasoning shows that no alternate feasible solutions are possible with  $\alpha < 0$ . (This behaviour is also exhibited at  $\gamma = 1.803$ ,  $\sigma = \{1, 2, 4\}$ .)

**THEOREM 2.** *If  $P_a$  is a feasible partition,  $P_b$  is  $\sigma$ -feasible and  $P_c$  is  $\bar{\sigma}$ -feasible, then:*

- (i)  $F_a(\mathbf{z}_a) \leq F_b(\mathbf{z}_b)$ ;
- (ii)  $F_a(\mathbf{z}_a) \geq F_c(\mathbf{z}_c)$ .

*Proof.* (i) Let  $\sigma_d = \sigma_a \cap \sigma_b$ . then from the corollary of Lemma 7 and Lemma 5,

$$F_a(\mathbf{z}_a) \leq F_d(\mathbf{z}_b) \leq F_b(\mathbf{z}_b).$$

(ii) Let  $\sigma_d = \sigma_a \cap \sigma_c$ . Then from the corollary of Lemma 7 and Lemma 5,

$$F_c(\mathbf{z}_c) \leq F_d(\mathbf{z}_a) \leq F_a(\mathbf{z}_a). \quad \square$$

**COROLLARY.** *The function values of all feasible partitions are equal.*

**THEOREM 3.** *Let  $P_a$  and  $P_b$  be distinct feasible partitions, and let  $\sigma_c = \sigma_a \cap \sigma_b$ . Then:*

- (i)  $r_i(\mathbf{z}_a) = r_i(\mathbf{z}_b)$ ,  $i \in \sigma_c$ ;
- (ii)  $\text{sgn } r_i(\mathbf{z}_a) = \text{sgn } r_i(\mathbf{z}_b)$ ,  $i \in \bar{\sigma}_c$ .

*Proof.* From Theorem 1(b) and Lemma 3,  $\mathbf{z}_a$  and  $\mathbf{z}_b$  both minimize  $F_c(\mathbf{x})$ . The result now follows from Lemma 4.  $\square$

**COROLLARY.** *If  $P_a$  and  $P_b$  are distinct feasible partitions, then*

$$\sum |r_i(\mathbf{z}_a)| = \sum |r_i(\mathbf{z}_b)|.$$

*Proof.* From the corollary of Theorem 2 and noting that  $\sigma_a = \sigma_a \cap \sigma_c + \sigma_a \cap \bar{\sigma}_c = \sigma_a \cap \sigma_b + \sigma_a \cap \bar{\sigma}_c$ ,

$$\begin{aligned} \frac{1}{2} \sum_{\sigma_a \cap \sigma_b} r_i(\mathbf{z}_a)^2 + \frac{1}{2} \sum_{\sigma_a \cap \bar{\sigma}_c} r_i(\mathbf{z}_a)^2 + \sum_{\bar{\sigma}_a} \{ \gamma |r_i(\mathbf{z}_a)| - \frac{1}{2} \gamma^2 \} \\ = \frac{1}{2} \sum_{\sigma_a \cap \sigma_b} r_i(\mathbf{z}_b)^2 + \frac{1}{2} \sum_{\sigma_b \cap \bar{\sigma}_c} r_i(\mathbf{z}_b)^2 + \sum_{\bar{\sigma}_b} \{ \gamma |r_i(\mathbf{z}_b)| - \frac{1}{2} \gamma^2 \}. \end{aligned}$$

From Theorem 1(b), and using  $\frac{1}{2} r_i^2 = \gamma |r_i| - \frac{1}{2} \gamma^2$  when  $|r_i| = \gamma$ , and observing that  $\sigma_a \cap \bar{\sigma}_c + \bar{\sigma}_a = \bar{\sigma}_c$ ,

$$\frac{1}{2} \sum_{\sigma_c} r_i(\mathbf{z}_a)^2 + \sum_{\bar{\sigma}_c} \gamma |r_i(\mathbf{z}_a)| = \frac{1}{2} \sum_{\sigma_c} r_i(\mathbf{z}_b)^2 + \sum_{\bar{\sigma}_c} \gamma |r_i(\mathbf{z}_b)|.$$

The result now follows from Theorem 3.  $\square$

Our next two theorems are concerned mainly with algorithm building. In particular, for our finite algorithm Theorem 5 is needed to show that  $\mathbf{z}(\gamma)$  is piecewise linear, and Theorem 4 is required to demonstrate the feasibility of the new partition in a new range of  $\gamma$ .

**THEOREM 4.** *Let  $P_a$  and  $P_b$  be adjacent partitions with unique minima  $\mathbf{z}_a$  and  $\mathbf{z}_b$ , respectively, and let  $\sigma_a = \sigma_b \cup \{k\}$ . Then:*

(i)  $|r_k(\mathbf{z}_a)| > \gamma \Leftrightarrow |r_k(\mathbf{z}_b)| > \gamma$ ;

(ii)  $|r_k(\mathbf{z}_a)| \leq \gamma \Leftrightarrow |r_k(\mathbf{z}_b)| \leq \gamma$ .

*Proof.* Assume  $|r_k(\mathbf{z}_a)| > \gamma$  and  $|r_k(\mathbf{z}_b)| \leq \gamma$ , or  $|r_k(\mathbf{z}_a)| \leq \gamma$  and  $|r_k(\mathbf{z}_b)| > \gamma$ . Define  $\mathbf{x} = \alpha \mathbf{z}_a + (1 - \alpha) \mathbf{z}_b$ ,  $0 \leq \alpha \leq 1$ , such that  $|r_k(\mathbf{x})| = \gamma$ . Then from, in turn, Lemma 5,  $\mathbf{z}_a$  minimizing  $F_a$ , Lemma 2 and convexity,

$$(10) \quad F_b(\mathbf{z}_a) \leq F_a(\mathbf{z}_a) \leq F_a(\mathbf{x}) = F_b(\mathbf{x}) \leq \alpha F_b(\mathbf{z}_a) + (1 - \alpha) F_b(\mathbf{z}_b).$$

If  $\alpha = 1$ , from Lemma 3,  $\mathbf{z}_a$  minimizes  $F_b$ , and if  $\alpha < 1$ , (10) implies  $F_b(\mathbf{z}_a) \leq F_b(\mathbf{z}_b)$ , and again  $\mathbf{z}_a$  minimizes  $F_b$ . This contradicts the assumption of the uniqueness of the minimizer of  $F_b$ .  $\square$

*Remark 4.* The above theorem limits the behaviour only of  $r_k$  at the change of partition. Strange things can happen to other residuals, even when one of the partitions is feasible.

*Example 4.*

$$\begin{array}{cccccc} 1 & 10 & 1 & 0.9 & 10 & \\ 2.5 & 10 & 3.9 & 5 & 40 & \gamma = 1. \end{array}$$

For  $\sigma = \{3\}$ ,  $z = 3.8$  and  $\mathbf{r} = (1.3, 28, -0.1, -1.58, -2)^T$ , and for  $\sigma = \{2, 3\}$ ,  $z = 1.147$  and  $\mathbf{r} = (-1.35, 1.47, -2.75, -4, -28.53)^T$ . Note the behaviour of  $r_1$  and  $r_3$  at the change of partition and that  $\sigma = \{3\}$  is the only feasible partition.

**THEOREM 5.** *Provided rank  $A = m$ , there exists a feasible partition for which there are  $m$  linearly independent (LI) vectors in  $\sigma$ .*

*Proof.* Assume that a feasible partition  $P_1$  has fewer than  $m$  LI vectors in  $\sigma_1$ . Then there will be a vector  $\mathbf{d}$  orthogonal to  $\mathbf{a}_i$ ,  $i \in \sigma_1$ , and as rank  $A = m$ ,  $\exists k \in \bar{\sigma}_1$  for which  $\mathbf{a}_k^T \mathbf{d} > 0$  (possibly after replacing  $\mathbf{d}$  with  $-\mathbf{d}$ ). As in Lemma 6,  $\mathbf{z} + \alpha \mathbf{d}$  also minimizes  $F_1(\mathbf{x})$ , small  $\alpha$ . Let  $\alpha_1 = \min \{\alpha \mid |r_k(\mathbf{z} + \alpha \mathbf{d})| = \gamma, k \in \bar{\sigma}_1\}$ , achieved when  $k = k_1$ . Then from Lemma 3,  $\mathbf{z} + \alpha_1 \mathbf{d}$  also minimizes  $\sigma_2 = \sigma_1 \cup \{k_1\}$ , and  $\sigma_2$  has one more LI vector than  $\sigma_1$ . This process can be repeated until there are  $m$  LI vectors in  $\sigma_l$ .  $\square$

Finally, we further explore the relationship with the  $L_1$  estimator. Clearly, for  $\gamma = 0$ ,  $\mathbf{z}(\gamma)$  is the  $L_1$  estimator. Here we show that for a range of  $\gamma$  the feasible partition is a subset of  $\sigma_0$ .

**THEOREM 6.** *Let  $\mathbf{z}_0$  be the  $L_1$  estimator and  $\sigma_0 = \{i \mid r_i(\mathbf{z}_0) = 0\}$ . Then for small enough but finite  $\gamma$ ,  $\sigma_\gamma \subseteq \sigma_0$ .*

*Proof.* From the characterization of the  $L_1$  optimum (see, e.g., Watson [13]), at  $\mathbf{z}_0 \exists \lambda, |\lambda_i| \leq 1$ , such that

$$(11) \quad \sum_{\sigma_0} \lambda_i \mathbf{a}_i + \sum_{\bar{\sigma}_0} \theta_i \mathbf{a}_i = \mathbf{0}.$$

Now let  $\gamma$  be small but positive, and consider the partition  $\sigma_\gamma = \sigma_0$ . Provided  $r_i(\mathbf{z}_\gamma) \neq 0$ ,  $i \in \bar{\sigma}_\gamma$ , the characterization of the optimum (Lemma 1) gives

$$(12) \quad \sum_{\sigma_0} \frac{r_i}{\gamma} \mathbf{a}_i + \sum_{\bar{\sigma}_0} \theta_i \mathbf{a}_i = \mathbf{0}.$$

Now as  $\mathbf{z}(\gamma) \rightarrow \mathbf{z}(0)$  as  $\gamma \rightarrow 0$  (the partition being fixed), and  $r_i(\mathbf{z}_0) \neq 0$ ,  $i \in \bar{\sigma}_0$ , then for small enough  $\gamma$ ,  $r_i(\mathbf{z}_\gamma) \neq 0$  and, further,  $\text{sgn } r_i(\mathbf{z}_\gamma) = \text{sgn } r_i(\mathbf{z}_0)$ ,  $i \in \bar{\sigma}_0$ .

If there is no degeneracy in  $\sigma_0$ , i.e. if rank  $[\mathbf{a}_i]$ ,  $i \in \sigma_0 = |\sigma_0|$ , then (11) and (12) will have unique solutions for  $\lambda_i$  and  $r_i/\gamma$  so that  $\lambda_i = r_i/\gamma$ , and hence  $|r_i| \leq \gamma$ ,  $i \in \sigma_0$ . And as  $|r_i(\mathbf{z}_\gamma)| > \gamma$ ,  $i \in \bar{\sigma}_0$  and small enough  $\gamma$ ,  $\sigma_0$  is a feasible partition and  $\sigma_\gamma = \sigma_0$ .

If, however, there is degeneracy in  $\sigma_0$ , there is no longer a unique solution to (11) and we can no longer guarantee  $|r_i| \leq \gamma, i \in \sigma_0$ . Indeed, as is shown in Example 5, below,  $\sigma_0$  may not be a feasible partition for any  $\gamma > 0$ . In this case, however, there will be a subset of  $\sigma_0$  which is a feasible partition. The detailed proof for this is rather lengthy, and is only sketched below.

The proof depends on an algorithm ([2, Algorithm 5.2, p. 83]) which will be published in our subsequent paper. This algorithm, in a finite number of steps, moves from a  $\bar{\sigma}$ -feasible partition to a feasible partition by interchanges to adjacent partitions, i.e.  $\sigma \rightarrow \sigma \pm \{k\}$ . Finiteness of this algorithm is proved using Lemmata 3 and 5.

By choosing  $\gamma$  small enough,  $\bar{\sigma}$ -feasibility is ensured for  $\sigma_0$ . Again by choosing  $\gamma$  small enough, it can be guaranteed that only elements  $k \in \sigma_0$  are involved in changes to partitions in the algorithm above, i.e. at each stage of the algorithm  $\sigma_i \subseteq \sigma_0$ , and as the algorithm terminates finitely, there must be a subset of  $\sigma_0$  which is a feasible partition for small enough  $\gamma$ .  $\square$

*Example 5.*

3	4	0	2	7.5	
2	0	3	3	7	.
0	4	3	5	20	

The  $L_1$  solution is  $\mathbf{x} = (1, 1)^T$  with  $\sigma_0 = \{2, 3, 4\}$ , and for  $0 \leq \gamma < 1.34, \sigma_\gamma = \sigma_0$  is feasible. This is the normal situation where  $\sigma_\gamma = \sigma_0$ . However, if the "7.5" is changed to "8", then for  $0 < \gamma < 1.23, \mathbf{z}_\gamma = (1 + 3\gamma/16, 1 + 2\gamma/9)^T$ , with  $\sigma_\gamma = \{2, 3\}$  being the only feasible partition.

**COROLLARY.** *If the  $L_1$  estimator is not unique, there is a range of  $\gamma$  for which the  $M$ -estimator is not unique.*

*Remark 5.* The converse of the above corollary is not true, although it generally holds. In Example 3 above, the feasible partitions for each range of  $\gamma$  are as follows:

- For  $\gamma \geq 115,$   $\sigma = \{1, 2, 3, 4, 5\},$
- For  $115 > \gamma \geq 1.869,$   $\sigma = \{1, 2, 3, 4\},$
- For  $1.869 > \gamma \geq 1.803,$   $\sigma = \{1, 2, 4\},$
- For  $1.803 > \gamma > 1.775,$   $\sigma = \{1, 2\}$  or  $\{1, 4\}$  or  $\{1\},$
- For  $\gamma = 1.775,$   $\sigma = \{1, 2, 3\}$  or  $\{1, 4\}$  or  $\{1\},$
- For  $1.775 > \gamma > 1.6,$   $\sigma = \{1, 3\}$  or  $\{1, 4\}$  or  $\{1\},$
- For  $1.6 \geq \gamma \geq .384,$   $\sigma = \{1, 3, 4\},$
- For  $.384 > \gamma,$   $\sigma = \sigma_0 = \{3, 4\},$

so that a unique  $L_1$  solution is consistent with a nonunique  $M$ -estimator.

*Remark 6.* A final example is illustrative of the lack of predictability of the  $M$ -estimator. It is clear that for large enough  $\gamma$ , the  $M$ -estimator is simply the  $L_2$  estimator and the feasible partition is  $\sigma = N, \bar{\sigma} = \emptyset$ . As we have seen, for small enough  $\gamma, \sigma = \sigma_0$ , and it is not difficult to show that  $\mathbf{z}(\gamma)$  is a continuous piecewise linear function of  $\gamma$ , moving from the  $L_2$  to the  $L_1$  estimator as  $\gamma$  is reduced. In general, for  $0 < \gamma_2 < \gamma_1, \sigma_0 \subseteq \sigma_{\gamma_2} \subseteq \sigma_{\gamma_1}$ . However, this need not be the case, and in the following example, for a range of  $\gamma, \sigma_\gamma \cap \sigma_0 = \emptyset$ .



*Example 6.*

1	9.5	7	1	-5	2.466
1	7	9	3	1	1.475.
1	-110	90	85.6	-59	-1

For  $1.804 > \gamma > 1.776$ ,  $\sigma = \{1, 2\}$ , but for  $\gamma < 0.185$ ,  $\sigma = \sigma_0 = \{3, 4\}$ , and in both ranges there is only the one feasible partition.

**4. Summary.** We have seen that the defining function of the  $M$ -estimator is not a simple one. Odd things can happen (e.g. Example 4) which make the finding of a feasible partition difficult. We do know, however, of some limits on the behaviour at the change of partition, and how function values and the composition of  $\sigma$  change between feasible and infeasible partitions.

We have been able to establish necessary conditions for recognizing nonuniqueness, and although they are not quite sufficient, we can recognize those times that they are, and we know certain properties of nonunique solutions.

Finally, we have explored the relationship with the  $L_1$  estimator, its extent and its limits.

**Appendix. Proof of Lemma 7.**

LEMMA 7. *If  $S$  is any proper subset of  $\sigma_a$  such that  $|r_i(\mathbf{z}_a)| \leq \gamma$ ,  $i \in S$ , and  $\mathbf{x}'$  is a vector satisfying  $|r_i(\mathbf{x}')| > \gamma$ ,  $i \in S$ , then  $F_b(\mathbf{z}_a) \leq F_b(\mathbf{x}')$  where  $\sigma_b = \sigma_a \cap \bar{S}$ .*

*Further, if  $|r_i(\mathbf{z}_a)| < \gamma$  for any  $i \in S$ ,  $F_b(\mathbf{z}_a) < F_b(\mathbf{x}')$ .*

*Proof.* Let  $S = \{s_1, \dots, s_r\}$ . For convenience, let  $\mathbf{y}_0 \equiv \mathbf{z}_a$ ,  $\mathbf{y}_{r+1} \equiv \mathbf{x}'$ ,  $F_0(\mathbf{x}) \equiv F_a(\mathbf{x})$ ,  $F_r(\mathbf{x}) \equiv F_b(\mathbf{x})$ .

Now as  $|r_i(\mathbf{y}_0)| \leq \gamma$  and  $|r_i(\mathbf{y}_{r+1})| > \gamma$  for  $i \in S$ , we can, after reordering  $S$ , define points  $\mathbf{y}_1, \dots, \mathbf{y}_r$  to satisfy

- (i)  $\mathbf{y}_i = \phi_i \mathbf{y}_{r+1} + (1 - \phi_i) \mathbf{y}_0$ ;
- (13) (ii)  $|r_{s_i}(\mathbf{y}_i)| = \gamma$ ;
- (iii)  $0 \leq \phi_i \leq \phi_{i+1} \leq 1$ .

From 13(i) and 13(iii), we have

(14)  $\mathbf{y}_i = \alpha_i \mathbf{y}_0 + (1 - \alpha_i) \mathbf{y}_{i+1}, \quad 0 \leq \alpha_i \leq 1,$

where  $\alpha_i = 1 - \theta_i / \theta_{i+1}$ .

If  $\alpha_i = 1$ ,  $\theta_i = 0$ ,  $\mathbf{y}_i = \mathbf{y}_0$ , and from Lemma 2 and 13(ii),  $F_0(\mathbf{y}_0) = F_i(\mathbf{y}_0)$ , and from Lemma 3,  $\mathbf{y}_i$  minimizes  $F_i$ , thus

(15)  $F_0(\mathbf{y}_0) = F_i(\mathbf{y}_0) = F_i(\mathbf{y}_i) \leq F_i(\mathbf{y}_{i+1}).$

Let  $\sigma_i = \sigma_a \cup \{s_{i+1}, \dots, s_r\}$  define  $F_i(\mathbf{x})$ ;  $i = 1, \dots, r-1$ . Then from Lemma 6

(16)  $F_0(\mathbf{x}) \geq F_1(\mathbf{x}) \geq \dots \geq F_r(\mathbf{x}).$

From 13(ii) and Lemma 2,

(17)  $F_{i-1}(\mathbf{y}_i) = F_i(\mathbf{y}_i), \quad i = 1, \dots, r.$

Now from, in turn, (16),  $\mathbf{y}_0$  minimizing  $F_0$ , (17), convexity plus (14),

(18)  $F_1(\mathbf{y}_0) \leq F_0(\mathbf{y}_0) \leq F_0(\mathbf{y}_1) = F_1(\mathbf{y}_1) \leq \alpha_1 F_1(\mathbf{y}_0) + (1 - \alpha_1) F_1(\mathbf{y}_2),$

so that

$$(19) \quad F_1(\mathbf{y}_0) \leq \alpha_1 F_1(\mathbf{y}_0) + (1 - \alpha_1) F_1(\mathbf{y}_2).$$

Thus from either (15) ( $\alpha_1 = 1$ ), or (16) and (19) ( $\alpha_1 < 1$ ),

$$(20) \quad F_1(\mathbf{y}_0) \leq F_1(\mathbf{y}_2).$$

Again, from, in turn (16), (20), (17), convexity plus (14),

$$(21) \quad F_2(\mathbf{y}_0) \leq F_1(\mathbf{y}_0) \leq F_1(\mathbf{y}_2) = F_2(\mathbf{y}_2) \leq \alpha_2 F_2(\mathbf{y}_0) + (1 - \alpha_2) F_2(\mathbf{y}_3),$$

so that

$$(22) \quad F_2(\mathbf{y}_0) \leq \alpha_2 F_2(\mathbf{y}_0) + (1 - \alpha_2) F_2(\mathbf{y}_3).$$

Thus from either (15) ( $\alpha_2 = 1$ ), or (16) and (22) ( $\alpha_2 > 1$ ),

$$(23) \quad F_2(\mathbf{y}_0) \leq F_2(\mathbf{y}_3).$$

This process is repeated until we reach  $F_r(\mathbf{y}_0) \leq F_r(\mathbf{y}_{r+1})$ . Moreover, if, at any stage  $|r_i(\mathbf{y}_0)| < \gamma$ , i.e.  $\alpha_i \neq 1$ , then from Lemma 6,  $F_{i+1}(\mathbf{y}_0) < F_i(\mathbf{y}_0)$ , and so

$$F_r(\mathbf{y}_0) < F_r(\mathbf{y}_{r+1}). \quad \square$$

COROLLARY.  $F_a(\mathbf{z}_a) \leq F_b(\mathbf{x}')$ .

*Proof.* In the above proof, from (18),

$$(24) \quad F_0(\mathbf{y}_0) \leq \alpha_1 F_1(\mathbf{y}_0) + (1 - \alpha_1) F_1(\mathbf{y}_2),$$

and so, from (15), or (16) and (24),

$$(25) \quad F_0(\mathbf{y}_0) \leq F_1(\mathbf{y}_2),$$

and the process is repeated as before.  $\square$

**Acknowledgments.** This paper is based on part of the author's doctoral thesis, supported by an Australian National University scholarship. The author is indebted to Dr. M. R. Osborne for many helpful discussions and for advice on the presentation of this paper. The author also wishes to thank the referees for their careful reading and comments.

#### REFERENCES

- [1] 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.
- [2] D. I. CLARK (1981), *Finite algorithms for linear optimization problems*, doctoral thesis, Australian National University, Canberra.
- [3] D. I. CLARK AND M. R. OSBORNE, *Finite algorithms for Huber's M-estimator*, to appear.
- [4] C. F. GAUSS (1821), *Göttingische Gelhrte Anzeigen*, pp. 321-327 (quoted in [7]).
- [5] F. R. HAMPEL (1973), *Robust estimation: a condensed partial survey*, *Z. Wahrsch. Verw. Geb.*, 27, pp. 87-104.
- [6] P. W. HOLLAND AND R. E. WELSCH (1977), *Robust regression using iteratively reweighted least squares*, *Comm. Statist.*, A6 (9), pp. 813-827.
- [7] P. J. HUBER (1972), *Robust statistics: a review*, *Ann. Math. Statist.*, 43, pp. 1041-1067.
- [8] ——— (1973), *Robust regression: asymptotics, conjectures and Monte Carlo*, *Ann. Statist.*, 1, pp. 799-821.
- [9] ——— (1974), *Comment on adaptive robust procedures*, *J. Amer. Statist. Assoc.*, 69, pp. 926-927.
- [10] ——— (1977), *Robust Statistical Procedures*, CBMS Regional Conference Series in Applied Mathematics 27, Society for Industrial and Applied Mathematics, Philadelphia.

- [11] P. J. HUBER AND R. DUTTER (1974), *Numerical solution of robust regression problems*, COMPSTAT 1974, Proc. Symposium on Computational Statistics, pp. 165–172.
- [12] J. W. TUKEY (1960), *A survey of sampling from contaminated distributions*, Contributions to Probability and Statistics, I. Olkin, ed., Stanford Univ. Press, Stanford, CA.
- [13] G. WATSON (1980), *Approximation Theory and Numerical Methods*, John Wiley, New York.

## BLOCK PRECONDITIONING FOR THE CONJUGATE GRADIENT METHOD\*

P. CONCUS<sup>†</sup>, G. H. GOLUB<sup>‡</sup> AND G. MEURANT<sup>§</sup>

**Abstract.** Block preconditionings for the conjugate gradient method are investigated for solving positive definite block tridiagonal systems of linear equations arising from discretization of boundary value problems for elliptic partial differential equations. The preconditionings rest on the use of sparse approximate matrix inverses to generate incomplete block Cholesky factorizations. Carrying out of the factorizations can be guaranteed under suitable conditions. Numerical experiments on test problems for two dimensions indicate that a particularly attractive preconditioning, which uses special properties of tridiagonal matrix inverses, can be computationally more efficient for the same computer storage than other preconditionings, including the popular point incomplete Cholesky factorization.

**Key words.** conjugate gradient method, elliptic partial differential equations, incomplete factorization, iterative methods, preconditioning, sparse matrices

**1. Introduction.** In this paper we study some preconditioning techniques for the conjugate gradient method to solve the linear systems of equations that arise from the discretization of partial differential equations. We consider for example elliptic equations such as

$$-\sum_{i=1}^d \frac{\partial}{\partial \xi_i} \left[ \lambda_i(\xi) \frac{\partial u}{\partial \xi_i} \right] + \sigma(\xi)u = f \quad \text{in } \Omega \subset R^d, \quad \xi = (\xi_1, \xi_2, \dots, \xi_d) \quad (1)$$

with

$$u(\xi) = g(\xi) \quad \text{or} \quad \frac{\partial u}{\partial n} = g(\xi) \quad \text{on } \partial\Omega,$$

where  $n$  is the exterior normal,  $\lambda_i(\xi) > 0$ , and  $\sigma(\xi) \geq 0$ . The techniques that we describe are suitable for standard finite-difference discretizations of equations such as the above that yield certain symmetric positive definite block tridiagonal linear systems of the form

$$Ax = b, \quad (2)$$

where

$$A = \begin{pmatrix} D_1 & A_2^T & & & & & & \\ A_2 & D_2 & A_3^T & & & & & \\ & & \ddots & \ddots & \ddots & & & \\ & & & A_{n-1} & D_{n-1} & A_n^T & & \\ & & & & A_n & D_n & & \end{pmatrix}.$$

\*Received by the editors May 2, 1983, and in revised form January 16, 1984. This paper is an abridged version of [2], which was presented at the SIAM 30th Anniversary Meeting, Stanford University, Stanford, California, 1982. It was typeset at the Lawrence Berkeley Laboratory using a *troff* program running under UNIX. The final copy was produced on July 6, 1984. This work was supported in part by the Applied Mathematical Sciences subprogram of the Office of Energy Research, U.S. Department of Energy under contracts DE-AC03-76SF00098 and DE-AC03-76SF00515 and by the National Science Foundation under grant MCS-78-11985.

<sup>†</sup>Lawrence Berkeley Laboratory and Department of Mathematics, University of California, Berkeley, California 94720.

<sup>‡</sup>Computer Science Department, Stanford University, Stanford, California 94305.

<sup>§</sup>Commissariat à l'Energie Atomique, Limeil 94190 Villeneuve-Saint-Georges, France, and Computer Science Department, Stanford University, Stanford, California 94305.

Such equations can arise also from finite element discretizations (for example, see [11]).

The prototype model problem in two dimensions is the Dirichlet problem,  $\sigma \equiv 0$ ,  $\lambda_i \equiv 1$ ,  $g \equiv 0$ ,  $\Omega$  the unit square,

$$-\Delta u = f,$$

$$u = 0 \quad \text{on the boundary,}$$

with standard five-point differencing on a uniform mesh of width  $h$ . We focus attention on the matrix structure obtained for natural ordering, which yields for the model problem (after multiplication by  $h^2$ )

$$A_i = -I, \quad D_i = \begin{pmatrix} 4 & -1 & & & \\ -1 & 4 & -1 & & \\ & \ddots & \ddots & \ddots & \\ & & -1 & 4 & -1 \\ & & & -1 & 4 \end{pmatrix}.$$

In three dimensions, standard 7-point differencing with this ordering would yield  $D_i$  that have two additional nonzero diagonals. Different orderings or higher order approximations would give rise to different structures, to which our techniques could be applied also.

To solve (2) we use the generalized or preconditioned conjugate gradient method, which may be written as follows [3]. Let  $x^0$  be given, define  $p^{-1}$  arbitrarily, and let  $r^0 = b - Ax^0$ . For  $k = 0, 1, \dots$  perform the steps

$$Mz^k = r^k,$$

$$\beta_k = \frac{(z^k, Mz^k)}{(z^{k-1}, Mz^{k-1})}, \quad k \geq 1, \quad \beta_0 = 0,$$

$$p^k = z^k + \beta_k p^{k-1},$$

$$\alpha_k = \frac{(z^k, Mz^k)}{(p^k, Ap^k)},$$

$$x^{k+1} = x^k + \alpha_k p^k,$$

$$r^{k+1} = r^k - \alpha_k Ap^k.$$

The matrix  $M$  is the preconditioning matrix, which should be in some sense an approximation of  $A$ . It is known that the preconditioned conjugate gradient method converges rapidly if the condition number  $\kappa(M^{-1}A)$ , which is the ratio of the largest to the smallest eigenvalue of  $M^{-1}A$ , is small or if the eigenvalues are clustered (e.g., [3] and the references therein).

The goal of this study is to devise good preconditioning matrices  $M$ . For this purpose we exploit the structure of  $A$  in constructing some block preconditionings, one special case of which is the one introduced by R. R. Underwood [20].

In §2 to motivate the use of our block techniques we recall some results on block Cholesky factorization. Section 3 deals with the main problem—finding good approximate inverses for tridiagonal matrices that are diagonally dominant.

New block techniques for two-dimensional problems ( $d = 2$  in (1)) are introduced in §4. Three-dimensional problems will be discussed in detail in a subsequent study.

In §5 we present the results of numerical experiments for several test problems, including comparisons with point preconditioning techniques. We compare techniques on the basis of number of iterations and number of floating point operations required for convergence. Also, we illustrate graphically the spectral properties of the matrices corresponding to the various preconditionings.

**2. Block Cholesky factorization.** Let  $A$  be the symmetric positive definite block tridiagonal matrix of (2). Let  $m_i$  be the order of the  $i^{th}$  square diagonal block  $D_i$  and  $N = \sum_{i=1}^n m_i$  the order of  $A$ . We denote

$$D = \begin{pmatrix} D_1 & & & & & \\ & D_2 & & & & \\ & & \ddots & & & \\ & & & D_{n-1} & & \\ & & & & D_n & \end{pmatrix}, \quad L = \begin{pmatrix} 0 & & & & & \\ A_2 & 0 & & & & \\ & \ddots & \ddots & & & \\ & & & A_{n-1} & 0 & \\ & & & & A_n & 0 \end{pmatrix},$$

$$A = D + L + L^T,$$

and we denote by  $a_{ij}$  the elements (pointwise) of  $A$ .

Since  $A$  is positive definite, there holds  $a_{ii} > 0, i = 1, \dots, N$ . We assume that the following holds also.

HYPOTHESIS (H1).

- (a) *The off-diagonal elements  $a_{ij}, i \neq j$  of  $A$  are nonpositive.*
- (b)  *$A$  is (weakly) diagonally dominant; i.e., there holds*

$$a_{ii} \geq \sum_{j \neq i} |a_{ij}|, \quad i = 1, \dots, N,$$

and there exists at least one  $k, 1 \leq k \leq N$ , such that

$$a_{kk} > \sum_{j \neq k} |a_{kj}|.$$

- (c) *Each column of  $A_i, i = 2, \dots, n$ , has at least one nonzero element.*

Hypothesis (H1)(a) implies that the positive definite symmetric matrix  $A$  is a Stieltjes matrix, i.e., a positive definite  $M$ -matrix [21], [22].

If the inequality of Hypothesis (H1)(b) holds strictly for all rows,

$$a_{ii} > \sum_{j \neq i} |a_{ij}|, \quad i = 1, \dots, N,$$

then  $A$  is termed *strictly diagonally dominant*.

Let  $\Sigma$  be the symmetric block diagonal matrix with  $m_i \times m_i$  blocks  $\Sigma_i$  satisfying

$$\begin{aligned} \Sigma_1 &= D_1, \\ \Sigma_i &= D_i - A_i \Sigma_{i-1}^{-1} A_i^T, \quad 2 \leq i \leq n. \end{aligned} \tag{3}$$

Then the block Cholesky factorization of  $A$  can be written as

$$A = (\Sigma + L)\Sigma^{-1}(\Sigma + L^T).$$

The factor  $\Sigma + L$  is block lower bidiagonal. Since  $A$  is positive definite symmetric, the factorization can be carried out.

The following results concerning the properties of the  $\Sigma_i$  are well known, but as we did not find them in the literature in a form suitable for our application, we give them here for completeness. These properties provide guidance in our selection of preconditioning matrices for the conjugate gradient method.

Let

$$B = \begin{pmatrix} B_1 & -C^T \\ -C & B_2 \end{pmatrix},$$

with  $B_1$  and  $B_2$  square, be a symmetric positive definite  $M$ -matrix, which implies that the diagonal elements are positive and the off-diagonal elements are nonpositive.

LEMMA 1.  $B'_2 = B_2 - CB_1^{-1}C^T$  is a symmetric positive definite  $M$ -matrix.

$B'_2$  is called the Schur complement of  $B_1$  in  $B$ . For properties of the Schur complement see [4].

*Proof.* We can write

$$\begin{pmatrix} B_1 & 0 \\ 0 & B_2 - CB_1^{-1}C^T \end{pmatrix} = \begin{pmatrix} I & 0 \\ CB_1^{-1} & I \end{pmatrix} B \begin{pmatrix} I & B_1^{-1}C^T \\ 0 & I \end{pmatrix}.$$

Since the leading principal minors of  $B$  are unchanged by the transformation on the right side of the equality, the matrix on the left side is positive definite, and hence so is  $B'_2$ . In particular the diagonal elements of  $B'_2$  are positive and, as  $B_1^{-1} > 0$  and  $C \geq 0$  hold, it follows that the off diagonal elements are nonpositive.

It can be shown easily that if  $B_2$  is strictly diagonally dominant, then  $B'_2$  is also.

Now we apply these results to  $A$  with  $B_1 = D_1$ ,  $-C^T = (A_2^T \ 0 \ \dots \ 0)$ , and

$$B_2 = \begin{pmatrix} D_2 & A_3^T & & & \\ A_3 & D_3 & A_4^T & & \\ & \ddots & \ddots & \ddots & \\ & & A_{n-1} & D_{n-1} & A_n^T \\ & & & A_n & D_n \end{pmatrix}.$$

We have

$$B'_2 = \begin{pmatrix} D_2 - A_2 D_1^{-1} A_2^T & A_3^T & & & \\ A_3 & D_3 & A_4^T & & \\ & \ddots & \ddots & \ddots & \\ & & A_{n-1} & D_{n-1} & A_n^T \\ & & & A_n & D_n \end{pmatrix}.$$

There follows

THEOREM 1. Under Hypothesis (H1) all the  $\Sigma_i$  are symmetric strictly diagonally dominant  $M$ -matrices.

It is of interest to note, that in the particular case of the model problem, the block Cholesky factorization can be shown to reduce to a Fast Poisson Solver [18].

**3. Incomplete block Cholesky factorization.** Because of the work and storage that may be required in large problems for computing the  $\Sigma_i$ , carrying out the complete block Cholesky factorization is not of interest to us here as a general means for solving (2). For example, for the two-dimensional model problem, although  $\Sigma_1 = D_1$  is tridiagonal,  $\Sigma_1^{-1}$  and hence  $\Sigma_i, i \geq 2$ , are dense.

In this paper our interest focuses on approximate block Cholesky factorizations obtained by using in (3) instead of  $\Sigma_i^{-1}$  a sparse approximation  $\Lambda_{i-1}$ . One thereby obtains instead of  $\Sigma$  the block diagonal matrix  $\Delta$  with  $m_i \times m_i$  blocks  $\Delta_i$  satisfying

$$\Delta_1 = D_1, \tag{4a}$$

$$\Delta_i = D_i - A_i \Lambda_{i-1} A_i^T, \quad 2 \leq i \leq n, \tag{4b}$$

where for each  $i$  in (4b),  $\Lambda_{i-1}$  is the sparse approximation to  $\Delta_{i-1}^{-1}$ . The incomplete block Cholesky preconditioning matrix for use with the conjugate gradient algorithm is then

$$M = (\Delta + L)\Delta^{-1}(\Delta + L^T). \tag{5}$$

One has

$$M = A + \Delta - D + L\Delta^{-1}L^T = A + R,$$

where  $R$  is a block diagonal matrix

$$R = \begin{pmatrix} R_1 & & & & & \\ & R_2 & & & & \\ & & \ddots & & & \\ & & & R_{n-1} & & \\ & & & & R_n & \end{pmatrix}$$

with

$$R_1 = \Delta_1 - D_1 = 0,$$

$$R_i = \Delta_i - D_i + A_i \Delta_{i-1}^{-1} A_i^T, \quad 2 \leq i \leq n.$$

The factor  $\Delta + L$  in (5) is lower block bidiagonal. Using the Cholesky factors  $L_i$  of  $\Delta_i$ ,

$$\Delta_i = L_i L_i^T,$$

one can express  $M$  in terms of (point) lower and upper triangular factors

$$M = \begin{bmatrix} L_1 & & & & & \\ W_2 & L_2 & & & & 0 \\ & \ddots & \ddots & & & \\ & & & W_{n-1} & L_{n-1} & \\ & & & & W_n & L_n \end{bmatrix} \begin{bmatrix} L_1^T & W_2^T & & & & \\ & L_2^T & W_3^T & & & \\ & & \ddots & \ddots & & \\ & & & L_{n-1}^T & W_n^T & \\ & & & & L_n^T & \end{bmatrix}, \tag{6}$$

where

$$W_i = A_i L_{i-1}^{-T}, \quad i = 2, \dots, n.$$

This form is generally more efficient computationally than is (5). For specific  $\Lambda_i$  of interest, we show in subsequent sections that all the  $\Delta_i$  are positive definite,



which implies that the above factorization can be carried out.

Note that in the conjugate gradient algorithm  $M$  is not required explicitly, only the linear system  $Mz^k = r^k$  need be solved for  $z^k$ . Since this can be done with block backward and forward substitution, the block off-diagonal elements  $W_i$  need not be computed explicitly. The requisite products with vectors can be obtained by solving linear systems with triangular coefficient matrices  $L_i$  and  $L_i^T$ . Generally, for preconditionings of interest, the  $\Delta_i$ , and correspondingly the  $L_i$ , will be sparse. These features were first used in this context by R. R. Underwood in [20], where block incomplete Cholesky preconditioning for the conjugate gradient algorithm was introduced.

For the standard five-point discretization of (1) in two dimensions,  $D_i$  is tridiagonal, and  $A_i$  is diagonal. This is the case on which this paper focuses. Of central interest is the choice that the  $\Lambda_{i-1}$  be tridiagonal, so that all the  $\Delta_i$  in (4b) are tridiagonal. Correspondingly, in the remainder of this section we discuss techniques for approximating the inverse of a tridiagonal, diagonally-dominant matrix.

Let

$$T = \begin{pmatrix} a_1 & -b_1 & & & & \\ -b_1 & a_2 & & & & \\ & \ddots & \ddots & & & \\ & & & -b_{m-2} & a_{m-1} & -b_{m-1} \\ & & & & -b_{m-1} & a_m \end{pmatrix} \tag{7}$$

be a nonsingular tridiagonal matrix. We assume that the following holds.

**HYPOTHESIS (H2).** *The elements  $a_i$  and  $b_i$  of  $T$  satisfy*

$$\begin{aligned} a_i &> 0, \quad 1 \leq i \leq m, \\ b_i &> 0, \quad 1 \leq i \leq m - 1, \end{aligned}$$

and  $T$  is strictly diagonally dominant.

**3.1. Diagonal approximation.** The simplest approximation  $\tilde{T}_1$  of  $T^{-1}$  we consider is the diagonal matrix whose elements are

$$(\tilde{T}_1)_{ii} = \frac{1}{(T)_{ii}}. \tag{8}$$

**3.2. Banded approximation from the exact inverse.** One can do much better than the diagonal approximation  $\tilde{T}_1$  by using the following powerful result, which characterizes the inverses of symmetric tridiagonal matrices, (cf. [1], [10]).

**THEOREM 2.** *There exist two vectors  $u$  and  $v \in R^m$  such that*

$$(T^{-1})_{ij} = u_i v_j \quad \text{for } i \leq j.$$

Since the inverse of  $T$  is

$$T^{-1} = \begin{pmatrix} u_1 v_1 & u_1 v_2 & \cdots & u_1 v_m \\ u_1 v_2 & u_2 v_2 & \cdots & u_2 v_m \\ \cdot & \cdot & & \cdot \\ \cdot & \cdot & & \cdot \\ u_1 v_m & u_2 v_m & \cdots & u_m v_m \end{pmatrix},$$

one can compute recursively the components of  $u$  and  $v$ . Under Hypothesis (H2)  $T$  is positive definite, so that  $T^{-1}$  is also, which implies that  $u_i \neq 0, v_i \neq 0$ , for all  $i$ . We remark that all of Hypothesis (H2), which will be used later, is not required for Theorem 2. It is necessary only that  $T$  in (7) be nonsingular and irreducible (all of the  $b_i$  nonzero).

LEMMA 2. *The components of  $u$  and  $v$  can be computed as follows:*

$$\begin{aligned} u_1 &= 1, \quad u_2 = \frac{a_1}{b_1}, \\ u_i &= \frac{a_{i-1}u_{i-1} - b_{i-2}u_{i-2}}{b_{i-1}}, \quad 3 \leq i \leq m, \\ v_m &= \frac{1}{-b_{m-1}u_{m-1} + a_m u_m}, \\ v_i &= \frac{1 + b_i u_i v_{i+1}}{a_i u_i - b_{i-1} u_{i-1}}, \quad 2 \leq i \leq m-1, \\ v_1 &= \frac{1 + b_1 u_1 v_2}{a_1 u_1}. \end{aligned} \tag{9}$$

*Proof.* By substitution.

Alternative recurrences for generating  $u$  and  $v$  can be obtained by several means, such as by computing the first and last columns of  $T^{-1}$  from the Cholesky factors of  $T$ . For numerical computation scaling may be required in (9) to prevent underflow or overflow, or it may be desirable to work with the ratios  $u_{i+1}/u_i$  and  $v_{i+1}/v_i$  considered below. If only a few of the main diagonals of  $T^{-1}$  are required and not  $u$  and  $v$  explicitly, the diagonals can be computed conveniently from the Cholesky factors of  $T$ .

Several papers have characterized the elements of inverses of diagonally dominant matrices. In [15] results are proved for tridiagonal matrices and in [5] they are extended to matrices of larger bandwidth. It is known that the elements of  $(T^{-1})_{ij}$  are bounded in an exponentially decaying manner along each row or column. Specifically, there exist  $\rho < 1$  and a constant  $C_0$  such that

$$(T^{-1})_{ij} \leq C_0 \rho^{|i-j|}.$$

This result does not imply that the elements actually decay along each row; it merely provides a bound. With Hypothesis (H2), however, one can prove the following:

LEMMA 3. *Under Hypothesis (H2) the sequence  $\{u_i\}_{i=1}^m$  is strictly increasing and the sequence  $\{v_i\}_{i=1}^m$  is strictly decreasing.*

*Proof.* It is clear that  $u_2 = a_1/b_1 > 1 = u_1$ . The proof continues by induction, using formulas of Lemma 2. Since  $u_{i-1} > u_{i-2}$ , one has from (9) that

$$u_i > u_{i-1} \left( \frac{a_{i-1} - b_{i-2}}{b_{i-1}} \right) > u_{i-1},$$

because  $a_{i-1} - b_{i-1} - b_{i-2} > 0$ . To prove that the  $v_i$  are decreasing we need to modify the formulas of Lemma 2 slightly, using the ones for  $u$  to simplify those for  $v$ . Note that

$$a_i u_i - b_{i-1} u_{i-1} - b_i u_{i+1} = 0$$

and

$$(a_{i+1}u_{i+1} - b_i u_i)v_{i+1} = 1 + b_{i+1}u_{i+1}v_{i+2}.$$

Thus

$$v_i = \frac{a_{i+1}u_{i+1}v_{i+1} - b_{i+1}u_{i+1}v_{i+2}}{b_i u_{i+1}} = \frac{a_{i+1}}{b_i} v_{i+1} - \frac{b_{i+1}}{b_i} v_{i+2}, \text{ for } i \leq m-2,$$

and

$$v_{m-1} = \frac{a_m}{b_{m-1}} v_m.$$

Clearly  $v_{m-1} > v_m$ , and by induction  $v_i > v_{i+1} ((a_{i+1} - b_{i+1})/b_i) > v_{i+1}$ .

Note that we can prove the same result if we suppose only that  $T$  is diagonally dominant with  $a_1 > b_1$  and  $a_m > b_{m-1}$ .

One can characterize the decay of the element along a row away from the diagonal. Let  $\bar{\alpha}_i$  and  $\bar{\beta}_i$  be such that  $u_i = \bar{\alpha}_{i-1}u_{i-1}$ ,  $v_i = v_{i-1}/\bar{\beta}_{i-1}$ ,  $i \geq 2$ . We have

$$\begin{aligned} \bar{\alpha}_i &= \frac{a_i}{b_i} - \frac{b_{i-1}}{b_i} \frac{1}{\bar{\alpha}_{i-1}}, & \bar{\alpha}_1 &= \frac{a_1}{b_1}, \\ \bar{\beta}_i &= \frac{a_{i+1}}{b_i} - \frac{b_{i+1}}{b_i} \frac{1}{\bar{\beta}_{i+1}}, & \bar{\beta}_{m-1} &= \frac{a_m}{b_{m-1}}. \end{aligned}$$

In the general case we do not know the solution of the recurrences (which are simply the recurrences for computing the elements of  $T^{-1}$ ), but the previous discussion gives us the bounds

$$\begin{aligned} \bar{\alpha}_i &> \frac{a_i - b_{i-1}}{b_i} > 1, \\ \bar{\beta}_i &> \frac{a_{i+1} - b_{i+1}}{b_i} > 1. \end{aligned}$$

In particular, we have, for  $i > j$ ,

$$(T^{-1})_{ij} = \frac{1}{\bar{\alpha}_{i-1} \cdots \bar{\alpha}_j} (T^{-1})_{ii} \leq \frac{(T^{-1})_{ii}}{\prod_{k=j}^{i-1} \left( \frac{a_k - b_{k-1}}{b_k} \right)}.$$

If  $1/\rho = \min_{k \geq 2} ((a_k - b_{k-1})/b_k)$  we find, for  $i > j$ ,

$$(T^{-1})_{ij} \leq (T^{-1})_{ii} \rho^{i-j}, \quad \rho < 1.$$

This latter bound is not very sharp. For example, for the matrix  $T$  with  $a_i = 4$ ,  $i = 1, \dots, m$  and  $b_i = 1$ ,  $i = 1, \dots, m-1$ , which will be of interest later, we get  $\rho = 1/3$ . But for this case

$$\bar{\alpha}_1 = 4, \quad \bar{\alpha}_i = 4 - \frac{1}{\bar{\alpha}_{i-1}}, \quad i \geq 2.$$

The  $\bar{\alpha}_i$  form a decreasing sequence that converges very quickly towards  $2 + \sqrt{3} \approx 3.732$ , which corresponds to a reduction factor of  $1/(2 + \sqrt{3}) \approx 0.2679$ , which is considerably less than  $1/3$ . Of course if the  $a_i$ 's and  $b_i$ 's are constant, we could construct the inverse in another way from the eigenvalues and eigenvectors of  $T$ , which are known in this case.

It is of importance to observe that if  $T$  is strictly diagonally dominant the elements of the inverse decrease strictly away from the diagonal -- the stronger the diagonal dominance the faster the decay. This suggests the following means for approximating the inverse of  $T$  with a matrix of small bandwidth.

If  $A$  is any matrix, denote by  $\mathbf{B}(A, p)$  the band matrix consisting of the  $2p + 1$  main diagonals of  $A$ . For a banded approximation  $\tilde{T}_2(p)$  to the inverse of  $T$  we consider

$$\tilde{T}_2(p) = \mathbf{B}(T^{-1}, p) \tag{10}$$

with  $p$  small, say 1 or 2.

**3.3. Approximation from Cholesky factors.** Another way of approximating  $T^{-1}$  is to use the Cholesky factorization of  $T$ ,

$$T = U^T U,$$

with

$$U^T = \begin{pmatrix} \gamma_1 & & & & \\ -\delta_1 & \gamma_2 & & & 0 \\ & \ddots & \ddots & & \\ & & & -\delta_{m-2} & \gamma_{m-1} \\ & & & & -\delta_{m-1} & \gamma_m \end{pmatrix}$$

a lower bidiagonal matrix. We have

$$\begin{aligned} \gamma_1^2 &= a_1, & \gamma_1 \delta_1 &= b_1, \\ \delta_{i-1}^2 + \gamma_i^2 &= a_i, & \gamma_i \delta_i &= b_i, \quad i \geq 2. \end{aligned}$$

The  $\delta_i$ 's are positive, and the diagonal dominance of  $T$  implies

$$\delta_i < \gamma_i, \quad 1 \leq i \leq m-1.$$

The matrix  $U^{-T}$  is lower triangular and dense. We denote

$$U^{-T} = \begin{pmatrix} \frac{1}{\gamma_1} & & & & \\ \xi_1 & \frac{1}{\gamma_2} & & & 0 \\ \eta_1 & \xi_2 & \frac{1}{\gamma_3} & & \\ \vdots & \ddots & \ddots & \ddots & \\ \vdots & & & \dots & \eta_{m-2} & \xi_{m-1} & \frac{1}{\gamma_m} \end{pmatrix}.$$

It is easy to see that the elements of  $U^{-T}$  can be computed diagonal by diagonal, since

$$\xi_i = \frac{\delta_i}{\gamma_i \gamma_{i+1}}, \quad 1 \leq i \leq m-1, \quad \eta_i = \frac{\delta_i \xi_{i+1}}{\gamma_i}, \quad 1 \leq i \leq m-2,$$

and so on. We note also that  $U^{-T}$  can be generated diagonal by diagonal by taking successive terms of its Neumann series in  $U^T$ .

We have the following result similar to the one for the inverse of  $T$ .

LEMMA 4. *For each row, the elements of  $U^{-T}$  decrease away from the diagonal.*

*Proof.* Since  $\delta_i/\gamma_i < 1$  we have  $\eta_{i-1} < \zeta_i < 1/\gamma_{i+1}$ ; the proof is the same for the other elements.

As an approximation for  $U^{-T}$  we can, therefore, take  $\mathbf{B}(U^{-T}, p)$  with  $p$  small. As an approximation for  $T^{-1}$  we can use correspondingly

$$\tilde{T}_3(p) = \mathbf{B}(U^{-1}, p)\mathbf{B}(U^{-T}, p). \tag{11}$$

Note that  $\tilde{T}_3(p)$  is positive definite. For  $p=1$ , one has the tridiagonal matrix

$$\tilde{T}_3(1) = \begin{pmatrix} \frac{1}{\gamma_1^2} + \zeta_1^2 & \frac{\zeta_1}{\gamma_2} & & & \\ \frac{\zeta_1}{\gamma_2} & \frac{1}{\gamma_2^2} + \zeta_2^2 & \frac{\zeta_2}{\gamma_3} & & \\ & \ddots & \ddots & \ddots & \\ & & & & \ddots \end{pmatrix}.$$

Unless the Cholesky decomposition is needed explicitly, it is necessary to compute only the square of the  $\gamma_i$ 's to obtain  $\tilde{T}_3(1)$ , because

$$\frac{\zeta_i}{\gamma_{i+1}} = \frac{b_i}{\gamma_i^2 \gamma_{i+1}^2}, \quad \zeta_i^2 = \frac{a_{i+1} - \gamma_{i+1}^2}{\gamma_i^2 \gamma_{i+1}^2}.$$

Thus one obtains  $\tilde{T}_3(1)$  directly from  $a_i$ ,  $b_i$ , and  $\gamma_i^2$ .

Note that  $\tilde{T}_3(1)T$  is the five diagonal matrix

$$\tilde{T}_3(1)T = \begin{pmatrix} 1 & -b_2u_1v_3 & b_2u_1v_2 & & & \\ -b_2u_1v_3 & 1 & -b_3u_2v_4 & b_3u_2v_3 & & \\ b_2u_1v_2 & -b_3u_2v_4 & 1 & -b_4u_3v_3 & b_4u_3v_4 & \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \end{pmatrix}.$$

Since the  $u_i v_j$  are expected to be small,  $\tilde{T}_3(1)$  can be expected to be a good approximation to  $T^{-1}$ .

**3.4. Polynomial approximation.** A classical way to obtain an approximation of  $T^{-1}$  is to use a polynomial expansion in powers of  $T$ . Let  $D_T$  be the diagonal of  $T$  and denote

$$\bar{T} = T - D_T.$$

Then

$$T^{-1} = (I + D_T^{-1}\bar{T})^{-1}D_T^{-1}.$$

Since  $T$  is strictly diagonally dominant, the corresponding Jacobi iteration is convergent, which implies that the eigenvalues of the Jacobi iteration matrix  $-D_T^{-1}\bar{T}$  (which are real) are contained in  $(-1,+1)$  (see for example [12], [21]). Thus one can write

$$(I + D_T^{-1}\bar{T})^{-1} = \sum_{k=0}^{\infty} (-1)^k (D_T^{-1}\bar{T})^k,$$

the series being convergent.

The powers of  $D_T^{-1}\bar{T}$  contain more and more nonzero diagonals as  $k$  increases. As an approximate inverse we can take simply the first few terms, which are the sparsest ones (Taking only the first term gives the diagonal approximation  $\bar{T}_1$  of §3.1.). It is well known, however, that if the eigenvalues of  $D_T^{-1}\bar{T}$  are not close enough to zero, the truncated series could be a poor approximation. Better polynomial approximations can be found (cf. [14]).

Let  $S = D_T^{-1}\bar{T}$ , and suppose we want to find a polynomial  $P$  of degree less than or equal to  $\nu$  that minimizes  $\|(I+S)^{-1} - P(S)\|_2$ . Since  $S$  is similar to a symmetric matrix there exists a unitary matrix  $Q$  such that

$$S = Q\Theta Q^T,$$

where  $\Theta$  is a diagonal matrix whose elements are the eigenvalues of  $S$ .

We have

$$P(S) = QP(\Theta)Q^T,$$

so that

$$\|(I+S)^{-1} - P(S)\|_2 = \|(I+\Theta)^{-1} - P(\Theta)\|_2 \leq C_1 \max_i \left\| \frac{1}{1+\theta_i} - P(\theta_i) \right\|,$$

where  $C_1$  is constant and  $\theta_i$ ,  $1 \leq i \leq m$ , are the eigenvalues of  $S$ . To minimize the right-hand side (the minimum, of course, need not minimize also the left-hand side) we must find the polynomial approximation of  $1/(1+x)$  on the set of eigenvalues  $\theta_i$  of  $S$ . Instead we could solve the simpler problem of finding

$$\min \max_{\theta \in [\theta_1, \theta_m]} \left\| \frac{1}{1+\theta} - P(\theta) \right\|,$$

where  $\theta_1$  (respectively,  $\theta_m$ ) is the smallest (respectively, largest) eigenvalue of  $S$ . The solution to this problem is given by Chebyshev polynomials.

In general, however, even the extremal eigenvalues  $\theta_1$  and  $\theta_m$  are not known; all one knows is that  $-1 < \theta_1 \leq \theta_m < 1$  holds. Since  $1/(1+x)$  is discontinuous at  $x = -1$ , we could simply compute  $P$  to yield

$$\min_P \max_{\theta \in [0, 1]} \left\| \frac{1}{1+\theta} - P(\theta) \right\|.$$

This should give a good result for the eigenvalues between 0 and 1, but a poor one for the smaller eigenvalues. For a first degree polynomial we obtain

$$P(\Theta) \approx 0.9412 - 0.4706 \Theta.$$

As will be seen later, it is possible to obtain a better approximation when additional information about the eigenvalues is available. In general, we shall be considering tridiagonal polynomial approximations  $\tilde{T}$  to  $T^{-1}$  of the form

$$\tilde{T}_4(\alpha, \beta) = \alpha D_T^{-1} + \beta D_T^{-1} \bar{T} D_T^{-1}, \tag{12}$$

where the coefficients  $\alpha$  and  $\beta$  are real.

**3.5. Comparison of approximations for the model problem.** We now compare the above approximations for the model problem, for which in (7)  $a_i = 4$ ,  $i = 1, \dots, m$ , and  $b_i = 1$ ,  $i = 1, \dots, m-1$ . The case  $m = 10$  is considered. The upper triangular part of the inverse  $T^{-1}$  as computed in double precision

FORTTRAN on an IBM 3081 by MATLAB [19] to four places is

$$\begin{bmatrix} 0.2679 & 0.0718 & 0.0192 & 0.0052 & 0.0014 & 0.0004 & \dots \\ & 0.2872 & 0.0770 & 0.0206 & 0.0055 & 0.0015 & \dots \\ & & 0.2886 & 0.0773 & 0.0207 & 0.0056 & \dots \\ & & & 0.2887 & 0.0773 & 0.0207 & \dots \\ & & & & 0.2887 & 0.0773 & \dots \\ & & & & & 0.2887 & \dots \\ & & & & & & \dots & \dots \end{bmatrix},$$

illustrating the rapid decay away from the diagonal.

For the different approximations  $\tilde{T}_i$  to  $T^{-1}$  we get the following results (using MATLAB), as summarized in Table 1 and Figure 1. The last entry

TABLE 1  
Values of  $\|\tilde{T}_i - T^{-1}\|_2$  for the model problem,  $m = 10$ .

Approximation to $T^{-1}$		$\ \tilde{T}_i - T^{-1}\ _2$
Diagonal (§3.1)	$\tilde{T}_1$	0.2305
Banded from exact inverse (§3.2)	$\tilde{T}_2(1)$	0.0456
Banded from exact inverse (§3.2)	$\tilde{T}_2(2)$	0.0104
From Cholesky factors (§3.3)	$\tilde{T}_3(1)$	0.0569
From Cholesky factors (§3.3)	$\tilde{T}_3(2)$	0.0134
Polynomial (§3.4)	$\tilde{T}_4(1, -1)$	0.1106
Polynomial (§3.4)	$\tilde{T}_4(.9412, -.4706)$	0.1888
Polynomial (§3.4)	$\tilde{T}_4(1.1429, -1.1429)$	0.0577

	.5	.6	.7	.8	.9	1.0	1.1	1.2	1.3	1.4
$\tilde{T}_1$	2,8	7	9		3	7		1	3	2,8
$\tilde{T}_2(1)$				3,9	1,5,9	1,7,8	3,4			
$\tilde{T}_2(2)$					6,7,8,9	0,0,1,3,3,4				
$\tilde{T}_3(1)$				8,8	3,3	0,0,7,8	3,3			
$\tilde{T}_3(2)$					7,7,7,9	0,0,1,3,3,3				
$\tilde{T}_4$ (1, -1)			7,7	2,2,9,9	6,6,9,9					
$\tilde{T}_4$ (.9412, -.4706)		1,6	4	2	1,7	2,4,6,6				
$\tilde{T}_4$ (1.1429, -1.1429)				8,8	4,4	2,2,9,9	4,4			

FIG.1. Tabular display of eigenvalue distributions of  $\tilde{T}_i T$ .

$\tilde{T}_4(1.1429, -1.1429)$  corresponds to the min max polynomial over  $\theta \in [-0.5, 0.5]$ , which interval is approximately the one bounded by the extremal eigenvalues  $\theta_1 \approx -0.4797$ ,  $\theta_m \approx 0.4797$  of  $S$  for this problem. Thus one should expect this polynomial to give a better approximation than the other two. Values of  $\|\tilde{T}_i - T^{-1}\|_2$  to four places are given in Table 1. Figure 1 depicts the eigenvalues of  $\tilde{T}_i T$  in a format that permits a rough comparison of their distributions: The eigenvalues are rounded to two decimal places, and the least significant digit is entered in the column corresponding to the first digit(s).

It is evident that for this model problem the banded approximations from the exact inverse and the approximations from Cholesky factors can give better approximations to  $T^{-1}$  than the polynomial expansions, in the sense of clustering about 1 of the eigenvalues of  $\tilde{T}_i T$  and smallness of  $\|\tilde{T}_i - T^{-1}\|_2$ . It would be of interest to know if the same results would hold for matrices  $T$  of larger bandwidth.

**4. Block preconditionings for the two-dimensional case.** Based on the approximate inverses of §3, we define the corresponding block preconditionings for the two-dimensional problem. For this case the  $D_i$  are tridiagonal, and our goal is to keep the  $\Delta_i$ ,  $i \geq 2$ , in (4b) tridiagonal, or possibly of slightly greater bandwidth. For the preconditionings discussed below, only the Cholesky factors  $L_i$  of the  $\Delta_i$  are actually stored for computational purposes, corresponding to (6).

#### 4.1. The block preconditionings.

**4.1.1. BDIA.** The diagonal approximation  $\tilde{T}_1$  in (8) is used;  $\Lambda_{i-1}$  is diagonal with

$$(\Lambda_{i-1})_{jj} = \frac{1}{(\Delta_{i-1})_{jj}}.$$

The  $\Delta_i$ 's are tridiagonal matrices at each stage differing from  $D_i$  only in their diagonal elements.

**4.1.2. INV(1).** The banded approximation  $\tilde{T}_2(1)$  in (10) from the exact inverse is used,

$$\Lambda_{i-1} = \mathbf{B}(\Delta_{i-1}^{-1}, 1).$$

Each of the  $\Delta_i$ 's is tridiagonal. At each stage we compute two vectors  $u$  and  $v$  and use them to obtain the three main diagonals of  $\Delta_{i-1}^{-1}$ . We then compute and store the Cholesky factors of  $\Delta_i$ .  $2N$  words of storage are needed for  $M$ , as in BDIA. We do not consider here keeping more diagonals in the approximation to  $\Delta_{i-1}^{-1}$  for this case, as the particularly simple expression in Theorem 2 becomes more complex if the  $\Delta_i$ 's have more than 3 diagonals.

**4.1.3. CHOL( $p$ ).** We use  $\tilde{T}_3(p)$  from (11),

$$\Lambda_{i-1} = \mathbf{B}(U_{i-1}^{-1}, p)\mathbf{B}(U_{i-1}^{-T}, p),$$

where  $\Delta_{i-1} = U_{i-1}^T U_{i-1}$ , with  $U_{i-1}$  an upper triangular matrix. At each stage we compute  $U_{i-1}$ , which is (except for  $i = 2$ ) a matrix with  $p + 1$  nonzero main diagonals. The first  $p + 1$  diagonals of  $U_{i-1}^{-1}$  can be computed diagonalwise starting from the main diagonal. Since  $\Lambda_{i-1}$  is a symmetric matrix of bandwidth  $2p + 1$ , approximately  $2m_1 + (p + 1) \sum_{i=2}^n m_i$  words of storage are needed for  $\Delta_i$ .



CHOL( $p$ ) is a special case of the following method proposed by Underwood [20] in a slightly different setting.

**4.1.4. UND( $p, q$ ).** For this case

$$\Lambda_{i-1} = \mathbf{B}(\mathbf{B}(U_{i-1}^{-1}, q-1)\mathbf{B}(U_{i-1}^{-T}, q-1), 2p-1),$$

with  $q \geq p$ . One computes the  $q$  main diagonals of  $U_{i-1}^{-T}$ , but then stores only the  $2p-1$  main diagonals of the product to form  $\Lambda_{i-1}$ . More information about  $U_{i-1}^{-T}$  is used in UND( $p, q$ ) than in CHOL( $p$ ). The storage needed is  $2m_1 + p \sum_{i=2}^n m_i$ . Note that  $\text{UND}(p, p) \equiv \text{CHOL}(p-1)$ .

**4.1.5. POL( $\alpha, \beta$ ).** We use the polynomial approximation  $\tilde{T}_4(\alpha, \beta)$  defined in §3.4,

$$\Delta_{i-1} = D_{T,i-1} + \bar{T}_{i-1},$$

$$\Lambda_{i-1} = \alpha D_{T,i-1}^{-1} + \beta D_{T,i-1}^{-1} \bar{T}_{i-1} D_{T,i-1}^{-1}.$$

Each  $\Delta_i$  is tridiagonal. Different values of  $\alpha$  and  $\beta$  are used. The storage requirements are the same as for BDIA and INV(1).

**4.2. Properties of the  $\Delta_i$ .** Now we study the properties of the  $\Delta_i$  in order to prove that all of the methods described above can be carried out (i.e., we prove that the  $\Delta_i$  satisfy hypothesis (H2) placed on  $T$ ).

**THEOREM 3.** *Under Hypothesis (H1) each  $\Delta_i$  computed by BDIA, INV(1), CHOL( $p$ ), UND( $p, q$ ), and POL( $\alpha, \beta$ ) with  $\beta \leq 0$ ,  $0 < \alpha \leq 1$ ,  $\beta + \alpha \geq 0$  is strictly diagonally dominant with positive diagonal elements and nonpositive off diagonal elements.*

*Proof.* This can be proved by induction using the same technique as in Lemma 1. As the proof is essentially the same for all cases, we carry it out only for CHOL( $p$ ).

Let

$$B = \begin{pmatrix} B_1 & -C^T \\ -C & B_2 \end{pmatrix}$$

be a positive definite  $M$ -matrix satisfying (H1) with  $B_1$  and  $B_2$  square, and let  $B_1 = L_B L_B^T$  be the Cholesky decomposition of  $B_1$ . Denote

$$L_{B_1}^{-1} = \tilde{L}_{B_1}^{-1} + R_{B_1},$$

where  $\tilde{L}_{B_1}^{-1}$  contains the  $p+1$  diagonals of  $L_{B_1}^{-1}$  that are kept for the approximation, and  $R_{B_1}$  contains the remaining diagonals. Under hypothesis (H1) both  $\tilde{L}_{B_1}^{-1}$  and  $R_{B_1}$  are nonnegative. From the remark following Lemma 1 we know that  $B_2 - CL_{B_1}^{-T} L_{B_1}^{-1} C^T$  is strictly diagonally dominant. We have

$$B_2 - C\tilde{L}_{B_1}^{-T} \tilde{L}_{B_1}^{-1} C^T = B_2 - CL_{B_1}^{-T} L_{B_1}^{-1} C^T + C(\tilde{L}_{B_1}^{-T} R_{B_1} + R_{B_1}^T \tilde{L}_{B_1}^{-1} + R_{B_1}^T R_{B_1}) C^T.$$

The last matrix on the right is nonnegative, which implies that  $B_2 - C\tilde{L}_{B_1}^{-T} \tilde{L}_{B_1}^{-1} C^T$  is at least as strongly diagonally dominant as is  $B_2 - CL_{B_1}^{-T} L_{B_1}^{-1} C^T$ . The desired result for CHOL( $p$ ) then follows by induction, taking  $B_1 = D_1$ , the first diagonal block of  $A$ .

**4.3. Modified block preconditionings.** It is known that point incomplete Cholesky decomposition can be modified to yield a better approximation to  $A$  in some cases. The modified decomposition is obtained from  $R$  by altering the diagonal elements of the Cholesky factors so that the row sums of  $M$  are equal to the corresponding row sums of  $A$  (i.e., the row sums of  $R$  are zero). This gives an improvement of the condition number of  $M^{-1}A$  for natural ordering of the unknowns and for  $A$  diagonally dominant [7], [13].

As noted in §3, the remainder  $R$  for the block incomplete Cholesky preconditioning is a block diagonal matrix whose elements are

$$R_1 = 0,$$

$$R_i = \Delta_i - D_i + A_i \Delta_i^{-1} A_i^T = A_i (\Delta_i^{-1} - \Lambda_{i-1}) A_i^T, \quad 2 \leq i \leq n.$$

Thus the row sum of  $\Delta_i^{-1}$  must be available if  $R_i$  is to have row sum zero.

**4.3.1. MINV(1).** For the case of INV(1),  $\Delta_i^{-1}$  itself is readily available, thus it is easy to define MINV(1), the modified form of INV(1): Compute  $\Delta_i^{-1}$  at each stage from the two vectors  $u$  and  $v$ . Form the product  $R_i = A_i [\Delta_i^{-1} - \mathbf{B}(\Delta_i^{-1}, 1)] A_i^T$ , which is a matrix with positive elements except for the 3 main diagonals, which are zero. Then subtract from  $D_i - A_i \mathbf{B}(\Delta_i^{-1}, 1) A_i^T$  the diagonal matrix made up of the row sums of  $R_i$ , to yield the modified  $\Delta_i$  corresponding to a remainder with a zero row sum.

We note that it follows from Hypothesis (H1) that the remainder matrix is nonpositive definite, hence the eigenvalues of  $M^{-1}A$  are greater than or equal to 1 for MINV(1).

**THEOREM 4.** *Under Hypothesis (H1) each  $\Delta_i$  given by MINV(1) is a strictly diagonally dominant matrix with positive diagonal elements and negative off diagonal elements.*

*Proof.* Consider

$$\begin{pmatrix} B_1 & -C^T \\ -C & B_2 \end{pmatrix}.$$

Let  $S_2 = C[B_1^{-1} - \mathbf{B}(B_1^{-1}, 1)]C^T$  and let  $R_2$  be the diagonal matrix of row sums of  $S_2$ . Since  $B_1^{-1} \geq 0$ , the elements of  $S_2$  and hence of  $R_2$  are positive. Note that  $B_2 - C\mathbf{B}(B_1^{-1}, 1)C^T - R_2$  has the same row sums as  $B_2 - C[\mathbf{B}(B_1^{-1}, 1)]C^T - S_2 = B_2 - CB_1^{-1}C^T$ . This, together with the positivity of the elements, shows that  $B_2 - C\mathbf{B}(B_1^{-1}, 1)C^T - R_2$  is diagonally dominant.

**4.3.2. MUND( $p, q$ ).** For the other block preconditionings the row sums of  $R_i$  can be calculated easily, but not quite so directly. However, in UND( $p, q$ ) with  $q > p$  a part of the remainder is immediately available and can be subtracted from the diagonal. Recall that

$$\Delta_{i-1} = L_{i-1} L_{i-1}^T,$$

$$R_i = A_i [\Delta_i^{-1} - \mathbf{B}(L_{i-1}^{-T}, q-1) \mathbf{B}(L_{i-1}^{-1}, q-1), 2p-1].$$

Denote by  $\tilde{L}_{i-1}^{-1} = \mathbf{B}(L_{i-1}^{-1}, q-1)$  the  $q$  diagonals of the inverse of  $L_{i-1}$  that are computed, and by  $Q_{i-1}$  the diagonals that are not computed

$$L_{i-1}^{-1} = \tilde{L}_{i-1}^{-1} + Q_{i-1}.$$

Then

$$R_i = A_i[\tilde{L}_{i-1}^{-T}\tilde{L}_{i-1}^{-1} + \tilde{L}_{i-1}^{-T}Q_{i-1} + Q_{i-1}^T\tilde{L}_{i-1}^{-1} + Q_{i-1}^TQ_{i-1} - \mathbf{B}(\tilde{L}_{i-1}^{-T}\tilde{L}_{i-1}^{-1}, 2p-1)]A_i^T.$$

We can obtain  $\tilde{L}_{i-1}^{-T}\tilde{L}_{i-1}^{-1} - \mathbf{B}(\tilde{L}_{i-1}^{-T}\tilde{L}_{i-1}^{-1}, 2p-1)$ , since it is made up of the diagonals of the product that are not kept in the algorithm. Thus, instead of discarding these diagonals we could subtract their row sums from the main diagonal. This constitutes the algorithm MUND( $p, q$ ): Compute  $q$  diagonals of  $L_{i-1}^{-1}$ . Form the product  $\tilde{L}_{i-1}^{-T}\tilde{L}_{i-1}^{-1}$ . Use the  $2p-1$  main diagonals to form  $D_i - A_i\mathbf{B}(\tilde{L}_{i-1}^{-T}\tilde{L}_{i-1}^{-1}, 2p-1)A_i^T$ . Let  $S_{i-1}$  be the matrix made up of the  $q-p$  outer diagonals of  $\tilde{L}_{i-1}^{-T}\tilde{L}_{i-1}^{-1}$ . Compute the row sums of  $A_iS_{i-1}A_i^T$  and subtract them from the diagonal of  $D_i - A_i\mathbf{B}(\tilde{L}_{i-1}^{-T}\tilde{L}_{i-1}^{-1}, 2p-1)A_i^T$  to obtain  $\Delta_i$ .

**THEOREM 5.** *Under Hypothesis (H1) each  $\Delta_i$  given by MUND( $p, q$ ) is a strictly diagonally dominant matrix with positive diagonal elements and negative off-diagonal elements.*

*Proof.* Along the same lines as for Theorem 4.

**4.4. Higher dimensions.** One can develop block incomplete Cholesky factorizations for three dimensional problems similarly, using, for example, incomplete instead of complete factorizations  $L_i$  for the  $\Delta_i$ . It is planned to investigate these preconditionings in a subsequent study.

**5. Numerical experiments.** In this section we present the results of numerical experiments on two-dimensional test problems comparing the preconditionings introduced in the previous sections and some other, commonly used, point and block preconditionings. The other preconditionings include: the point incomplete Cholesky decomposition IC( $p, q$ ) introduced by Meijerink and van der Vorst [16], [17], in which  $p$  bands adjacent to the main diagonal and  $q$  outer bands are kept in the factorization; its modified version MIC( $p, q$ ), of which the simplest MIC(1,1), first introduced by Dupont, Kendall, and Rachford for five diagonal matrices [7], is denoted here by DKR (and is used without parameters); symmetric successive overrelaxation (SSOR) and its block version BSSOR (which in our case is line SSOR); and for a few cases 1-line Jacobi preconditioning (LJAC). In addition, results will be given for some problems for the point Jacobi preconditioning DIAG, for which  $M$  is a diagonal matrix whose diagonal elements are those of  $A$ , and for conjugate gradients without preconditioning ( $M = I$ , the identity matrix).

For a five diagonal matrix the work per iteration and storage for each of the methods is given in Table 2. (For simplicity, the technique of [8] for reducing the work requirements of the conjugate gradient method is not incorporated.) The work is represented by number of floating point multiplies; about the same number of additions are required also.

Table 2 does not include the overhead operations required to construct  $M$ . If one carries out many iterations or solves several systems with different right-hand sides, then this overhead can usually be neglected. Specific cases are discussed in §5.1. Also not included in Table 2 is the work that might be required for evaluating iteration termination criteria.

It should be noted that the work requirements for the preconditionings depend on the manner in which the computer programs are written. Generally we have organized our programs with a preference toward multiplication over division; for example, in INV(1) we use Varga's implementation of Gauss elimination

TABLE 2

*Work per iteration and storage for the preconditionings.*

Preconditioning $M$	Mults.	Storage
I	10N	0
DIAG	11N	0
IC(1,1), DKR	16N	N
SSOR	17N	0
IC(1,2), MIC(1,2)	18N	3N
IC(1,3), MIC(1,3)	20N	4N
IC(2,4)	24N	6N
BSSOR	18N	2N
BDIA, INV(1), MINV(1), POL( $\alpha, \beta$ )	18N	2N
CHOL( $p$ ), UND( $p+1, q$ ), MUND( $p+1, q$ )	$(4p+14)N$	$(p+1)N$

for tridiagonal matrices, which stores the reciprocals of the diagonals [21]. If a division is carried out, as in DIAG when it is desired neither to scale the matrix in advance nor to store the reciprocals of the diagonal, then, as is customary, a division is counted as equivalent to a multiply. In CHOL( $p: p > 1$ ), UND( $p, q$ ), and MUND( $p, q$ ) routines from LINPACK [6] are used, but the operation counts entered in Table 2 are made to correspond to the manner in which we implement the other preconditionings. Thus the entries in Table 2, though basically consistent, should be considered as approximate. They are used in subsequent tables to convert observed number of iterations to computational work.

Our implementation of the conjugate gradient algorithm requires 4  $N$ -vectors of storage, plus 3  $N$ -vectors for the matrix  $A$  and 1  $N$ -vector for the right-hand side. If it is not necessary to save the right-hand side, then 1  $N$ -vector of storage could be eliminated. The additional storage required by each of the preconditionings is given in the last column of Table 2.

**5.1. First test problem.** The first test problem is the model problem

$$-\Delta u = f \quad \text{in } \Omega \text{ the unit square } (0,1) \times (0,1)$$

with

$$u \Big|_{\partial\Omega} = 0.$$

We use the standard five point stencil on a square mesh with  $h = (n+1)^{-1}$ ,  $N = n^2$ , and natural ordering to obtain the corresponding linear algebraic system (2). The experimental results are given for different values of  $h$  and different stopping criteria. An estimate of the condition number of  $M^{-1}A$  is given for each of the preconditionings, as obtained from the conjugate gradient algorithm (cf. [3]), and for small dimension ( $n = 10$ ) the complete spectrum of  $M^{-1}A$  is visualized.

The computations were carried out in double precision FORTRAN on an IBM 3081. Unless otherwise noted the solution of the linear system is smooth

(the right-hand side  $b$  in (2) corresponds to the solution  $\xi_i(\xi_i - 1)\eta_j(\eta_j - 1)\exp(\xi_i\eta_j)$  at a point  $(\xi_i, \eta_j)$ ), and the starting vector has random elements in  $[-1, 1]$ . As the number of additions is roughly the same as the number of multiplications, we indicate only the work required for the multiplications. The divisions that may appear to be needed by some methods are not indicated, since they can be removed with alternative coding. In Table 3 are given the number of iterations

TABLE 3  
*Number of iterations and total work  
 per point for  $\|r^k\|_\infty / \|r^0\|_\infty \leq 10^{-6}$ .  
 Test problem 1,  $N = 2500$ .*

$M$	# iterations	work/ $N$
I	109	1090
DIAG	109	1199
IC(1,1)	33	528
IC(1,2)	21	378
IC(1,3)	17	340
IC(2,4)	12	288
DKR	23	368
MIC(1,2)	17	306
MIC(1,3)	14	280
SSOR $\omega = 1$	40	680
SSOR $\omega = 1.7$	21	357
LJAC	80	1040
BSSOR $\omega = 1$	28	504
BSSOR $\omega = 1.7$	16	288
BDIA	22	396
POL(1,-1)	18	324
POL(0.9412,-0.4706)	21	378
POL(1.143,-1.143)	17	306
INV(1)	15	270
MINV(1)	11	198
CHOL(1)	16	288
CHOL(2)	12	264
CHOL(3)	9	234
CHOL(4)	8	240
CHOL(5)	7	238
UND(2,3)	15	270
UND(2,4)	15	270
UND(3,4)	11	242
UND(3,5)	11	242
UND(4,5)	9	234
UND(4,6)	9	234
UND(5,6)	7	210
MUND(2,3)	12	216
MUND(2,4)	10	180
MUND(2,5)	9	162
MUND(3,4)	10	220
MUND(3,5)	8	176
MUND(3,6)	8	176
MUND(4,5)	8	208
MUND(4,6)	7	182
MUND(5,6)	7	210

and the corresponding total work per point required to achieve the stopping criterion  $\|r^k\|_\infty/\|r^0\|_\infty \leq 10^{-6}$ , for the case  $N = 2500$ . The value  $\omega = 1.7$  for SSOR and BSSOR is the observed optimal for each case to the nearest 0.1 for minimizing the number of iterations required for convergence.

From Table 3, the following observations can be made.

- (i) For the patterns chosen, the larger the number of diagonals in the incomplete Cholesky decomposition, the fewer the number of iterations required for convergence, as observed in [17] for the point preconditionings.
- (ii) The modified versions of the preconditionings give better results (for this problem and ordering of the mesh points).
- (iii) In general, there is a trade off between storage and execution speed, but if a low storage point-preconditioning is desired, DKR seems a good choice. SSOR can give good results, but suitable parameter values are needed.
- (iv) For methods of comparable storage the block methods give better results than point methods, both in terms of number of iterations and work requirements.
- (v) For CHOL( $p$ ) it is not effective to go to values of  $p$  larger than  $p = 3$ , and, as observed also in [2], for UND( $p, q$ ) to values of  $q$  beyond  $q = p + 1$ . It is better to use the additional information given by UND( $p, q$ ) for larger  $q$  to obtain a modified version of the factorization for  $q = p + 1$ .
- (vi) The best polynomial, as expected, is POL(1.1429, -1.1429).
- (vii) For this problem the best all-around preconditioning appears to be MINV(1), because it has very low storage requirements and gives almost the best work count -- approximately half of IC(1,2) and two thirds of MIC(1,2), which require more storage.

Table 4 gives a comparison of some of the methods for solving the test problem to only moderate accuracy  $\|r^k\|_\infty/\|r^0\|_\infty \leq 10^{-4}$ , comparable to discretization error. The conclusions drawn for the smaller residuals in Table 3 are in general unchanged.

TABLE 4

*Number of iterations and total work per point for  $\|r^k\|_\infty/\|r^0\|_\infty \leq 10^{-4}$ .  
Test problem 1,  $N = 2500$ .*

$M$	# iterations	work/ $N$
I	63	630
IC(1,1)	20	320
IC(2,4)	7	168
DKR	16	256
SSOR $\omega = 1.7$	13	221
BSSOR $\omega = 1.7$	10	180
INV(1)	9	162
MINV(1)	7	126
CHOL(1)	9	162
CHOL(5)	4	136

In Table 5 are given the values of the smallest and largest eigenvalues of  $M^{-1}A$ , as estimated by the conjugate gradient algorithm, as well as the corresponding condition numbers. It is seen that a considerable reduction in the condition number can be achieved with some of the modified preconditionings, with only a low cost in storage.

TABLE 5  
Extremal eigenvalues and condition number of  $M^{-1}A$ .  
Test problem 1,  $N = 2500$ .

$M$	$\lambda_{\min}(M^{-1}A)$	$\lambda_{\max}(M^{-1}A)$	$\kappa(M^{-1}A)$
I	0.0076	7.992	1053
IC(1,1)	0.0128	1.206	94.0
IC(1,2)	0.033	1.179	35.6
IC(1,3)	0.049	1.131	23.2
IC(2,4)	0.091	1.138	12.5
DKR	1.003	15.36	15.3
MIC(1,2)	1.003	8.83	8.3
MIC(1,3)	1.006	6.19	6.15
SSOR $\omega = 1$ .	0.0075	1.	132.5
SSOR $\omega = 1.7$	0.040	1.	25.1
LJAC	0.0038	1.99	527.
BSSOR $\omega = 1$ .	0.0150	1.	66.8
BSSOR $\omega = 1.7$	0.074	1.	13.5
BDIA	0.024	1.023	42.6
POL(1,-1)	0.035	1.	28.7
POL(0.9412,-0.4706)	0.027	1.002	37.2
POL(1.143,-1.143)	0.043	1.023	23.8
INV(1)	0.059	1.073	18.2
MINV(1)	1.006	4.261	4.24
CHOL(1)	0.050	1.050	20.8
CHOL(2)	0.090	1.065	11.8
CHOL(3)	0.142	1.076	7.56
CHOL(4)	0.204	1.078	5.29
CHOL(5)	0.272	1.078	3.97
UND(2,3)	0.058	1.07	18.5
UND(2,4)	0.059	1.073	18.2
UND(2,5)	0.059	1.073	18.2
UND(3,4)	0.104	1.086	10.5
UND(3,5)	0.106	1.089	10.2
UND(4,5)	0.162	1.091	6.75
UND(4,6)	0.166	1.096	6.59
UND(5,6)	0.228	1.088	4.78
MUND(2,3)	0.102	1.242	12.2
MUND(2,4)	0.202	1.564	7.74
MUND(2,5)	0.380	2.024	5.33
MUND(3,4)	0.164	1.242	7.58
MUND(3,5)	0.291	1.518	5.22
MUND(3,6)	0.483	1.887	3.91
MUND(4,5)	0.234	1.221	5.21
MUND(4,6)	0.375	1.449	3.87
MUND(5,6)	0.309	1.197	3.88

In Table 6 are given the estimated condition numbers  $\kappa(M^{-1}A)$  for different values of  $n = (1/h) - 1$ . The quantity  $\alpha$  is the estimated value, from the  $n = 25$  and  $n = 50$  data, of the exponent corresponding to the assumed asymptotic relationship  $\kappa(M^{-1}A) \sim Ch^{-\alpha}$ , where  $C$  is a constant. It is known theoretically that for  $M = I$  and  $M = IC(1,1)$  there holds  $\kappa(M^{-1}A) = O(h^{-2})$  and that for  $M = DKR$ ,  $\kappa(M^{-1}A) = O(h^{-1})$ . The values of  $\alpha$  obtained from the numerical

TABLE 6  
*Estimated condition number for different mesh sizes  
 and exponent  $\alpha$  of asymptotic dependence on  $h = 1/(n + 1)$ .  
 Test problem 1.*

$M$	$\kappa(M^{-1}A)$				$\alpha$
	$n = 10$	$n = 20$	$n = 25$	$n = 50$	
I	48.37	178.1	273.3	1053	2.00
IC(1,1)	5.10	16.59	25.	94	1.97
IC(1,2)	2.38	6.67	9.8	35.6	1.91
IC(1,3)	1.80	4.56	6.6	23.2	1.87
IC(2,4)	1.32	2.75	3.8	12.5	1.77
DKR	3.04	5.93	7.4	15.3	1.08
MIC(1,2)	1.84	3.36	4.2	8.3	1.01
MIC(1,3)	1.49	2.56	3.15	6.1	0.98
SSOR $\omega = 1$ .	6.88	23.12	35.	132	1.97
LJAC	24.68	89.5	137.	527	2.00
BSSOR $\omega = 1$ .	3.93	12.04	18.	66.7	1.94
BDIA	2.76	7.9	11.7	42.5	1.91
POL(1,-1)	2.09	5.52	8.	28.6	1.89
POL(0.9412,-0.4706)	2.5	7.	10.3	37.1	1.90
POL(1.143,-1.143)	1.86	4.7	6.7	23.8	1.88
INV(1)	1.61	3.74	5.3	18.2	1.83
MINV(1)	1.3	1.94	2.31	4.23	0.90
CHOL(1)	1.73	4.18	6.	20.8	1.85
CHOL(2)	1.32	2.65	3.65	11.85	1.75
CHOL(3)	1.14	1.93	2.53	7.54	1.62
CHOL(4)	1.06	1.55	1.95	5.28	1.48
CHOL(5)	1.026	1.34	1.61	3.98	1.34
UND(2,3)	1.63	3.8	5.4	18.52	1.83
UND(2,4)	1.62	3.75	5.33	18.24	1.83
UND(3,4)	1.26	2.42	3.3	10.47	1.71
UND(3,5)	1.25	2.39	3.24	10.24	1.71
UND(4,5)	1.12	1.8	2.33	6.73	1.57
UND(4,6)	1.11	1.77	2.28	6.54	1.56
UND(5,6)	1.05	1.47	1.82	4.8	1.44
MUND(2,3)	1.39	2.76	3.79	12.95	1.82
MUND(2,4)	1.29	2.1	2.72	7.74	1.55
MUND(2,5)	1.28	1.89	2.26	5.33	1.27
MUND(3,4)	1.18	1.97	2.58	7.55	1.59
MUND(3,5)	1.15	1.67	2.04	5.22	1.39
MUND(3,6)	1.14	1.6	1.85	3.9	1.11
MUND(4,5)	1.09	1.57	1.96	5.22	1.45
MUND(4,6)	1.07	1.43	1.68	3.8	1.21
MUND(5,6)	1.04	1.35	1.62	3.9	1.30



experiments are in accord with these relationships. We see that all the point incomplete decompositions  $IC(p,q)$  seem to be  $O(h^{-2})$ , although the more diagonals that are taken the slower is the convergence to this asymptotic behavior. The MIC methods are  $O(h^{-1})$ .

For the block methods INV and CHOL the limiting value of  $\alpha$  seems to be two, and for MINV one. The observed values of  $\alpha$  for the range of  $h$  considered are smaller for the block methods than for the point methods with the same storage. It is difficult to assess from the results the order of the MUND methods; we believe that they are somewhere between 1 and 2, closer to 1 if more diagonals are used to form  $M$ . Finally, Table 6 shows that even for smaller values of  $n$  block methods give better reduction of the condition number than point methods.

It is well known that the rate of convergence of the conjugate gradient method depends not only on the condition number but on the distribution of the interior eigenvalues as well. It is therefore of interest to compare the eigenvalue spectra for the different methods. These are compared for  $n = 10$  in Figs. 2-4. Each eigenvalue is designated by a vertical bar drawn at the appropriate abscissa value. This representation depicts in an easily observable manner the separation and clustering of the eigenvalues.

The spectra for all of the methods shown in Fig. 2 are on the same scale for easy comparison. From the figure it is seen that for the block methods the eigenvalues are more clustered than for the point ones having the same storage requirements. (The relatively greater clustering for block SSOR over point SSOR is a well-known property, cf. [9].) The values  $\omega = 1.7$  and  $\omega = 1.5$  are to the nearest 0.1 those for which the condition numbers for SSOR and BSSOR, respectively, are smallest. The point modified methods, for which the eigenvalue range is different than for the other methods, are shown separately in Fig. 3. Fig. 4 shows on the same scale four methods with comparable storage:  $IC(1,1)$  and DKR, with one vector of storage, and INV(1) and MINV(1) with two. Spectra for block SSOR preconditioning for the values  $\omega = 1.0(0.1)1.9$  can be found in [2], and enlargements showing the fine structure of the spectra of Figs. 2-4 are in an Appendix to [2], available separately from the authors.

Table 7 gives the number of iterations required to solve the test problem for different convergence criteria. For these cases the initial approximation was  $x^0 \equiv 0$ , and the solution was the same smooth vector as for Tables 4 and 5 with  $N = 2500$ .

From these results, it appears that, at least for the test problem with a smooth solution, the relative norm of the residual gives a good stopping criterion.

In Table 8 we give results for  $N = 2500$  for the same smooth solution as for previous tables, with two different choices of the starting vector,  $x^0 \equiv 0$  and  $x^0$  consisting of random numbers in  $[-1,1]$ . The stopping criterion is  $\|r^k\|_\infty / \|r^0\|_\infty \leq 10^{-6}$ . The initial approximation  $x^0$  random appears to give better results. This feature will be developed in a subsequent study.

From the tables one can conclude that for this test problem block methods give better results than point ones. The most promising block method is MINV(1). Since the setup time for constructing  $M$  was not included in the tables, it is of interest to consider it, as it can be of importance if only one problem is to be solved or only a few iterations taken. Table 9 gives the effect of including the setup time for three of the preconditionings for the  $N = 2500$  test problem. Times are in CPU seconds for an IBM 3081 computer. Even if the setup times are

included, MINV(1) still gives considerable improvement for this problem.

The effects of Neumann boundary conditions were examined as well in [2], where it was found that the relative merits of the different preconditionings are about the same as for this test problem.

TABLE 7  
Number of iterations for different convergence criteria.  
Test problem 1,  $x^0 \equiv 0$ .

$M$	Number of iterations			
	$\frac{\ r^k\ _\infty}{\ r^0\ _\infty} \leq 10^{-6}$	$\ x - x^k\ _\infty \leq 10^{-6}$	$\ x - x^k\ _2 \leq 10^{-6}$	$\ x - x^k\ _A \leq 10^{-6}$
I	117	99	114	110
IC(1,1)	38	31	36	35
IC(1,2)	26	22	26	24
IC(1,3)	21	19	22	20
IC(2,4)	16	14	16	15
DKR	25	18	22	21
MIC(1,2)	18	14	17	16
MIC(1,3)	18	16	18	17
SSOR $\omega = 1$ .	44	37	43	41
SSOR $\omega = 1.7$	22	17	20	19
BSSOR $\omega = 1$ .	36	28	34	32
BSSOR $\omega = 1.7$	18	15	18	16
BDIA	27	24	28	26
POL(1,-1)	23	20	24	22
INV(1)	19	16	19	18
MINV(1)	13	9	11	11
CHOL(1)	20	18	21	19
CHOL(2)	15	13	16	14
CHOL(3)	12	11	13	12
CHOL(4)	10	9	10	10
CHOL(5)	9	8	9	8
UND(2,3)	19	16	19	18
UND(3,4)	14	13	15	14
UND(4,5)	12	10	12	11
UND(5,6)	9	8	10	9
MUND(2,3)	15	14	16	15
MUND(2,4)	13	11	13	12
MUND(2,5)	12	9	11	10
MUND(3,4)	12	11	13	12
MUND(3,5)	11	9	11	10
MUND(4,5)	10	9	10	10
MUND(4,6)	9	8	9	9
MUND(5,6)	9	8	9	8

TABLE 8

*Number of iterations  
for  $\|r^k\|_\infty / \|r^0\|_\infty \leq 10^{-6}$   
for different starting vectors.  
Test problem 1.*

$M$	# of iterations	
	$x^0 = 0$	$x^0$ random
I	117	109
IC(1,1)	38	33
IC(1,2)	26	21
IC(1,3)	21	17
IC(2,4)	16	12
DKR	25	23
MIC(1,2)	18	17
MIC(1,3)	18	14
SSOR $\omega = 1.$	44	40
SSOR $\omega = 1.7$	22	21
BSSOR $\omega = 1.$	36	28
BSSOR $\omega = 1.7$	18	16
BDIA	27	22
POL(1,-1)	23	18
INV(1)	19	15
MINV	13	11
CHOL(1)	20	16
CHOL(2)	15	12
CHOL(3)	12	9
CHOL(4)	10	8
CHOL(5)	9	7
UND(2,3)	19	15
UND(3,4)	14	11
UND(4,5)	12	9
UND(5,6)	9	7
MUND(2,3)	15	12
MUND(2,4)	13	10
MUND(2,5)	12	9
MUND(3,4)	12	10
MUND(3,5)	11	8
MUND(4,5)	10	8
MUND(4,6)	9	7
MUND(5,6)	9	7

TABLE 9

*Total time including  
setup in CPU seconds for  
 $\|r^k\|_\infty / \|r^0\|_\infty \leq 10^{-6}$ .  
Test problem 1.*

$M$	total time
IC(1,1)	1.37
INV(1)	0.963
MINV(1)	0.723

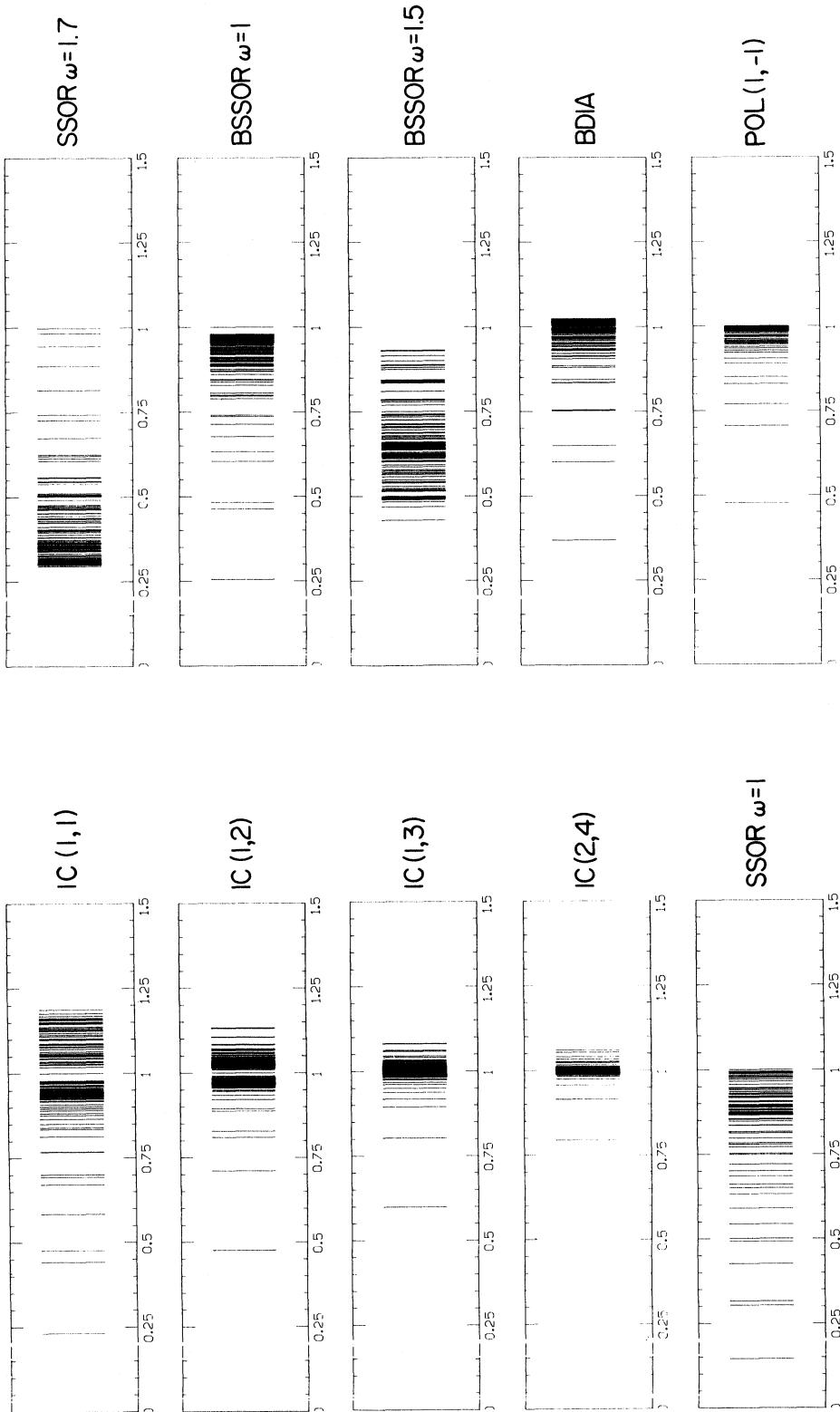


FIG. 2. Spectra of  $M^{-1}A$  for different preconditionings  $M$ .  
Test problem 1.  $N = 100$ .

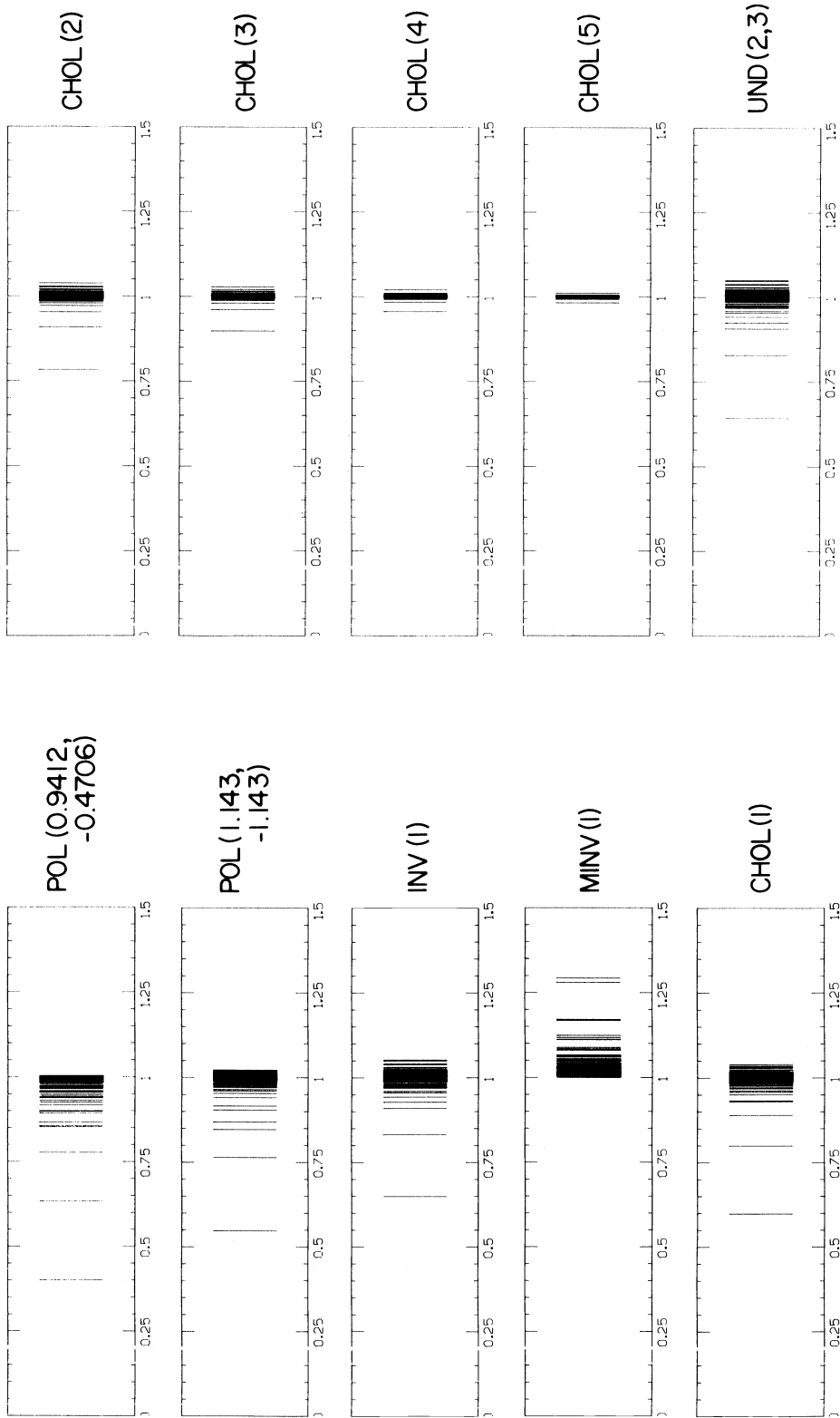


FIG. 2. (cont.)

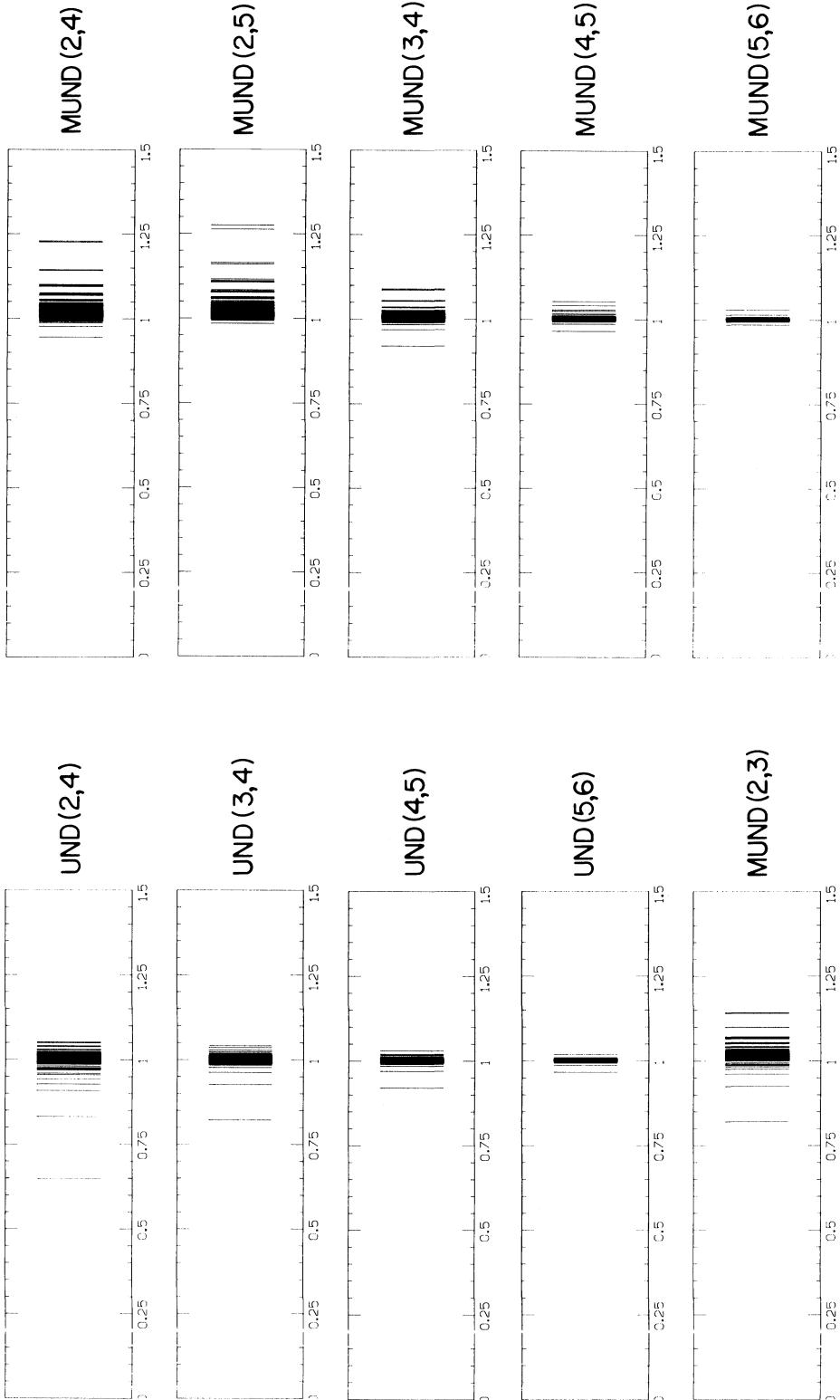


Fig. 2. (cont.)

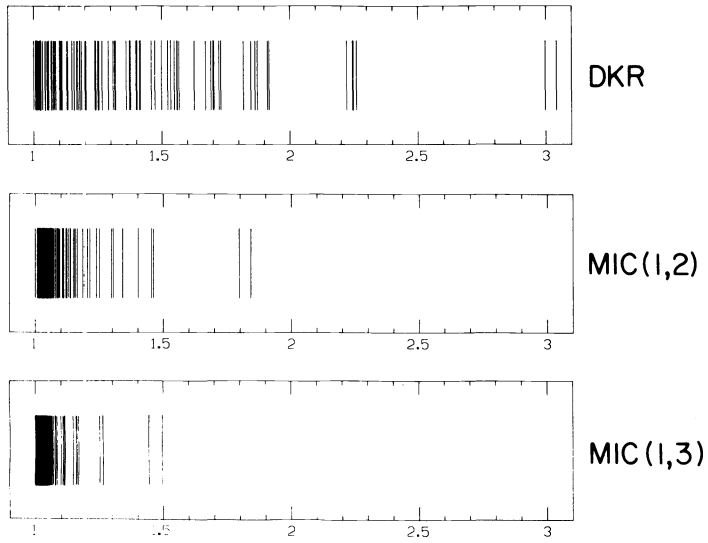


FIG. 3. Spectra of  $M^{-1}A$  for modified preconditionings. Test problem 1.  $N = 100$ .

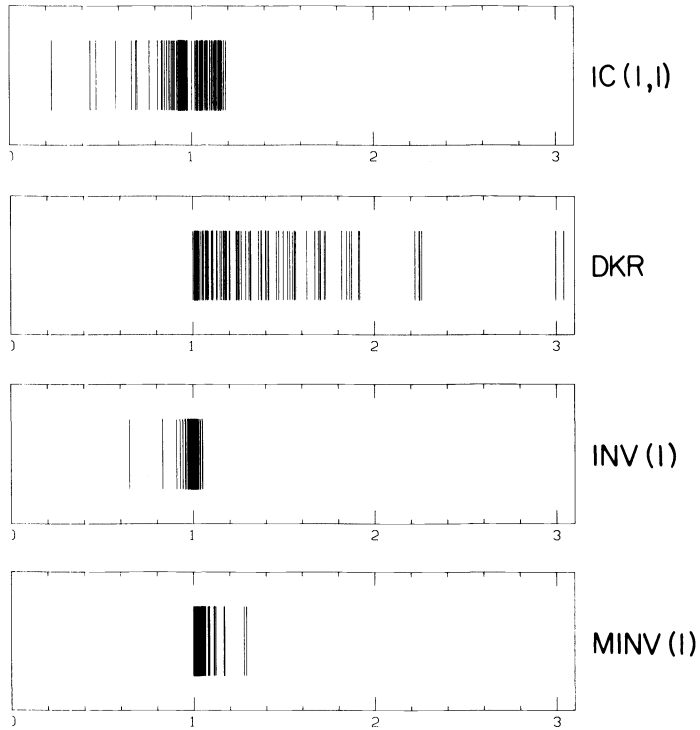


FIG. 4. Spectra of  $M^{-1}A$  for four preconditionings with comparable, minimal storage. Test problem 1.  $N = 100$ .

**5.2. Second test problem.** We solve the linear system obtained by the standard five point discretization of the problem

$$-\frac{\partial}{\partial \xi_1} \left[ \lambda(\xi_1, \xi_2) \frac{\partial u}{\partial \xi_1} \right] - \frac{\partial}{\partial \xi_2} \left[ \lambda(\xi_1, \xi_2) \frac{\partial u}{\partial \xi_2} \right] = f \quad \text{in } \Omega = (0,1) \times (0,1),$$

$$u = 0 \quad \text{on } \partial\Omega,$$

for the discontinuous  $\lambda$  depicted in Fig. 5. The solution is the same smooth one as for the first test problem, the starting vector is random, and the stopping criterion is  $\|r^k\|_\infty / \|r^0\|_\infty \leq 10^{-6}$ .

Table 10 gives the results for the number of iterations, the work required, and an estimate of the condition number as obtained from the conjugate gradient parameters. The values  $\omega = 1.6$  for SSOR and  $\omega = 1.5$  for BSSOR are the observed optimal ones to the nearest 0.1.

The very large condition numbers for most of the entries result from the small first eigenvalue, which is isolated from the others. Thus the number of iterations does not change much, for example from IC(1,1), which has a small isolated eigenvalue, to DKR, which has all eigenvalues greater than one. It is the distribution of the other eigenvalues that is important. In terms of work per point, block methods give better results than point ones. Again MINV(1) seems a good compromise between efficiency and storage. This example shows that block methods can be effective for problems with coefficients having large jump discontinuities.

**5.3. Third test problem.** This example, which is frequently used in the literature, was presented in [21]. The problem is to solve

$$-\frac{\partial}{\partial \xi_1} \left[ \lambda_1 \frac{\partial u}{\partial \xi_1} \right] - \frac{\partial}{\partial \xi_2} \left[ \lambda_2 \frac{\partial u}{\partial \xi_2} \right] + \sigma u = 0 \quad \text{in } \Omega = (0,2.1) \times (0,2.1),$$

$$\frac{\partial u}{\partial n} \Big|_{\partial\Omega} = 0.$$

The domain is shown in Fig. 6 and depicts the values of the coefficients, which are discontinuous. The solution is  $u \equiv 0$ .

We take  $h = 1/42$ ,  $x^0$  a vector with random elements in  $[-1,1]$ , and stopping criterion  $\|x^k\|_\infty \leq 10^{-6}$ . The results are given in Table 11. The values  $\omega = 1.7$  for SSOR and  $\omega = 1.5$  for BSSOR are the observed optimal ones to the nearest 0.1.

Table 11 indicates that for this problem the larger the number of diagonals retained, the lower the work required for convergence. This holds both for point and block methods. Generally, the block methods are slightly better.

In order to compare our methods with those presented by Meijerink and Van der Vorst [17] for this problem, we give the results in Table 12 for convergence criterion  $\|r^k\|_2 \leq 10^{-6}$ . For the IC methods, we obtain about the same results as in [17], within a few iterations. (The distribution from which the starting vectors were drawn is different—our random numbers are between -1 and 1, while theirs are between 0 and 1.)

To compare point and block methods with the same storage, one can take, for example, IC(1,2) or MIC(1,2) and CHOL(2). It is clear that the block method is better. The situation is the same if more diagonals are taken. To get down to 16



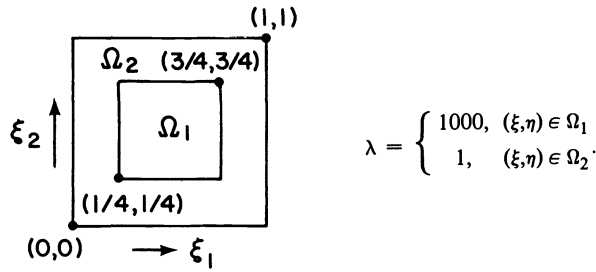


FIG. 5. Test problem 2.

TABLE 10

Number of iterations, total work per point,  
and estimated condition number of  $M^{-1}A$ .  
Test problem 2,  $N = 2500$ ,  $\|r^k\|_\infty / \|r^0\|_\infty \leq 10^{-6}$ .

$M$	# iterations	work/ $N$	$\kappa(M^{-1}A)$
DIAG	137	1507	
IC(1,1)	47	752	46770
IC(1,2)	30	540	17062
IC(1,3)	25	500	11102
IC(2,4)	18	432	5668
DKR	32	512	40
MIC(1,2)	23	414	26
MIC(1,3)	20	400	24
SSOR $\omega = 1$ .	55	935	66162
SSOR $\omega = 1.6$	36	612	16620
BSSOR $\omega = 1$ .	41	738	33929
BSSOR $\omega = 1.5$	23	414	14777
BDIA	34	612	21489
POL(1,-1)	28	504	14182
INV(1)	22	396	8790
MINV(1)	17	306	20
CHOL(1)	24	432	10288
CHOL(2)	18	396	5531
CHOL(3)	14	364	3307
CHOL(4)	12	360	2154
CHOL(5)	10	340	1490
UND(2,3)	22	396	8946
UND(3,4)	17	374	4762
UND(4,5)	14	364	2876
UND(5,6)	12	360	1899
MUND(2,3)	19	342	5825
MUND(2,4)	17	306	3472
MUND(2,5)	16	288	2135
MUND(3,4)	15	330	3355
MUND(3,5)	14	308	2135
MUND(3,6)	14	308	1379
MUND(4,5)	12	312	2136
MUND(4,6)	12	312	1416
MUND(5,6)	11	330	1451
LJAC	111	1443	

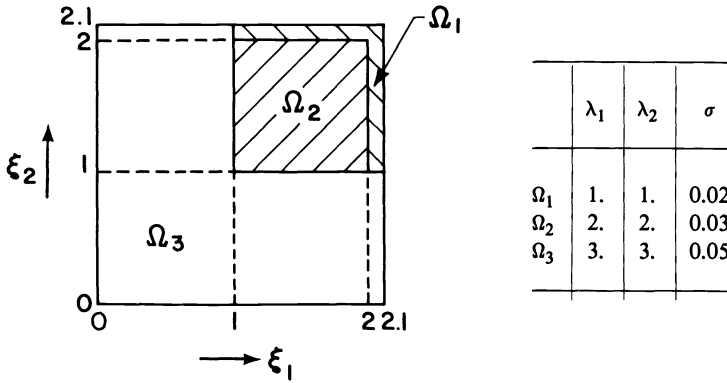


FIG. 6. Test problem 3.

TABLE 11

Number of iterations and total work per point for  $\|x^k\|_\infty \leq 10^{-6}$ .  
 Test problem 3,  $N = 1849$ .

$M$	# iterations	work/ $N$
IC(1,1)	74	1184
IC(1,2)	47	846
IC(1,3)	38	760
IC(2,4)	29	696
DKR	53	848
MIC(1,2)	36	648
MIC(1,3)	29	580
SSOR $\omega = 1$ .	88	1496
SSOR $\omega = 1.7$	52	884
BSSOR $\omega = 1$ .	65	1170
BSSOR $\omega = 1.5$	46	828
BDIA	52	936
POL(1,-1)	43	774
INV(1)	34	612
MINV(1)	25	450
CHOL(1)	36	648
CHOL(2)	28	616
CHOL(3)	22	572
CHOL(4)	19	570
CHOL(5)	16	544
UND(2,3)	34	612
UND(3,4)	26	572
UND(4,5)	21	546
UND(5,6)	18	540
MUND(2,3)	28	504
MUND(2,4)	25	450
MUND(2,5)	23	414
MUND(3,4)	23	506
MUND(3,5)	21	462
MUND(4,5)	19	494
MUND(4,6)	18	468
MUND(5,6)	17	510

TABLE 12

*Number of iterations and total work  
per point for  $\|r^k\|_2 \leq 10^{-6}$ .  
Test problem 3,  $N = 1849$ .*

$M$	# iterations	work/ $N$
IC(1,1)	79	1264
IC(1,2)	49	882
IC(1,3)	39	780
IC(2,4)	30	720
DKR	66	1056
MIC(1,2)	43	774
MIC(1,3)	35	700
SSOR $\omega = 1$ .	94	1598
SSOR $\omega_{opt}$	56	952
BSSOR $\omega = 1$ .	68	1224
BSSOR $\omega_{opt}$	48	864
BDIA	55	990
POL(1,-1)	45	810
INV(1)	36	648
MINV(1)	29	522
CHOL(1)	38	684
CHOL(2)	29	638
CHOL(3)	23	598
CHOL(4)	20	600
CHOL(5)	17	578
UND(2,3)	36	648
UND(3,4)	28	616
UND(4,5)	22	572
UND(5,6)	19	570
MUND(2,3)	30	540
MUND(2,4)	26	468
MUND(2,5)	24	432
MUND(3,4)	24	528
MUND(3,5)	22	484
MUND(4,5)	20	520
MUND(4,6)	19	494
MUND(5,6)	17	510

iterations with point preconditioning IC(5,7) is used in [17], but approximately the same goal can be achieved with only six instead of 12 vectors of storage using the block preconditioning CHOL(5).

**6. Concluding remarks.** The above examples show that, for linear problems coming from finite-difference approximations of elliptic partial differential equations, the block preconditionings we have introduced can give better results for two-dimensional problems than the corresponding point ones currently in use. The results are better also than for block SSOR preconditioning. Generally, for natural ordering of the unknowns, the modified methods give better results for our test problems than unmodified ones. Particularly attractive is the preconditioning INV(1)—and its modified form MINV(1)—because of the low storage require-

ments and rapid convergence. The results for three dimensional problems await further study. It would be of interest to explore the behavior of our block preconditioning methods on more general problems such as the ones arising from finite element approximation with node orderings leading to a block tridiagonal matrix.

**7. Acknowledgment.** We are pleased to acknowledge that much of this work has been stimulated by the paper of R. R. Underwood [20] and our personal association with him.

#### REFERENCES

- [1] E. ASPLUND, *Inverse of matrices  $\{a_{ij}\}$  which satisfy  $a_{ij} = 0$  for  $j > i + p$* , Math. Scand., 7 (1959), pp. 57-60.
- [2] P. CONCUS, G. H. GOLUB, AND G. MEURANT, *Block preconditioning for the conjugate gradient method*, Report LBL-14865, Lawrence Berkeley Lab., Univ. of California, 1982.
- [3] 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.
- [4] R. W. COTTLE, *Manifestations of the Schur complement*, Linear Algebra Appl., 8 (1974), pp. 120-211.
- [5] S. DEMKO, *Inverses of band matrices and local convergence of spline projections*, SIAM J. Numer. Anal., 14 (1977), pp. 616-619.
- [6] J. J. DONGARRA, C. B. MOLER, J. R. BUNCH, AND G. W. STEWART, *LINPACK Users' Guide*, Society for Industrial and Applied Mathematics, Philadelphia, 1979.
- [7] T. DUPONT, R. P. KENDALL, AND H. RACHFORD, *An approximate factorization procedure for solving self adjoint elliptic difference equations*, SIAM J. Numer. Anal., 5 (1968), pp. 559-573.
- [8] S. EISENSTAT, *Efficient implementation of a class of preconditioned conjugate gradient methods*, SIAM J. Sci. Statist. Comput., 2 (1981), pp. 1-4.
- [9] L. W. EHRLICH, *The block symmetric successive overrelaxation method*, SIAM J. Appl. Math., 12 (1964), pp. 807-826.
- [10] D. K. FADDEEV, *Properties of the inverse of a Hessenberg matrix*, in Numerical Methods and Computational Issues, 5 (1981), V. P. Il'in and V. N. Kublanovskaya, eds. (in Russian).
- [11] A. GEORGE AND J. W. H. LIU, *Algorithms for matrix partitioning and the numerical solution of finite element systems*, SIAM J. Numer. Anal., 15 (1978), pp. 297-327.
- [12] G. H. GOLUB AND G. MEURANT, *Résolution numérique des grands systèmes linéaires*, Collection de la Direction des Etudes et Recherches de l'Electricité de France, vol. 49, Eyrolles, Paris, 1983.
- [13] I. GUSTAFSSON, *A class of first order factorization methods*, BIT, 18 (1978), pp. 142-156.
- [14] O. G. JOHNSON, C. A. MICCHELLI, AND G. PAUL, *Polynomial preconditioners for conjugate gradient calculations*, SIAM J. Numer. Anal., 20 (1983), pp. 362-376.
- [15] D. KERSHAW, *Inequalities on the elements of the inverse of a certain tridiagonal matrix*, Math. Comp., 24 (1970), pp. 155-158.
- [16] J. A. MEIJERINK AND H. 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.
- [17] J. A. MEIJERINK AND H. VAN DER VORST, *Guidelines for the usage of incomplete decompositions in solving sets of linear equations as they occur in practical problems*, J. Comput. Phys., 44 (1981), pp. 134-155.
- [18] G. MEURANT, *The Fourier/tridiagonal method for the Poisson equation from the point of view of block Cholesky factorization*, Report LBID-764, Lawrence Berkeley Lab., Univ. of California, 1983.
- [19] C. MOLER, *MATLAB Users' Guide*, Dept. of Computer Science, Univ. of New Mexico, Albuquerque, NM, 1981.
- [20] R. R. UNDERWOOD, *An approximate factorization procedure based on the block Cholesky decomposition and its use with the conjugate gradient method*, Report NEDO-11386, General Electric Co., Nuclear Energy Div., San Jose, CA, 1976.
- [21] R. S. VARGA, *Matrix Iterative Analysis*, Prentice Hall, Englewood Cliffs, NJ, 1962.
- [22] D. M. YOUNG, *Iterative Solution of Large Linear Systems*, Academic Press, New York, 1971.

## A CONIC ALGORITHM FOR OPTIMIZATION\*

HERVÉ GOURGEON† AND JORGE NOCEDAL‡

**Abstract.** This paper describes a method that will minimize a conic function  $f$  in  $n$  steps, where  $n$  is the dimension of the domain of  $f$ . The algorithm can be considered a generalization of the conjugate gradient method, and has similar orthogonality properties. Some error bounds are given and the numerical stability of the algorithm is discussed.

**Key words.** optimization, conic function, conjugate gradient

**1. Introduction.** Most of the algorithms that are presently being used for solving unconstrained optimization problems are based on the idea of iteratively forming an approximating function, and minimizing it. The approximating function is usually a quadratic as in the class of quasi-Newton methods. Recently, Davidon noticed that the information provided by the problem could perhaps be used more productively if, instead of using a quadratic approximating function, one used a certain rational function called a conic (see Davidon (1980)). He made a thorough investigation of the interpolating properties of conic functions and of their geometry. His paper provides the essential mathematical background for the development of new algorithms.

In this paper we describe a method based on a conic approximating function. Like the conjugate gradient method, it does not require matrix storage. We will show how to derive this algorithm from the quadratic case. Our procedure can be used for deriving other optimization methods based on conics, which would be the analogues of quasi-Newton methods, variable storage methods, etc.

The distinctive feature of the algorithms based on conics is that they can use both function and gradient values to construct the approximating function, while the methods based on quadratics use only gradient values. One hopes that the new algorithms will be faster and more reliable than the current ones; this will not be known, however, until the conic methods are fully developed. This paper has the following limited objective: to develop a numerically stable method that will minimize a conic function in at most  $n$  steps, where  $n$  is the dimension of the domain of the function. This algorithm can be adapted for minimizing general nonlinear functions. However, the correct way of adapting the algorithm involves many difficult questions and we will leave their study for a future work.

The algorithm described here is related to one developed independently by Davidon (1982), and to several algorithms that are presently being studied by Davidon and Sorensen (1982). The method of Davidon (1982), which he calls the  $O(n)$  method, is also a generalization of the conjugate gradient method; however, its statement and derivation are quite different from those presented in this paper. Davidon (1982) describes another method, which is a generalization of the BFGS method and uses  $O(n^2)$  storage. Using the orthogonality results of § 5, one can show that Davidon's two algorithms and the one described here generate the same points when applied, with exact arithmetic, to normal conic functions. This is analogous to the case of quadratics, where a wide class of quasi-Newton and conjugate gradient methods are shown to be equivalent in the absence of round-off errors (see Huang (1970)).

---

\* Received by the editors March 19, 1982, and in final revised form November 3, 1983.

† Lycée Descartes, Tours, France.

‡ Courant Institute of Mathematical Sciences, New York University, New York, New York, 10012.

Even though the paper of Davidon (1982) and this one describe similar algorithms, the papers have little in common. Davidon concentrates on extending the notion of conjugacy to conics, while we study the properties of the method and its practical implementation.

Conic models have been studied in other types of methods. Schnabel (1982) considered a Newton-like algorithm, and Sorensen (1980) studied the convergence of a method that resembles the BFGS quasi-Newton method.

This paper is essentially self-contained. We assume only that the reader is familiar with the conjugate gradient method. All the tools needed for working with conics are developed. Most of these appear in a more general setting in Davidon’s extensive study (Davidon (1980)). To see how quasi-Newton methods construct a quadratic model consult Dennis and Moré (1977).

**2. Basic properties of conic functions.** Let  $x_0$  be a point in  $\mathbb{R}^n$  that will be called the *reference point*. For any  $x \in \mathbb{R}^n$  we will denote  $x - x_0$  by  $s$ :

$$s \equiv x - x_0.$$

DEFINITION 2.1. Let  $a$  be a vector in  $\mathbb{R}^n$  and let  $X = \{x \in \mathbb{R}^n : 1 - a^T s \neq 0\}$ . A *conic function*  $f: X \rightarrow \mathbb{R}$  is given by

$$(2.1) \quad f(x) = f_0 + \frac{g_0^T s}{1 - a^T s} + \frac{1}{2} \frac{s^T A s}{(1 - a^T s)^2},$$

where  $f_0 \in \mathbb{R}$ ,  $g_0 \in \mathbb{R}^n$  and  $A$  is a symmetric matrix of order  $n$ . If  $a = 0$ ,  $f$  is quadratic. Let us assume that  $a \neq 0$  and define  $H = \{x \in \mathbb{R}^n : 1 - a^T s = 0\}$ , which is an affine hyperplane. We will call  $H$  the *singular hyperplane* and we will write  $\gamma(x) = 1 - a^T s$ . We call  $\gamma(x)$  the gauge of  $f$  at  $x = x_0 + s$ ; the vector  $a$  will be referred to as the *horizon vector*. The conic function  $f$  can be transformed into a quadratic by introducing the variable

$$(2.2) \quad w = \frac{s}{1 - a^T s}.$$

More precisely, let  $W = \mathbb{R}^n \setminus \{w : a^T w = -1\}$  and define the map  $S: W \rightarrow X$  by

$$(2.3) \quad S(w) = x_0 + \frac{w}{1 + a^T w}.$$

Then  $f \circ S(w): W \rightarrow \mathbb{R}$  is quadratic:

$$(2.4) \quad h(w) \equiv f \circ S(w) = f_0 + g_0^T w + \frac{1}{2} w^T A w.$$

Note that (2.2) implies that

$$(2.5) \quad s = \frac{w}{1 + a^T w}.$$

For future reference we summarize (2.2) and (2.5):

$$(2.6) \quad w = \frac{s}{1 - a^T s} = \frac{s}{\gamma(x)}, \quad \gamma(x) = 1 - a^T s = \frac{1}{1 + a^T w}.$$

The map  $S$  will be used to translate the properties of the quadratic function  $h$  in the  $W$ -space into the properties of the conic  $f$  in the  $X$ -space. Note that  $S^{-1}(x) =$

$s/(1 - a^T s)$  and that the Jacobian matrix of  $S^{-1}$  is

$$(2.7) \quad J(x) = \frac{1}{\gamma(x)} \left( I + \frac{sa^T}{\gamma(x)} \right).$$

The matrix  $J(x)$  is invertible for all  $x \in X = \mathbb{R}^n \setminus H$ , and

$$(2.8) \quad J^{-1}(w) = \frac{1}{1 + a^T w} \left( I - \frac{wa^T}{1 + a^T w} \right) = \gamma(x)(I - sa^T).$$

In what follows,  $f$  represents a conic function in  $X = \mathbb{R}^n \setminus H$ ,  $g$  its gradient and  $x_0$  the reference point in (2.1). Then

$$(2.9) \quad h'(w) = (f \circ S)'(w) = J^{-T}(w)g(S(w))$$

and

$$(2.10) \quad g(x) = (h \circ S^{-1})'(x) = J^T(x)h'(S^{-1}(x)).$$

From (2.4) we have that

$$(2.11) \quad h'(w) = g_0 + Aw;$$

therefore (2.10) can be written as

$$(2.12) \quad g(x) = J^T(x) \left( g_0 + \frac{As}{\gamma(x)} \right).$$

Let us now characterize the critical points of a conic. Since  $J^T(x)$  is nonsingular, the gradient vanishes at  $x_0 + s_*$  iff

$$(2.13) \quad \gamma_* g_0 + As_* = 0$$

or

$$(2.14) \quad g_0 + (A - g_0 a^T) s_* = 0,$$

where  $\gamma_* = 1 - a^T s_*$ . The set of critical points of the quadratic function  $h$  given by (2.4) is an affine subspace which can be: (i) empty iff  $g_0 \notin \text{Im } A$ , (ii) a unique point iff  $A$  is invertible, and (iii) an affine subspace in the direction  $\text{Ker } A$ , otherwise. The map  $S$  preserves the maxima and minima because  $S$  is continuous and  $J$  is invertible. Therefore, the critical points of the conic  $f$  can be studied by considering their translation into  $W$ . We deduce from the quadratic function  $h$  the following, where the notation  $A > 0$  ( $A \geq 0$ ) means that  $A$  is positive definite (positive semi-definite):

- 1) The critical points of  $f$  in  $X$  are minimizers iff  $A \geq 0$ .
- 2) A minimizer of  $f$  in  $X$  is unique iff  $A > 0$ .
- 3) There are no critical points of  $f$  in  $X$  if  $g_0 \notin \text{Im } A$ , or if, for any  $z \in \mathbb{R}^n$  such that  $Az = g_0$ ,  $a^T z = 1$ .

From (2.13) we see that if  $A > 0$ , then  $s_* = -\gamma_* A^{-1} g_0$ , and

$$(2.15) \quad \gamma_*(1 - a^T A^{-1} g_0) = 1.$$

Therefore a condition that ensures that  $f$  has a unique minimizer in  $X$  is:

- 4) If  $A > 0$  and  $a^T A^{-1} g_0 \neq 1$ , there is a unique minimizer  $x_*$  of  $f$  in  $X$  given by  $x_* = x_0 + s_*$ , with

$$(2.16) \quad s_* = \frac{-A^{-1} g_0}{1 - a^T A^{-1} g_0}.$$

Finally, from (2.14) we have

- 5) The set of minimizers of  $f$  is the restriction to  $X$  of an affine subspace of direction  $\text{Ker}(A - g_0 a^T)$  if there is a  $z \in \mathbb{R}^n$  such that  $Az = g_0$  and  $a^T z \neq 1$ .

We are mainly interested in conic functions with unique minimizers.

DEFINITION 2.2. The conic function (2.1) is called *normal* if it has a unique minimizer, i.e., if  $A > 0$  and  $a^T A^{-1} g_0 \neq 1$ .

**3. Computation of the gauge.** The map  $S$  that made the composition  $f \circ S$  quadratic requires knowledge of the horizon vector  $a$ . Davidon (1982) has shown how to determine this vector using function and gradient values at three collinear points. He also showed (Davidon (1980)) how the gauge  $\gamma$  can be computed from function and gradient values, without knowledge of the horizon vector. These two observations are basic in the development of the algorithm, and we will now discuss them.

As will be done repeatedly in this paper, we first transform the function  $f$  into a quadratic. In that space we find a useful result and translate it into the original space. For all functions such as  $f, \gamma, g$ , etc., we will write  $f_k$  for  $f(x_k)$ , etc. Consider two points  $x_i, x_j \in X$  and the line joining them,  $x(\tau) = x_i + (x_j - x_i)\tau$ . Then, by linearity of  $\gamma, \gamma(x(\tau)) = \gamma_i + (\gamma_j - \gamma_i)\tau$ , where  $\gamma_k = 1 - a^T(x_k - x_0)$ . Multiplying (2.1) by  $\gamma^2$  shows that the function

$$(3.1) \quad q(\tau) = [\gamma_i + (\gamma_j - \gamma_i)\tau]^2 f(x(\tau))$$

is quadratic. Now, a quadratic  $q(\tau)$  satisfies

$$q(1) - q(0) = \frac{1}{2}[q'(0) + q'(1)].$$

From (3.1) this condition is equivalent to

$$(3.2) \quad f_j - f_i = \frac{1}{2} \left( \frac{\gamma_i}{\gamma_j} g_i + \frac{\gamma_j}{\gamma_i} g_j \right)^T (x_j - x_i).$$

If the conic function  $f$  is normal, then  $A > 0$  and the quadratic  $h = f \circ S$  is strictly convex; it satisfies

$$(3.3) \quad \frac{1}{2}[h'(w_i) - h'(w_j)]^T (w_i - w_j) > 0$$

for any  $w_i \neq w_j \in W$ . In the  $X$ -space this inequality becomes, using (2.6) and (2.11),

$$(3.4) \quad \rho_{ij} \equiv \frac{1}{2} \left( \frac{s_i}{\gamma_i} - \frac{s_j}{\gamma_j} \right)^T A \left( \frac{s_i}{\gamma_i} - \frac{s_j}{\gamma_j} \right) > 0.$$

The quantity  $\rho$  is important in the study of conics; there are other expressions for it, which we will now derive. From (2.8), and (2.9)

$$h'(w) - h'(0) = \gamma(x)(I - as^T)g(x) - g_0,$$

while from (2.6) and (2.11)

$$h'(w) - h'(0) = A \frac{s}{\gamma(x)}.$$

Let  $x_k = x_0 + s_k, k = i, j$ ; note that  $x_j - x_i = s_j - s_i$ . Equating the right-hand sides of the last two equations, and substituting into (3.4), we obtain

$$(3.5) \quad \rho_{ij} = \frac{1}{2} \left( \frac{\gamma_j}{\gamma_i} g_j - \frac{\gamma_i}{\gamma_j} g_i \right)^T (x_j - x_i).$$



Using (3.2) and (3.5) we have both

$$(3.6) \quad \rho_{ij} = f_i - f_j + \frac{\gamma_j}{\gamma_i} g_j^T(x_j - x_i) = f_j - f_i - \frac{\gamma_i}{\gamma_j} g_i^T(x_j - x_i).$$

Squaring (3.2) and (3.5), and comparing them, we obtain

$$(3.7) \quad \rho_{ij}^2 = (f_j - f_i)^2 - g_i^T(x_j - x_i)g_j^T(x_j - x_i).$$

Equations (3.6) and (3.7) establish the following theorem.

**THEOREM 3.1** (Davidon (1980)). *Let  $f$  be a normal conic function and let  $x_i$  and  $x_j$  be two points in  $X$ . Then*

$$(3.8) \quad v_{ji} \equiv \frac{\gamma_i}{\gamma_j} = \frac{-g_j^T(x_i - x_j)}{f_j - f_i + \rho_{ij}}$$

where

$$(3.9) \quad \rho_{ij} = [(f_j - f_i)^2 - g_i^T(x_j - x_i)g_j^T(x_j - x_i)]^{1/2}.$$

Note that  $\gamma(x_0) = 1$  so that Theorem 3.1, with  $i = k$  and  $j = 0$ , gives the value of the gauge at any point  $x_k \in X$  in terms of the function and gradient values at  $x_0$  and  $x_k$ . As  $\gamma$  is affine, knowledge of  $\gamma_0$  and  $\gamma_k$  allows us to find the value of  $\gamma$  along the line joining  $x_0$  and  $x_k$ .

We will now see that by evaluating the function and gradient at three collinear points we can determine the horizon vector  $a$  and hence the gauge  $\gamma$  in all the space.

**LEMMA 3.2** (Davidon (1982)). *Let  $x_1 = x_0 + \lambda_1 s$ ,  $x_2 = x_0 + \lambda_2 s$  be two points in  $X$ . Then the horizon vector is given by*

$$(3.10) \quad a = \frac{(\gamma_1/\lambda_1)(\gamma_1 g_1 - g_0) - (\gamma_2/\lambda_2)(\gamma_2 g_2 - g_0)}{(\gamma_1^2 g_1 - \gamma_2^2 g_2)^T s}.$$

*Proof.* From (2.7) and (2.12)

$$(3.11) \quad g_1 = \frac{1}{\gamma_1^3} (\gamma_1 I + \lambda_1 a s^T) (\gamma_1 g_0 + \lambda_1 A s).$$

Therefore

$$\gamma_1 g_0 + \lambda_1 A s = \gamma_1^2 (I - \lambda_1 a s^T) g_1.$$

Similarly

$$\gamma_2 g_0 + \lambda_2 A s = \gamma_2^2 (I - \lambda_2 a s^T) g_2,$$

$$A s = \frac{1}{\lambda_1} [\gamma_1^2 (I - \lambda_1 a s^T) g_1 - \gamma_1 g_0] = \frac{1}{\lambda_2} [\gamma_2^2 (I - \lambda_2 a s^T) g_2 - \gamma_2 g_0].$$

Hence

$$a [\gamma_1^2 s^T g_1 - \gamma_2^2 s^T g_2] = \frac{1}{\lambda_1} (\gamma_1^2 g_1 - \gamma_1 g_0) - \frac{1}{\lambda_2} (\gamma_2^2 g_2 - \gamma_2 g_0),$$

and from this (3.10) follows.

Observe that in Lemma 3.2,  $x_0$  coincides with the reference point in the definition of a conic function (2.1). We now generalize this result to accept any point.

LEMMA 3.3. *Let  $x_1, x_2, x_3$  be collinear points in  $X$ . Then the horizon vector is given by*

$$(3.12) \quad a = \frac{(\lambda_2 - \lambda_3)\gamma_1^2 g_1 + \lambda_3 \gamma_2^2 g_2 - \lambda_2 \gamma_3^2 g_3}{[(\lambda_2 - \lambda_3)\gamma_1^2 s_1^T g_1 + \lambda_3 \gamma_2^2 s_2^T g_2 - \lambda_2 \gamma_3^2 s_3^T g_3]}$$

where  $s_i = x_i - x_0, i = 1, 2, 3, x_i = x_1 + \lambda_i d, i = 2, 3$ .

*Proof.* From (2.12)

$$(3.13) \quad g_i = \frac{1}{\gamma_i} J_i^T (\gamma_i g_0 + A s_i), \quad i = 1, 2, 3.$$

Combining (3.13) for  $i = 1$  and  $i = 2$ , we have

$$(3.14) \quad \lambda_2 A d = A s_2 - A s_1 = \gamma_2 J_2^{-T} g_2 - \gamma_1 J_1^{-T} g_1 - (\gamma_2 - \gamma_1) g_0.$$

Similarly for  $i = 1$  and  $i = 3$ ,

$$(3.15) \quad \lambda_3 A d = A s_3 - A s_1 = \gamma_3 J_3^{-T} g_3 - \gamma_1 J_1^{-T} g_1 - (\gamma_3 - \gamma_1) g_0.$$

Using (2.8), we obtain both

$$(3.16) \quad \begin{aligned} A d &= \frac{1}{\lambda_2} [\gamma_2^2 (I - a s_2^T) g_2 - \gamma_1^2 (I - a s_1^T) g_1] - \frac{1}{\lambda_2} (\gamma_2 - \gamma_1) g_0 \\ &= \frac{1}{\lambda_3} [\gamma_3^2 (I - a s_3^T) g_3 - \gamma_1^2 (I - a s_1^T) g_1] - \frac{1}{\lambda_3} (\gamma_3 - \gamma_1) g_0. \end{aligned}$$

Note that for  $i = 2$  or  $3$

$$\frac{1}{\lambda_i} (\gamma_i - \gamma_1) = -\frac{1}{\lambda_i} a^T (s_i - s_1) = -a^T d.$$

Therefore (3.16) implies

$$\begin{aligned} &a \left[ \frac{1}{\lambda_2} (\gamma_2^2 s_2^T g_2 - \gamma_1^2 s_1^T g_1) - \frac{1}{\lambda_3} (\gamma_3^2 s_3^T g_3 - \gamma_1^2 s_1^T g_1) \right] \\ &= \frac{1}{\lambda_2} (\gamma_2^2 g_2 - \gamma_1^2 g_1) - \frac{1}{\lambda_3} (\gamma_3^2 g_3 - \gamma_1^2 g_1), \end{aligned}$$

and hence

$$a [(\lambda_2 - \lambda_3)\gamma_1^2 s_1^T g_1 + \lambda_3 \gamma_2^2 s_2^T g_2 - \lambda_2 \gamma_3^2 s_3^T g_3] = (\lambda_2 - \lambda_3)\gamma_1^2 g_1 + \lambda_3 \gamma_2^2 g_2 - \lambda_2 \gamma_3^2 g_3. \quad \square$$

**4. One-dimensional minimization.** The algorithm to be presented in § 5 requires a one-dimensional minimization of a conic at every step. We now study how to perform it and discuss some potential difficulties.

Consider the restriction of  $f$  to the line  $x(\tau) = x_0 + s(\tau)$ , where  $s(\tau) = (1 - \tau)s_1 + \tau s_2$ . Note that this is a line through  $x_1 = x_0 + s_1$  and  $x_2 = x_0 + s_2$ , and does not include the point  $x_0$ . Let us call  $\bar{\tau} = (1 - \tau)$ . As  $\gamma$  is affine,  $\gamma(x(\tau)) = \bar{\tau}\gamma_1 + \tau\gamma_2$ . We have, therefore,

$$(4.1) \quad \begin{aligned} f(x_0 + s(\tau)) &= f_0 + (\bar{\tau}g_0^T s_1 + \tau g_0^T s_2) / \gamma + \frac{1}{2} (\bar{\tau}^2 s_1^T A s_1 + \tau^2 s_2^T A s_2) / \gamma^2 \\ &\quad + \bar{\tau}\tau s_1^T A s_2 / \gamma^2, \end{aligned}$$

where  $\gamma$  stands for  $\gamma(x(\tau))$ . From (3.4)

$$\rho_{12} = \frac{1}{2} \begin{pmatrix} s_1 & s_2 \end{pmatrix}^T A \begin{pmatrix} s_1 & s_2 \\ \gamma_1 & \gamma_2 \end{pmatrix};$$

therefore

$$(4.2) \quad \bar{\tau}\tau\gamma_1\gamma_2\rho_{12} = \frac{1}{2}\bar{\tau}\tau\left(\frac{\gamma_2}{\gamma_1}s_1^TAs_1 + \frac{\gamma_1}{\gamma_2}s_2^TAs_2\right) - \bar{\tau}\tau(s_1^TAs_2).$$

As  $\bar{\tau}\tau\gamma_2/\gamma_1 = \bar{\tau}(\gamma/\gamma_1 - \bar{\tau})$  and  $\bar{\tau}\tau\gamma_1/\gamma_2 = \tau(\gamma/\gamma_2 - \tau)$ , one can show, using (4.2), that

$$(4.3) \quad f(x_0 + s(\tau)) = [(1 - \tau)\gamma_1f_1 + \gamma_2\tau f_2]/\gamma - [(1 - \tau)\tau\gamma_1\gamma_2\rho_{12}]/\gamma^2.$$

Differentiating, we obtain, using (3.5) and (3.6),

$$(4.4) \quad \frac{df(x_0 + s(\tau))}{d\tau} = g(x_0 + s(\tau))^T(s_2 - s_1) = \frac{[(1 - \tau)g_1 + \tau\nu_{12}^3g_2]^T}{(1 - \tau + \nu_{12}\tau)^3}(s_2 - s_1),$$

where

$$(4.5) \quad \nu_{ij} = \frac{\gamma_j}{\gamma_i}.$$

From (4.4) we see that  $df/d\tau$  vanishes at

$$(4.6) \quad \tau^* = \frac{-g_1^T(s_2 - s_1)}{(\nu_{12}^3g_2 - g_1)^T(s_2 - s_1)},$$

provided that the denominator is nonzero.

For some directions, the one-dimensional minimizer may not lie on the same side of the singular hyperplane as the solution of the problem. Problem 1 of § 8 illustrates this situation. This causes no difficulties and we will allow the algorithm to produce iterates on both sides of the singular hyperplane.

Another interesting fact (Sorensen (1982)) is that along some directions a one-dimensional minimizer does not exist. This is true even for normal conic functions. We will call these directions *special directions* and we can construct them as follows. Find a point where the contour lines of the quadratic function  $h = f \circ S$  intersect the affine hyperplane  $T = \{w \in \mathbb{R}^n : a^T w = -1\}$ . Then choose a line that is tangent to the contour line of  $h$  at this point. Transforming this line into the  $X$ -space, using  $S$ , will produce a special direction. We will discuss these directions further in the next section.

**5. A conjugate-direction conic method.** The method is based on the following idea. In order to minimize the normal conic  $f$ , consider the quadratic  $h = f \circ S$  and apply the conjugate gradient method to it. Then transform the iterates back to obtain a minimization method for  $f$ .

While minimizing the quadratic function  $h: W \rightarrow \mathbb{R}$ , we will denote the search directions by  $v_0, v_1, \dots, v_k$ , and the displacements by  $\mu_0v_0, \dots, \mu_kv_k$ . As  $S^{-1}(x_0) = 0$  the initial point in the new coordinates is  $w_0 = 0$ . Thus the  $k$ th iterate (or total displacement) is

$$(5.1) \quad w_k = w_0 + \sum_{i=0}^{k-1} \mu_i v_i = \sum_{i=0}^{k-1} \mu_i v_i.$$

The point  $w_k$  in  $W$  is mapped by  $S$  into the point  $x_0 + w_k/(1 + a^T w_k)$ , in  $X$ . Let us denote the search directions in  $X$  by  $d_0, \dots, d_k$ ; the displacements by  $\lambda_0d_0, \dots, \lambda_kd_k$ ; the total displacement by

$$(5.2) \quad s_k = \sum_{i=0}^{k-1} \lambda_i d_i.$$

The correspondence between  $s_k$  and  $w_k$  is given by (2.6):

$$s_k = \frac{w_k}{1 + a^T w_k} \quad \text{or} \quad w_k = \frac{s_k}{1 - a^T s_k}.$$

We will now find a relation between  $d_k$  and  $v_k$ .

$$(5.3) \quad \begin{aligned} \mu_k v_k = w_{k+1} - w_k &= \frac{s_{k+1}}{1 - a^T s_{k+1}} - \frac{s_k}{1 - a^T s_k} \\ &= \frac{s_{k+1} - s_k}{\gamma_{k+1}} - s_k \left( \frac{1}{\gamma_k} - \frac{1}{\gamma_{k+1}} \right) \end{aligned}$$

$$(5.4) \quad = \frac{1}{\gamma_{k+1}} \left( I + \frac{s_k a^T}{\gamma_k} \right) \lambda_k d_k$$

or

$$(5.5) \quad \lambda_k d_k = \gamma_{k+1} (I - s_k a^T) \mu_k v_k.$$

We now express  $\gamma_{k+1}$  using information at  $x_k$ . Using (2.6), we have

$$\begin{aligned} 1 + \gamma_k a^T \mu_k v_k &= 1 + \gamma_k a^T w_{k+1} - \gamma_k a^T w_k \\ &= 1 + \gamma_k (1 + a^T w_{k+1}) - \gamma_k (1 + a^T w_k) = \gamma_k / \gamma_{k+1}. \end{aligned}$$

Hence  $\gamma_{k+1} = \gamma_k / (1 + \gamma_k a^T \mu_k v_k)$  and (5.5) can be written as

$$(5.6) \quad \lambda_k d_k = \frac{\gamma_k (I - s_k a^T) \mu_k v_k}{1 + \gamma_k a^T \mu_k v_k}.$$

Let us split this expression as

$$(5.7) \quad \begin{aligned} d_k &= \frac{\gamma_k (I - s_k a^T) v_k}{1 + \gamma_k a^T v_k} = \frac{J_k^{-1} v_k}{1 + \gamma_k a^T v_k}, \\ \lambda_k &= \frac{\mu_k (1 + \gamma_k a^T v_k)}{1 + \gamma_k a^T \mu_k v_k}, \end{aligned}$$

so that  $\mu_k = 1$  corresponds to  $\lambda_k = 1$ .

We will now derive the algorithm. Assume for the moment that no special directions are generated, i.e., that along all the directions considered, the normal conic function has a minimizer. Later we will discuss how to cope with special directions. We apply the conjugate gradient method of Hestenes and Stiefel (1952) with exact line searches to minimize  $h = f \circ S$ .

Choose an initial point  $x_0$ , which will be taken as the reference point in the definition (2.1) of the conic  $f$ . Evaluate  $f_0$  and  $g_0$ . Recall that  $S^{-1}(x_0) = w_0 = 0$  so that the map  $S$  is initially the identity map. Hence we have from (2.3) and (2.9) that  $h(w_0) = f(x_0)$  and  $h'(w_0) = g(x_0)$ . Take as initial search direction the one of steepest descent,  $d_0 = -g_0$ . Choose a step length  $\lambda_m$  and evaluate the function and gradient at  $x_m = x_0 + \lambda_m d_0$ . Find the minimizer  $x_p$  along the direction  $d_0$  by means of (4.6), and evaluate the function and gradient at  $x_p$ . With this information we determine the horizon vector  $a$  using (3.10). The first iterate is  $x_1 = x_p$ . From (2.6), (5.1) and (5.2) it follows that  $v_0$  is parallel to  $d_0$ . Therefore in  $W$  the first step is also a steepest descent step, and the first iterate  $w_1$  is the minimizer of  $h$  along the gradient direction. Hence  $h'_1 v_0 = 0$ .

At the  $k$ th step the search direction is

$$(5.8) \quad v_k = -h'_k + \beta_k v_{k-1}$$

with

$$(5.9) \quad \beta_k = \frac{(h'_k)^T A v_{k-1}}{v_{k-1}^T A v_{k-1}}.$$

The translation of (5.8) and (5.9) into the  $X$  space is, using (2.9) and (2.11),

$$(5.10) \quad h'_k = J_k^{-T} g_k = \gamma_k (I - a s^T) g_k,$$

$$(5.11) \quad \mu_{k-1} A v_{k-1} = h'_k - h'_{k-1} = J_k^{-T} g_k - J_{k-1}^{-T} g_{k-1}.$$

From (5.4) and (2.7),

$$(5.12) \quad \mu_{k-1} v_{k-1} = \frac{\gamma_{k-1}}{\gamma_k} J_{k-1} \lambda_{k-1} d_{k-1}.$$

Some simplifications are possible. As we made an exact line search during the first step,  $g_1^T d_0 = g_1^T s_1 = 0$ . This implies that  $h'_1$  and  $g_1$  are parallel; in fact  $h'_1 = \gamma_1 g_1$ .

Consider the second step. The property of increasing subspace minimization of the conjugate gradient method for  $h$  implies that  $h_2'^T v_0 = h_2'^T v_1 = 0$ . Therefore

$$(5.13) \quad h_2'^T (\mu_0 v_0 + \mu_1 v_1) = h_2'^T w_2 = \frac{1}{\gamma_2} h_2'^T s_2 = 0$$

or

$$g_2^T (I - s_2 a^T) s_2 = \gamma_2 g_2^T s_2 = 0,$$

so that  $g_2^T s_2 = 0$ . Similarly, as  $h_2'^T v_0 = 0$  and  $v_0 // s_1$ , we have  $g_2^T s_1 = 0$ . Therefore  $g_2^T d_1 = g_2^T (s_2 - s_1) = 0$  and  $h_2' = \gamma_2 g_2$ . Proceeding by induction, we can show that  $g_k^T s_j = 0$  for  $j \leq k$ ;  $g_k^T d_j = 0$  for  $j < k$ , and that the gradients of  $h$  and  $f$  are parallel.

The previous discussion establishes the following theorem.

**THEOREM 5.1.** *Let  $f$  be a normal conic function and assume that none of the search directions generated by the algorithm is a special direction. Then*

- 1)  $g_k^T d_j = 0$  for  $j < k$  (increasing subspace minimization);
- 2)  $h'_k = \gamma_k g_k$ ,  $k = 0, 1, \dots$ ;
- 3)  $g_k^T g_j = 0$  for  $j < k$ .

Note that (3) follows from (2) and from the orthogonality of the gradients  $h'_k$ .

We can now simplify (5.10) and (5.11):

$$(5.14) \quad h'_k = \gamma_k g_k$$

and

$$(5.15) \quad \mu_{k-1} A v_{k-1} = \gamma_k g_k - \gamma_{k-1} g_{k-1}.$$

Substituting (5.9), (5.14) and (5.15) into (5.8), we have

$$(5.16) \quad v_k = -\gamma_k g_k + \frac{\gamma_k g_k^T y_k}{v_{k-1}^T y_k} v_{k-1},$$

where  $y_k \equiv \gamma_k g_k - \gamma_{k-1} g_{k-1}$ . Using (5.12), we obtain

$$(5.17) \quad v_k = -\gamma_k g_k + \frac{\gamma_k g_k^T y_k}{y_k^T J_{k-1} d_{k-1}} J_{k-1} d_{k-1}.$$

There are other simplifications:

(1) From (2.7), (2.8) and using  $s_k - s_{k-1} = \lambda_{k-1}d_{k-1}$ , one finds

$$(5.18) \quad J_k^{-1}J_{k-1}d_{k-1} = \frac{\gamma_k^2}{\gamma_{k-1}^2}d_{k-1}.$$

(2)

$$(5.19) \quad a^T J_{k-1} d_{k-1} = a^T d_{k-1} / \gamma_{k-1}^2.$$

(3) By the orthogonality of  $g_k$  and  $s_j$  for  $j \leq k$

$$(5.20) \quad J_{k-1}^T g_k = g_k / \gamma_{k-1}, \quad J_{k-1}^T g_{k-1} = g_{k-1} / \gamma_{k-1}.$$

We now substitute (5.17)–(5.20) into (5.7); thus

$$(5.21) \quad d_k = \left[ -\gamma_k^2 g_k + \gamma_k^2 s_k a^T g_k + \frac{\gamma_k^3 (g_k^T y_k)}{\gamma_{k-1} y_k^T d_{k-1}} d_{k-1} \right] / \eta$$

where

$$(5.22) \quad \eta = 1 - \gamma_k^2 a^T g_k + \frac{\gamma_k^2 g_k^T y_k a^T d_{k-1}}{y_k^T d_{k-1} \gamma_{k-1}}.$$

The denominator  $\eta$  was kept to scale the vector  $d_k$ .

Let us now consider what we could do if a special direction is generated. A minimizer along that direction does not exist in the  $X$ -space. However, let us transform this direction into the  $W$ -space using the map  $S^{-1}$ . The quadratic  $h$  has a minimizer along the transformed direction. Therefore, when a special direction is encountered, we proceed as follows. Map the direction into the  $W$ -space and perform the one-dimensional minimization there. Then generate a new search direction by applying the conjugate gradient method to  $h$ . Finally we transform this new direction back into the  $X$ -space and proceed as usual. Note that a special direction can be identified by the fact that the denominator in (4.6) is zero.

With this change, the algorithm will find the minimizer of a normal conic function in at most  $n$  steps. This follows from the quadratic termination property of the conjugate gradient method (or from Theorem 5.1).

In what follows we will assume that special directions never occur. There are two reasons for doing so. First, it is extremely unlikely that a special direction is generated. Second, our interest does not lie in minimizing conic functions per se, but in deriving an algorithm that can be extended for use with general nonlinear functions. If, while constructing a conic model for nonlinear optimization, we discover that it has no minimizer along the current search direction, we will discard the model. Therefore a strategy to cope with special directions will never be needed in practice.

**6. Description of the algorithm.** We will now give a detailed description of the method for minimizing normal conic functions. Let EPS denote the “machine epsilon,” or the greatest number such that  $\text{fl}(1 + \text{EPS}) = 1$ . Let  $\text{EPS1} > \text{EPS}$  be a small number (e.g.  $10\sqrt{\text{EPS}}$ ). Choose a starting point  $x_0 \in X$ .

*Step 0.* Evaluate  $f$  and  $g$  at  $x := x_0$ . Set  $d := -g$ ,  $\gamma := 1$ .

*Step k.* Set  $f_k := f$ ,  $g_k := g$ ,  $x_k := x$ . Choose a trial steplength  $\lambda_m$  (see the next section for some possible choices) and compute  $x_m := x_k + \lambda_m d$ . Evaluate  $f$  and  $g$  at  $x_m$ .

$$(6.1) \quad (\text{see 3.9}) \quad \rho := (f_k - f_m)^2 - (g_k^T d)(g_m^T d)\lambda_m^2$$

$$(6.2) \quad \rho := \rho^{1/2}$$

$$(6.3) \quad (\text{see 3.8}) \quad \nu_m := -\lambda_m(g_k^T d)/(f_k - f_m + \rho)$$

$$(6.4) \quad (\text{see 4.6}) \quad \lambda_p := -\lambda_m(g_k^T d)/[\nu_m^3 g_m^T d - g_k^T d]$$

$$(6.5) \quad \nu_p := 1 + (\nu_m - 1)\lambda_p/\lambda_m$$

$$x := x_k + \lambda_p d.$$

Evaluate  $f$  and  $g$  at  $x$ .

Test convergence:

If  $\|g\| < \text{EPS1}$

exit. Otherwise compute

$$\sigma_p := g^T d.$$

(Only during the first step) compute  $a$ :

$$(6.6) \quad (\text{see 3.10}) \quad w := \sigma_p \nu_p^2 - (g_m^T d) \nu_m^2$$

$$v_1 := (\nu_p g - g_k)/\lambda_p w$$

$$v_2 := (\nu_m g_m - g_k)/\lambda_m w$$

$$a := \nu_p v_1 - \nu_m v_2$$

$$\gamma_k := \gamma$$

$$(6.7) \quad \gamma := \gamma \nu_p$$

$$y := \gamma g - \gamma_k g_k$$

$$\alpha := \gamma^2 g^T y / d^T y$$

(see 5.11–5.22)

$$d := \gamma^2(-g + [x - x_0]a^T g) + \alpha \gamma d / \gamma_k$$

$$(6.8) \quad d := d / (1 - \gamma^2 a^T g + \alpha a^T d / \gamma_k)$$

*End of Step k.*

We now comment on this algorithm. The ratio of gauges is denoted by  $\nu$ , so that  $\nu_m = \gamma_m / \gamma_k$  and  $\nu_p = \gamma_p / \gamma_k$  in (6.3), (6.5). Formula (6.5) follows from the fact that the gauge is an affine function. The gauge vector is computed only once, at the beginning of the iteration. If it has to be recomputed later, we must use (3.12) instead of (6.6). Note that once the starting point  $x_0$  is chosen, we think of it as the reference point in the definition of the conic function. Therefore  $\gamma_0 = 1$ , and in (6.7) we are computing the value of the gauge at the current point. This is also why in (6.6) we used (3.10). For a normal conic function (6.1) produces a positive number so that the square root operation is well defined. The denominator in (6.4) is zero only if  $d$  is a special direction.

**7. Numerical stability and error bounds.** Finding the minimizer of  $f$  is equivalent to solving the linear system (2.14). This system indicates whether or not the problem of locating the minimizer of  $f$  is well conditioned. Even if it is well conditioned, the algorithm may be unstable at some intermediate point of the computations. This is due to the nature of conic functions. Recall that the function values tend to infinity as we approach the  $(n-1)$ -dimensional hyperplane  $H$ . It is therefore important to implement the algorithm in a numerically stable form, so as not to aggravate the intrinsic difficulties of the problem.

A critical stage of the process is the choice of the trial steplength  $\lambda_m$ . Suppose that  $\lambda_m$  is so small that  $g_0^T d$  and  $g_m^T d$  are almost equal. Then the two quantities in (6.1) will be of almost the same magnitude and cancellation will occur. When  $g_0^T d \cdot g_m^T d > 0$  the computation of  $\rho$ , (6.1), can be improved somewhat by writing  $\rho = (a + b)(a - b)$ , where  $a = (f_0 - f_m)$  and  $b = \lambda_m (g_0^T d g_m^T d)^{1/2}$ . However, the only real cure is to increase  $\lambda_m$ . Therefore after (6.1) we include a test to ensure that the computation of  $\rho$  is accurate. If a substantial loss of significant figures has occurred, we double  $\lambda_m$ . With this precaution the algorithm has performed quite well even with rather ill conditioned problems.

We will now mention some possible choices for  $\lambda_m$ . By the way we scaled the search direction, a step of length 1 in the  $W$ -space corresponds to a step of length 1 in  $X$  (see (5.7)). However  $\lambda_m = 1$  may not be a good choice. Shanno and Phua (1980) use the trial step

$$(7.1) \quad \mu_m = \hat{g}^T \hat{d} \hat{\lambda} / g_0^T d,$$

where  $\hat{g}^T \hat{d}$  is the directional derivative at the beginning of the previous iteration, and  $\hat{\lambda}$  was the steplength to the minimizer. Using (7.1) in (5.7), we obtain one possible trial steplength. We could also take  $\lambda_m$  as the right-hand side of (7.1), thereby reasoning directly in the  $X$ -space. This last choice was used in the tests described in the following section.

When the iterates approach the solution  $x_*$ ,  $(f_0 - f_m)$  is of order  $O(\|x_0 - x_*\|^2)$ . On the other hand,  $f_0$  and  $f_m$  can be assumed to be of order  $O(1)$ . Therefore, when  $\|x_0 - x_*\| \leq \text{EPS}^{1/2}$ , the computation of  $(f_0 - f_m)$  in (6.1) will have no correct significant figures. The tolerance  $\text{EPS1}$  should therefore be greater than  $\text{EPS}^{1/2}$ . Note that the conjugate gradient method only computes quantities of order  $O(\|x_0 - x_*\|)$ , and can get closer to the solution than the conic method.

We now turn to the question of finding error bounds for the conic algorithm. We will use the results for the conjugate gradient method, since our method consists of applying it to the transformed function  $h$ . Let  $x_* = x_0 + s_*$ ; then from (2.1) and (2.13)

$$f(x) - f(x_*) = \frac{1}{2} \left( \frac{s_*}{\gamma_*} - \frac{s}{\gamma} \right)^T A \left( \frac{s_*}{\gamma_*} - \frac{s}{\gamma} \right) = \frac{1}{2} (w_* - w)^T A (w_* - w) \equiv E(w).$$

From the quadratic case (see, for example, Luenberger (1973)), we have

$$(7.2) \quad E_k \leq \left( \frac{\sigma_{n-k} - \sigma_0}{\sigma_{n-k} + \sigma_0} \right)^2 E_0,$$

where  $\sigma_0 \leq \sigma_1 \leq \dots \leq \sigma_{n-1}$  are the eigenvalues of  $A$ . It also follows that the number of iterations needed by the conic algorithm is less than or equal to the number of distinct eigenvalues of  $A$ . Note, however, that the matrix  $A$  depends on the choice of the initial point  $x_0$ . If we take any other point  $x_1 \in X$ , the conic (2.1) can be written as

$$(7.3) \quad f(x) = f_1 + \frac{g_1^T s}{1 - a_1^T s} + \frac{\frac{1}{2} s^T A_1 s}{(1 - a_1^T s)^2},$$

where  $a_1$  and  $A_1$  will differ from  $a$  and  $A$  and  $s = x - x_1$ . To investigate the relationship between the eigenvalues of  $A$  and  $A_1$ , it is convenient to consider the Hessian matrix at the solution. From (2.10) we have

$$(7.4) \quad \nabla^2 f(x_*) = J_*^T A J_*,$$

where  $J_* = (1/\gamma_*)(I + (s_* a^T)/\gamma_*)$  and  $\gamma_* = 1 - a^T s_*$ . Let  $\hat{s}$  be the displacement to the



solution from  $x_1$ , i.e.,  $x_1 + \hat{s} = x_0 + s_* = x_*$ . Then

$$(7.5) \quad \nabla^2 f(x_*) = \hat{J}^T A_1 \hat{J},$$

where  $\hat{J} = (1/\hat{\gamma})(I + (\hat{s}a_1^T)/\hat{\gamma})$  and  $\hat{\gamma} = 1 - a_1^T \hat{s}$ . Equating (7.4) and (7.5), we obtain

$$A = J_*^{-T} \hat{J}^T A_1 \hat{J} J_*^{-1},$$

which can be written as  $A = (I + R)^T A_1 (I + R)$ , where  $R$  is a matrix of rank  $\leq 2$ . In general, all the eigenvalues of  $A$  and  $A_1$  will be different, and therefore different initial points will lead to different numbers of iterations for convergence.

**8. Numerical examples.** We now present three examples to show the behavior of the conic method. We used a VAX 11/780 at the Courant Mathematics and Computing Laboratory, with approximately 16 decimal digits of accuracy in double precision. All the test functions are normal conic functions.

*Problem 1.*  $N = 2, x_0 = 0,$

$$A = \begin{bmatrix} 4 & 0 \\ 0 & 1 \end{bmatrix}, \quad a = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \quad g_0 = \begin{bmatrix} 3 \\ -3 \end{bmatrix}, \quad x_* = \begin{bmatrix} -3 \\ 12 \end{bmatrix},$$

$\gamma_* = 4$ . This is the example mentioned in § 4. After the first iteration  $\|g_1\| = .509$  and  $\gamma_1 = -5$ . One more iteration gives  $\|g_2\| = .439E-16$  and  $\gamma_2 = 4$ . The singular hyperplane was crossed twice.

*Problem 2.* Here  $N = 11, x_0 = 0,$

$$A = \begin{bmatrix} 5 & 2 & 1 & 1 & & & & & & & & \\ 2 & 6 & 3 & 1 & 1 & & & & & & & \\ 1 & 3 & 6 & 3 & 1 & 1 & & & & & & \\ 1 & 1 & 3 & 6 & 3 & 1 & 1 & & & & & \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ & & & 1 & 1 & 3 & 6 & 3 & 1 & 1 & & \\ & & & & 1 & 1 & 3 & 6 & 3 & 1 & & \\ & & & & & 1 & 1 & 3 & 6 & 2 & & \\ & & & & & & 1 & 1 & 2 & 5 & & \end{bmatrix},$$

$$a(i) = -\frac{1}{2}i, \quad i = 1, \dots, 5, \quad a(i) = .01, \quad i = 6, \dots, 11.$$

$$g_0(i) = i^2/1000, \quad i = 1, \dots, 5, \quad g_0(i) = 1, \quad i = 6, \dots, 11, \quad \gamma_* \doteq 1.55.$$

The matrix  $A$  has only 10 different eigenvalues; see Gregory and Karney (1969). After 10 iterations of the conic method,  $\|g\| = .288E-11$ ; the solution was found to full accuracy. Then we tried the initial point  $x_0 = .3$ ; the algorithm now required 11 iterations to obtain the solution.

As could also be expected, the conjugate gradient method was unsuccessful when applied to (nonquadratic) conic functions.

*Problem 3.* Here we compare the numerical stability of the conic algorithm and the conjugate gradient method by applying them to increasingly ill conditioned quadratic functions.  $A$  is the Hilbert matrix of order  $n$ , with  $A_{ij} = 1/(i + j - 1)$ ,  $x_0 = 0$ ,  $a = 0$  and  $g_0^T = (1, 1, \dots, 1)$ . In the following table we give information about the last iteration of each run. With exact arithmetic the conic and quadratic methods are identical. In the tests both methods produced the same iterates except in the last few iterations.

TABLE 1

N	cond (A)	Conic method		Conjugate gradient method	
		last iteration	$\ g\ $	last iteration	$\ g\ $
3	.53E+03	3	.121E-12	3	.112E-14
4	.15E+05	4	.443E-09	4	.254E-10
5	.47E+06	6	.164E-07	6	.389E-07
6	.16E+08	9	.858E-06	8	.123E-06

As discussed in § 7, it was expected that the conjugate gradient method would be more accurate for quadratics. Note that the conic method is only slightly more sensitive to ill conditioning. We note that using  $\lambda_m = 1$  and not including the precautions described in § 7 is very detrimental to the conic method. Severe cancellation occurs in the computation of  $\rho$  and the algorithm is unable to approach the solution. However, with the refinements described earlier the conic method copes adequately with ill conditioning.

**9. Final remarks.** A conic algorithm for minimizing a general nonlinear function of one variable is uniquely specified. It was analyzed by Bjørstad and Nocedal (1979), who showed that its rate of convergence is quadratic. Note that the restriction of variable metric methods to one dimension yields the secant method, whose rate of convergence is  $1.618 \dots$ . The conic method is therefore faster in one dimension, and this suggests that it might be faster in  $N$  dimensions as well. This, however, has not yet been established.

To adapt the conic method described in this paper for use with general nonlinear functions of  $N$  variables, several important issues need to be resolved:

1. How often should we recompute the horizon vector?
2. The method may fail to produce a descent direction, if the horizon vector is changed. Should we restart the iteration?
3. How do we decide if three collinear points are suitable for building a conic model?

**Acknowledgments.** We are happy to express our gratitude to W. Davidon for his encouragement, insights and suggestions. We would also like to thank D. Sorensen and the referees for their excellent comments, and S. Bedard for help in the preparation of this paper.

## REFERENCES

- P. BJØRSTAD AND J. NOCEDAL (1979), *Analysis of a new algorithm for one-dimensional minimization*, Computing, 22, pp. 93-100.
- W. C. DAVIDON (1980), *Conic approximations and collinear scalings for optimizers*, SIAM J. Numer. Anal., 17, pp. 268-281.
- (1982), *Conjugate directions for conic functions*, in Nonlinear Optimization 1981, M. J. D. Powell, ed., Academic Press, New York.
- W. C. DAVIDON AND D. SORENSEN (1982), Unpublished manuscript.
- J. DENNIS AND J. MORÉ (1977), *Quasi-Newton methods, motivation and theory*, SIAM Rev., 19, pp. 46-89.
- M. HESTENES AND E. STIEFEL (1952), *Methods of conjugate gradients for solving linear systems*, J. Res. Nat. Bur. Standards, 49, pp. 409-436.
- H. HUANG (1970), *Unified approach to quadratically convergent algorithms for function minimization*, J. Optim. Theory Appl., 5, pp. 405-423.

- D. LUENBERGER (1973), *Introduction to Linear and Nonlinear Programming*, Addison-Wesley, Reading, MA.
- R. GREGORY AND D. KARNEY (1969), *A Collection of Matrices for Testing Computational Algorithms*, Wiley Interscience, New York.
- R. SCHNABEL (1982), *Conic methods for unconstrained minimization and tensor methods for nonlinear equations*, Dept. Computer Science, Report CU-227-82, Univ. Colorado, Boulder.
- D. SHANNO AND K. PHUA (1980), *Remark on Algorithm 500*, ACM Trans. Math. Software, 6, pp. 618–622.
- D. SORENSEN (1980), *The Q-superlinear convergence of a collinear scaling algorithm for unconstrained optimization*, SIAM J. Numer. Anal., 17, pp. 88–114.
- (1982), Personal communication.

## EVALUATION OF A BRANCH AND BOUND ALGORITHM FOR CLUSTERING\*

GEORGE DIEHR†

**Abstract.** A branch and bound algorithm for optimal clustering is developed and applied to a variety of test problems. The objective function is minimization of within-group sum-of-squares although the algorithm can be applied to loss functions which meet certain conditions. The algorithm is based on earlier work of Koontz et. al. (1975).

The efficiency of the method for determining optimal solutions is studied as a function of problem size, number of clusters, and underlying degree of separability of the observations. The value of the approach in determining lower bounds is also investigated.

We conclude that the method is practical for problems of up to 100 or so observations if the number of clusters is about 6 or less and the clusters are reasonably well separated. If separation is poor and/or a larger number of clusters are sought, the computing time increases significantly. The approach provides very tight lower bounds early in the enumeration for problems with moderate separation and six or fewer clusters.

**Key words.** cluster analysis, mathematical programming, branch and bound

**1. Introduction.** Cluster analysis is a general statistical methodology for partitioning entities. The goal is often to develop a classification of the entities. Thus, cluster analysis is frequently used in numerical taxonomy. It has also been applied in a diverse variety of other problems including partitioning computer files to minimize access time (Hoffer 1975), clustering data on headaches to identify distinct types of headaches (Diehr 1982) and to problems of aggregation in economics (Fisher 1969). Discussions of clustering applications and alternative clustering criteria can be found in Anderberg (1973), Sokal and Sneath (1973), and MacQueen (1979).

The clustering objective function considered here is minimization of within-group sum-of-squares, WGSS. The entities are observations in a  $p$ -dimensional metric space and the distance between observations is defined as the squared Euclidean distance. The WGSS is equivalently either 1) the sum of distances of observations to their cluster means or 2) the sum over clusters of the average pairwise distance between observations in each cluster. Specifically, let  $Q(i)$ ,  $i = 1, \dots, n$ , be a set of  $n$  vectors (the observations) in  $p$  dimensions. They are to be partitioned into  $m$  clusters,  $G(j)$ ,  $j = 1, \dots, m$ , where  $G(j)$  is a subset containing  $c(j)$  observations, so as to minimize:

$$(1.1) \quad \text{WGSS} = \sum_{j=1}^m \sum_{i \in G(j)} \|Q(i) - \bar{Q}(j)\|,$$

where  $\bar{Q}(j)$  is the vector mean of the  $c(j)$  observations in the  $j$ th cluster.

As noted above, an equivalent expression for WGSS uses distances between observations and therefore does not require computation of cluster means. Define  $d(i, k)$  as the distance between observations  $i$  and  $k$ . Then:

$$(1.2) \quad \text{WGSS} = \sum_{j=1}^m \sum_{i=1}^{n-1} \sum_{k=i+1}^n \frac{d(i, k)}{c(j)}.$$

Using (1.2) allows metrics other than squared Euclidean distance to be used for  $d(i, k)$ . (Note, however, that (1.1) and (1.2) are in general not equivalent for other metrics.) The algorithm developed and tested herein requires that the metric

\* Received by the editors April 21, 1982, and in revised form September 15, 1983.

† Graduate School of Business Administration, University of Washington, Seattle, Washington 98195.

meet the following condition: if a new observation is added to a set of  $n$  observations, the minimum WGSS value for the  $n+1$  observations must be greater than or equal to the minimum WGSS value for original  $n$  observations. To demonstrate that this condition holds for the WGSS criterion consider an optimal partition of  $n+1$  observations. Remove any observation from the set and recompute the loss without reassigning observations to new clusters (i.e. the clustering of the remaining  $n$  observations may be suboptimal). The loss is reduced by more than the squared distance of the removed observation from its cluster mean. Thus, the change in loss if an observation is added must be nonnegative. It may be zero if the added observation is exactly at a cluster mean.

This characteristic of the WGSS criterion was proven by Koontz, Narendra and Fukunaga, "KNF" (1975). They generalized the characteristic to show that if a set of  $n$  observations is first divided into (say) two subsets with  $n_1$  and  $n_2$  ( $n_1+n_2=n$ ) observations each, then the sum of the minimal WGSS for the two subsets is not greater than the minimal WGSS for the  $n$  observations. They use this characteristic as the basis for bounds in a branch and bound algorithm. The algorithm developed and tested here is a refinement of the KNF algorithm.

The KNF algorithm can be applied to several other objective functions. For example, the "complete linkage" criterion which sums distance between all pairs of observations in each cluster clearly meets the condition that an added observation cannot decrease the minimum objective function value. Another criterion which meets the condition is "minimize maximum within cluster distance." Note, however, that an algorithm developed by Hansen and Delattre (1978) appears to be quite efficient for this criterion. Some clustering criteria, such as the "single linkage" criterion, do not meet the condition; addition of another observation may increase or decrease the optimum solution value.

**2. Review of optimal algorithms for clustering.** Many suboptimal algorithms have been developed for the WGSS and related clustering criteria. Most are either hierarchical (they begin with  $n$  clusters, combine two observations to form  $n-1$  clusters, combine two more observations or the first observation pair with another observation to form  $n-2$  clusters,  $\dots$ ), or employ a local hill-climbing approach which reassigns one observation at a time from its cluster to another cluster if the reassignment will reduce the WGSS. A number of these algorithms appear in Hartigan (1975). FORT-RAN code for a refinement of the  $K$ -means algorithm of MacQueen (1967) appears in Hartigan (1975).

Optimal algorithms have been reported using dynamic programming (Jensen 1969), integer linear programming (Rao (1971)), Vinod (1969)), and branch and bound (KNF (1975)). The dynamic programming approach is very efficient in the case of one-dimensional observations. In one dimension the clusters must be a linear partition of the real line so that the dynamic programming approach requires evaluation of Order ( $mn^2$ ) partitions; problems of 1000 observations and 10 clusters can be solved in several minutes. In higher dimensions the dynamic programming approach does not make use of the fact that clusters must be convex partitions of the space. Computation time becomes excessive. For example, with  $n=25$  and  $m=6$ , approximately  $10^{12}$  partitions must be evaluated. Storage requirements exceed 33 million words.

An integer linear programming (ILP) formulation was developed by Vinod (1969) but it was based on a theorem which was subsequently shown to be false by Rao (1971). Rao then provided alternative ILP formulations which require cluster sizes to

be fixed in advance. Even with fixed cluster sizes, the formulation requires  $mn(n+1)/2$  variables and  $2m+n+mn(n-1)/2$  constraints. A small problem of  $n=25$ ,  $m=6$  would require 1950 variables and 1837 constraints. This is a very large ILP problem.

An alternative ILP formulation was developed by Diehr (1980) which requires the same number of variables but only  $mn+m+n$  constraints—181 for this example. This alternative formulation appears to provide better lower bounds from the linear relaxation of the ILP problem which suggests that it would also be faster; however, Rao presents no computational experience for comparison. Our experimentation with the alternative ILP approach using a general purpose LP code (with upper bounding of variables) indicated the approach was unlikely to be practical—a problem of only 9 observations and 2 clusters required 1.7 seconds computation time on an IBM 360/91 to find a bound within 10% of the known optimal (fixed cluster) size solution. Exhaustive enumeration requires evaluation of only 255 partitions (for all possible cluster sizes), which would require much less than one second. (Using the algorithm described herein, problems of 20 observations are easily solved in less than one second.)

Another approach has been explored (Diehr (1980)) which attempts only to determine tight lower bounds to solutions. The problem is formulated as an integer quadratic programming problem, then the integer constraint relaxed. A heuristic search is used to increase the lower bounds. Computational experience with the approach is mixed—for small values of  $m$  (e.g.  $m=2$ ), the bounds determined are reasonably tight (e.g. within 5%). But as  $m$  increases the bound decreases. Nevertheless, the approach merits further investigation of both the mathematics of the quadratic formulation and of the lower bounding algorithm.

The KNF algorithm uses bounds based on the characteristics of the objective function (as opposed to using, for example, linear programming bounds). These bounds, which are used in our algorithm, are described in § 3. KNF report computational experience for one problem of 120 observations in two dimensions partitioned into two clusters. The observations were generated by sampling 60 observations from each of two normal distributions which were reasonably well separated (i.e. one cluster contains observations strictly from the first generating distribution; the second contains observations from the second generating distribution plus two from the first.) The computing time for the clustering is 28 seconds on a CDC 6500.

KNF conclude that this algorithm “... provide(s) the kind of efficiency needed in practice” KNF (1975, p. 914). We would agree with this evaluation if this level of performance (or even an order of magnitude worse performance) held up in general for problems this large. Unfortunately, tests on several problems reported by Diehr (1980) showed that efficiency of the method is very sensitive to the number of clusters and the degree of separation of the generating distributions. Problems of 20 and 30 observations were generated and partitioned into two and four clusters. Time increased by a factor of 20 for four clusters versus two clusters. Furthermore, computation time also significantly increases if the number of clusters is greater than the number of distributions used to generate the observations—i.e., if the observations are not well separated into  $m$  clusters, time to find the optimal partition will be excessive.

The objective of the research reported here was to investigate modifications to the KNF algorithm and to study its computational efficiency as a function of number of observations, number of clusters, and degree of separation of the generating distributions. In practice,  $n$  will be known. The value of  $m$  may be specified or it may be necessary to explore different values of  $m$  to determine the “most natural” clustering. In most cases, the degree of separation of the clusters is not known a priori.

**3. Branch and bound clustering.** In this section we describe two bounds developed by KNF for their clustering algorithm. In § 4 our branch and bound algorithm is described. It uses these same bounds but alters the order of enumeration and employs several heuristics in an attempt to reduce the computational time.

Branch and bound methods operate by computing bounds to completions of “partial solutions.” In the context of a clustering problem, a partial solution is the assignment of a subset of the observations to clusters. The bound is a lower bound on WGSS values for all possible assignments of the remaining observations to clusters (the “completion”) given the partial solution.

To describe the first bound, assume that at some stage in the enumeration a proper subset  $S_1$  of the  $n$  observations has been assigned to the  $m$  clusters. For simplicity, and without loss of generality, define  $S_1$  as the subset of observations numbered  $1, 2, \dots, n_1$ . (We will use the term “subproblem” to mean a subset of observations such as  $S_1$ . This will help to distinguish between subsets of this type, subsets of the cluster type and the generic use of the term subset.) The assignment of observations to clusters is given by the vector  $A(S_1)$  and the loss by  $W[A(S_1)]$ . Define  $S_2$  as the complement of  $S_1$ . The assignment of the observations in  $S_1$  constitutes a partial solution. A lower bound on the WGSS for any completion of this partial solution is:

$$LB1 = W[A(S_1)] + \min_j C[i, j|A(S_1)] \quad \text{for any } i \in S_2,$$

where  $C[i, j|A(S_1)]$  is the increase in the WGSS which results from adding observation  $i$  to cluster  $j$  given the assignment  $A(S_1)$ .

This bound is based on the condition noted in the introduction that adding an observation to a cluster cannot decrease the WGSS of the cluster. The value of  $C[i, j|A(S_1)]$  will be zero iff cluster  $j$  is empty or observation  $i$  coincides with the mean of cluster  $j$ .

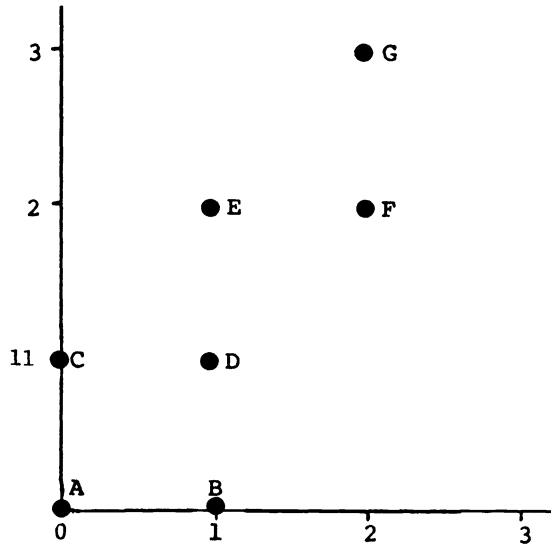
This bound provides the basis for an enumeration algorithm using the following approach:

1. Assume a feasible clustering with objective function value UB (UB is an upper bound on the optimal solution value); UB is set to infinity if no feasible solution is known.
2. Given a partial solution  $A(S_1)$  determine LB1 for any observation  $i$  not in  $S_1$ .
3. If LB1 is greater than or equal to UB then the current partial solution cannot lead to an improved solution (one better than UB). Thus, all partitions with the particular partial assignment  $A(S_1)$  have been implicitly enumerated.

As an illustration of this bound, consider Fig. 1 with seven observations in a two-dimensional space which are to be partitioned into two clusters. Suppose a solution has been found which assigns observations  $A, B, C, D$  to cluster 1; cluster 2 contains observations  $E, F, G$ . The WGSS for this solution is  $(1+1+2+2+1+1)/4 + (1+2+1)/3 = 3.33$ . This is the value of UB (which happens to be optimal).

Now consider the subproblem,  $S_1 = \{A, D, E, G\}$  and the partial solution  $A, D, E$  in cluster 1,  $G$  in cluster 2. The WGSS for this assignment is 2.67. Consider the addition of observation  $B$  to  $S_1$ : The best assignment of  $B$  is to cluster 1 which gives  $LB1 = 3.5$ . Since this exceeds the best known solution value,  $UB = 3.33$ , we know that no completion of the given partial solution can be optimal.

Note that to be efficient the algorithm should avoid multiple enumeration of equivalent solutions. For example, permutation of cluster numbers across clusters does not change the solution. Thus, the assignment of  $G$  to cluster 1, and  $A, D, E$  to cluster 2 has also been implicitly enumerated.



Matrix of squared distances

	A	B	C	D	E	F	G
A	0	1	1	2	5	8	13
B		0	2	1	4	5	10
C			0	1	2	5	8
D				0	1	2	5
E					0	1	2
F						0	1
G							0

FIG. 1. Example clustering problem.

The second bound, LB2, is based on the following characteristic of the WGSS criterion: Given the optimal clustering of the subproblem  $S_2$ , indicated by  $A^*(S_2)$ , a lower bound to the completion of a partial solution  $A(S_1)$  is given by:

$$LB2 = W[A(S_1)] + W[A^*(S_2)].$$

Thus, computing LB2 requires first determining an optimal clustering for  $S_2$ . If  $S_2$  is small (e.g. 10 observations or so) its optimal solution can be computed quickly using LB1.

A variation on LB2 is given by noting that if optimal clusterings are known for  $S_1$  and  $S_2$ , then:

$$W[A^*(S_1 \cup S_2)] \geq W[A^*(S_1)] + W[A^*(S_2)];$$

i.e. the optimal solution value for the union of disjoint subproblems of observations is greater than or equal to the sum of their independent optimal solutions. This fact and LB2 are easily established by noting that the optimal solution for the union of  $S_1$  and  $S_2$  has  $m$  cluster means. Now, consider the observations in  $S_1$  and  $S_2$  separately: without reassignment of cluster memberships compute the  $m$  means for the observations in  $S_1$  and the new loss,  $W(S_1)$ . This must be less than or equal to the WGSS for observations in  $S_1$  using the  $m$  means for the union. The same holds for  $S_2$ . Thus, without reassignment of observations to clusters the value  $W[A(S_1)] + W[A(S_2)]$  must



be less than or equal to  $W[A^*(S_1 \cup S_2)]$ . Clearly, with reassignment of observations in  $S_1$  and  $S_2$  to obtain optimal partitions,  $W[A^*(S_1)]$  and  $W[A^*(S_2)]$ , will each be less than or equal to  $W[A(S_1)]$  and  $W[A(S_2)]$  respectively.

This result is easily generalized to  $g$  subproblems,  $S_1, S_2, \dots, S_g$ , which partition the  $n$  observations, i.e.,

$$W\left[A^*\left(\bigcup_{f=1,2,\dots,g} S_f\right)\right] \cong \sum_{f=1}^g W[A^*(S_f)].$$

Therefore, one approach to a lower bound is to partition the observations into  $g$  subproblems and find the optimal clustering for each. The sum of these subproblem solutions gives a lower bound on the optimal WGSS for the complete problem. If each subproblem is small determining its optimal partition is fast—however, small subproblems give poor bounds (e.g. if a subproblem has  $m$  or fewer observations it has minimum WGSS of zero).

As an illustration of this bound, consider the same seven observations with subproblems  $S_1 = \{B, C, F\}$  and  $S_2 = \{A, E, D, G\}$ . The optimal partition of  $S_1$  into two clusters places  $B$  and  $C$  in one cluster and  $F$  in the other giving  $W[A^*(S_1)] = 1$ . The optimal clustering for  $S_2$  has clusters  $A, D$  and  $E, G$  giving  $W[A^*(S_2)] = 2$ . The lower bound for the union is 3.

To illustrate the use of LB2 for partial solutions, assume that there are  $g$  subproblems each containing observations as follows:

$$\begin{aligned} S_1 &= 1, 2, \dots, n_1, \\ S_2 &= n_1 + 1, n_1 + 2, \dots, n_2 + n_1, \\ &\dots \\ S_g &= n_1 + n_2 + \dots + n_{g-1} + 1, \dots, n. \end{aligned}$$

The first bound, LB1, is now used in an enumeration to find  $W[A^*(S_f)]$ ,  $f = 1, 2, \dots, g$ . If the enumeration for  $S_1$  maintains a fixed order of the observations, then optima are found for “suborders” of the subproblem  $S_1$ :  $W[A^*\{1\}]$ ,  $W[A^*\{1, 2\}]$ ,  $\dots$ ,  $W[A^*\{1, 2, \dots, n_1 - 1\}]$ ,  $W[A^*(S_1)]$ . Similar optima are found for suborders of the other subproblems. To define the computation of the bound using these optima we first introduce a more compact notation: Let  $W^*[u, v]$ ,  $u \leq v$ , be the solution value from the optimal clustering of observations  $u, u + 1, \dots, v$ . The lower bound for a partial solution  $A(r, r + 1, \dots, n)$  is given by:

$$\begin{aligned} \text{LB2} &= W[A(r, r + 1, \dots, n)] + W^*[1, 2, \dots, r - 1] \quad \text{if } r - 1 \leq n_1 \\ &= W^*[S_1] + W^*[n_1 + 1, \dots, r - 1] \quad \text{if } n_1 < r - 1 \leq n_2 + n_1 \\ &\dots \\ &= W^*[S_1] + W^*[S_2] + \dots + W^*[n_1 + n_2 + \dots + n_{g-1} + 1, \dots, r - 1] \\ &\hspace{15em} \text{if } (n_1 + n_2 + \dots + n_{g-1}) < r \leq n. \end{aligned}$$

Thus, if a partial solution is a subset of the observations numbered from  $r$  to  $n$ , the information contained in finding solutions for the subproblems  $S_1, S_2, \dots, S_g$  can be used to compute the bound.

To illustrate computation of a bound for a partial solution based on optimal solutions for subproblems, assume subproblems  $S_1 = \{B, F, C\}$  and  $S_2 = \{A, E, D, G\}$  with the observations ordered as indicated. Optimal solution for  $S_1, S_2$ , and their

suborders are:

$$\begin{aligned} W^*[B] &= 0, \\ W^*[B, F] &= 0, \\ W^*[B, F, C] &= 1, \\ W^*[A] &= 0, \\ W^*[A, E] &= 0, \\ W^*[A, E, D] &= 0.5, \\ W^*[A, E, D, G] &= 2. \end{aligned}$$

Now consider the partial solution which assigns observations  $D$  and  $G$  to the same cluster. The bound is:

$$W[D, G] + W^*[A, E] + W^*[B, F, C] = 2.5 + 0 + 1 = 3.5.$$

Since this exceeds the current upperbound, 3.33, no optimal solution can have observations  $D$  and  $G$  in the same cluster.

**4. Branch and bound algorithm.** In this section we first describe how the lower bounds are used by the enumeration algorithm. Then the “solution system” is outlined including discussion of heuristic methods used to obtain an initial feasible solution and the heuristic used for generation of subproblems.

**4.1. The basic algorithm.** The algorithm uses a fixed ordering of observations in each of a user specified number,  $g$ , of subproblems. Optimal solutions are found for each subproblem and its suborders.

The algorithm is a modification of the algorithm of KNF and is detailed in Fig. 2. Several definitions are needed for that figure:  $LB(u, v)$  is one of the lower bounds,  $LB1$  or  $LB2$ , for the partial solution of observations  $u, u+1, \dots, v$ . As described below, when the initial subproblems are optimized,  $LB1$  is used.  $LB2$  is used on the next phase of the algorithms which finds optimal clusters for unions of subproblems.  $PHI$  is a two-dimensional array with a row for each observation and a column for each cluster used for bookkeeping in the enumeration. If  $PHI(i, j) = 0$  then observation  $i$  has not yet been assigned to cluster  $j$  for the current partial solution for observations  $1, 2, \dots, i-1$ . If  $PHI(i, j) = 1$  then it has been assigned to  $j$ .  $ASAVE$  is a vector of assignments for the best known solution. The value of  $N$  is the number of observations in the particular subproblem which is being clustered. Thus, the algorithm in Fig. 2 is for one subproblem at a time. Observations are numbered from 1 to  $N$  for the subproblem for simplicity.

Two significant changes have been made to the KNF algorithm:

1. In a forward branch (Step 2A) with several as yet empty clusters the observation is assigned to the empty cluster of lowest number and the array  $PHI$  updated to indicate that assignments of this observation to all other empty clusters has been implicitly enumerated. This avoids multiple enumeration of solutions which are equivalent except for permutation of cluster numbers. Similarly, at Step 1, observation 1 is permanently assigned to the cluster determined by the heuristic solution.

2. The algorithm begins with a heuristic solution, Step 0, provided as part of the overall clustering *solution system* (described below).

After optimal solutions are found for the  $g$  subproblems, they are paired creating new subproblems with the order of observations reversed in the enumeration. To

Step 0: Apply some heuristic solution method to obtain a starting solution. The initial solution has value  $B$  and assignments  $A(I)$ ,  $I = 1, N$ .  
Set  $C(J)$  equal to the number of observations in cluster  $J$ .

Step 1: Initialize PHI to record heuristic solution.  
Save heuristic assignment in ASAVE.

```
FOR I = 1, N
  ASAVE(I) = A(I)
  FOR J = 1, M
    PHI(I, J) = 0
  NEXT J
NEXT I
FOR I = 1, N
  PHI(I, A(I)) = 1
NEXT I
```

(One observation can be fixed in one cluster. Permanently assign observation 1 to cluster  $A(1)$  to avoid multiple enumeration)

```
FOR J = 1, M
  PHI(1, J) = 1
NEXT J
```

$K = N$  (Will start enumeration with observation  $N$ , trying it in alternative clusters)

Step 2A: See if there is an empty cluster in which observation  $K$  has not been tried. If so assign it there and update PHI so that it will not be tried in any other empty clusters.

```
FOR J = 1, M
  IF C(J) = 0 AND PHI(K, J) = 0
    THEN A(K) = J
        C(J) = C(J) + 1
        FOR J1 = J, M
          IF C(J1) = 0 THEN PHI(K, J1) = 1
        NEXT J1
    GOTO STEP 3
NEXT J
```

(Failed to find an empty cluster in which  $K$  had not already been tried. Continue with step 2B)

Step 2B: Assign  $K$  to first cluster it has not been tried in. If none exist then backtrack at step 5.

```
FOR J = 1, M
  IF PHI(K, J) = 0 THEN
    A(K) = J
    C(J) = C(J) + 1
    PHI(K, J) = 1
    GOTO STEP 3
NEXT J
GOTO STEP 5
```

Step 3: See if lower bound to partial solution is less than upper bound; if not, back obs.  $K$  out of cluster  $J$  and return to Step 2A to try another cluster.

```
IF LB(1, K) < B THEN GOTO STEP 4
ELSE C(J) = C(J) - 1
    A(K) = 0
    GOTO STEP 2
```

Step 4: Lower bound is less than upper bound. If  $K = N$  then lower bound is exact value of current assignment and represents an improved solution.

```
IF K = N THEN B = LB(1, N)
    ASAVE = A
    (will continue at Step 5)
ELSE K = K + 1
    GOTO STEP 2
```

FIG. 2. Branch and bound algorithm.

Step 5: Back-track. Clear PHI for obs.  $K$ . Reduce  $C(J)$  by 1.  
 Clear  $A(K)$ . Reduce  $K$  by 1.  
 $C(J) = C(J) - 1$   
 FOR  $J = 1, M$   
     PHI( $K, J$ ) = 0  
 NEXT  $J$   
 $A(K) = 0$   
 $K = K - 1$   
 (Continue with Step 6)

Step 6: If  $K = 0$  then all done. Otherwise, if a possible assignment exists for  $K$  make it. Else, back-track one more observation.  
 IF  $K = 0$  THEN GOTO STEP 7 (All done)  
 FOR  $J = 1, M$   
     IF PHI( $K, J$ ) = 0 THEN GOTO STEP 2  
 NEXT  $J$   
 (No assignments remain for  $K$ —back-track)  
 GOTO STEP 5

Step 7: All done.  
 Optimal solution value is  $B$ . Optimal assignment is ASAVE.

FIG. 2—continued

illustrate the pairing and reversal of enumeration order refer to Fig. 3. Twenty observations are to be clustered and five initial subproblems are created. After optimal solutions are found for subproblems  $S_1$  through  $S_5$ ,  $S_1$  and  $S_2$  are combined creating  $S_{12}$  and solved.  $S_3$  and  $S_4$  are next combined and solved. In each pairing the order of enumeration is reversed to take advantage of the optimal solution values known for all suborders. Next,  $S_{12}$  and  $S_{34}$  are paired and solved. Finally,  $S_{1234}$  is paired with  $S_5$  (creating the full problem) and solved.

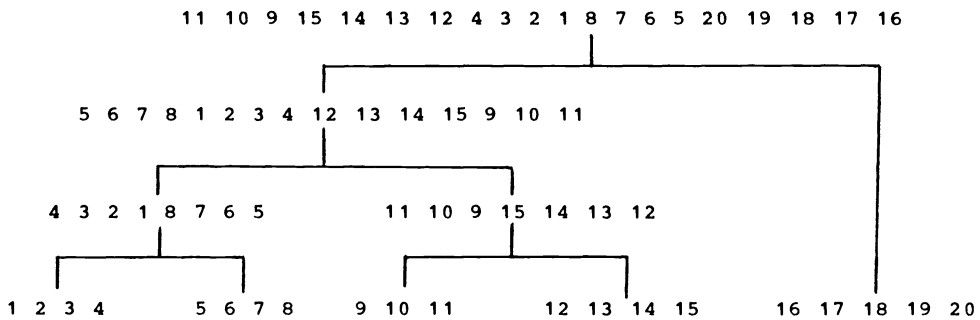


FIG. 3. Example partitions of observations into five subproblems and pairing of subproblems. Observation numbers are shown in order of enumeration.

**4.2. The solution system.** The time to find and verify the optimal solution is critically dependent on the quality of the initial subproblems. The “quality” is the nearness of the sum of minimal WGSS over subproblems to the minimal WGSS for the full problem. We have found that total solution time is significantly reduced if approximate solution methods are used to obtain locally optimal solutions to both the full problem and to the subproblems. A heuristic is also employed to generate subproblems which are in a sense “representative” of the full problem as opposed to using a random partition into subproblems. The solution system has the following steps:

1. Find an upper bound using the following approximate method on the full problem: Begin with a random assignment of observations to clusters. Each observation is then moved from its cluster to that cluster which results in the greatest reduction in WGSS. (The means of both the current cluster and the trial cluster are updated to determine the total change in WGSS of the reassignment.) This is done for each observation until no reduction in WGSS is possible (using this local search). The algorithm is performed five times then repeated until the two best solutions are within one percent of each other.

2. The observations in each cluster are next partitioned into  $g$  "cluster subproblems." The partitioning algorithm is identical to the clustering algorithm described in Step 1 except that the objective is to maximize WGSS.

3. Subproblems are constructed by selecting one cluster-subproblem from each cluster. Each cluster-subproblem is assigned to one subproblem. Figure 4 is a Venn diagram which illustrates clusters, subproblems and cluster subproblems.

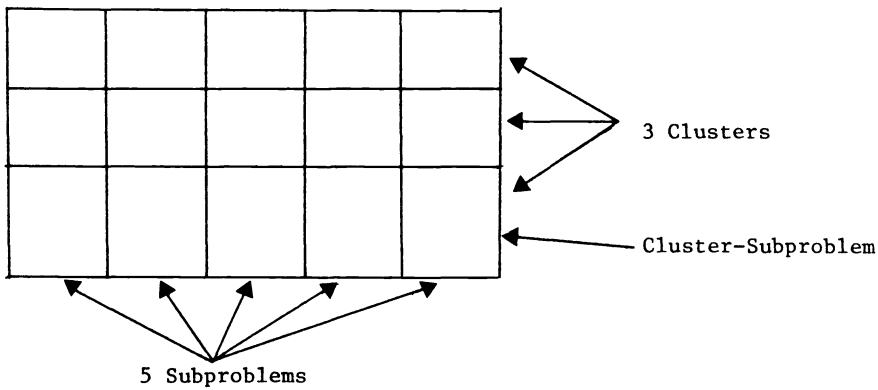


FIG. 4. Venn diagram describing terms used for various subsets of observations.

4. Each subproblem is clustered using the approximate algorithm described in Step 1 (i.e., random assignment and reassignment). This is done prior to optimization to increase the likelihood that the branch and bound algorithm starts with an optimal solution for the subproblem.

5. Each subproblem is solved optimally using the *B-B* algorithm and the lower bound LB1. Optimal solutions for suborders of the subproblems are saved for use in computing bounds for paired subproblems.

6. Subproblems are paired to create new subproblems.

7. A local optimum is found for each new subproblem using Step 1.

8. Each new subproblem is solved using the *B-B* algorithm. Lower bounds, LB2, are used during the enumeration.

9. If the problem solved at Step 8 was the full problem we are done. Otherwise, the process continues at Step 6.

**5. Computational experience.**

**5.1. Problem generation.** Problems of various sizes and generating distributions were created and solved for various numbers of clusters to investigate the performance of the algorithm as a function of these factors. We also studied the impact of variation in number of subproblems,  $g$ , the order of enumeration, and quality of the initial subproblems.

The problems were created by randomly selecting observations from one of the following five generating distributions:

Type D1. A single spherical normal distribution in two dimensions.

Type D2. A pair of spherical normal distributions in two dimensions with means (100, 100) and (130, 130). Standard deviations were 10 for both distributions in both dimensions. An equal number of observations were sampled from each.

Type D4. Four spherical normal distributions in two dimensions with means (100, 100), (100, 150), (150, 100), and (150, 150); all standard deviations were 10. The number of observations sampled from each distribution was the same or within one (e.g. for a 30 observation problem either seven or eight observations were sampled from each).

Type D2K. The distribution used by KNF: Two normal distributions with means (100, 100) and (150, 110); standard deviations of (7.1, 22.4), and (17.2, 7.1). Covariances were zero but distributions were not spherical. An equal number of observations was generated from each.

Type D8. Eight spherical normal distributions in three dimensions. The eight possible distributions using means 100 and 160 with standard deviations 10 were generated. Eight observations were sampled from each distribution.

**5.2. Solution times.** Table 1 gives solution times as a function of number of observations, number of clusters, and degree of separation of the generating distributions. The timings in this table are the best results from tests using various numbers of subsets. Times are CPU seconds on an IBM 3033 using the WATFIV compiler and include problem generation, heuristic solutions, and optimization.

Figure 5 presents this same information as two superimposed graphs of solution time versus number of observations. One graph has lower and left axes for observations and time, the other uses top and right axes. The following notation is used: circles are

TABLE 1  
*Timing summary.*

Observations	Clusters	Type	Time CPU sec	Notes
20	2	D2	0.15	
20	4	D4	0.4	
20	2	D1	0.34	
24	2	D1	1.1	
24	4	D1	3.7	
30	2	D2	0.6	
30	4	D4	1.2	
30	2	D1	3.5	
30	4	D1	19.5	
40	2	D2	0.8	
40	4	D4	1.9	
40	4	D1	>60	(bound 95.4%)
50	2	D2	1.4	
50	4	D4	3.4	
60	2	D2	1.9	
60	2	D2K	2.6	
60	4	D4	3.1	
64	8	D8	>60	(bound 78%)
120	2	D2K	9.95	
120	4	D4	21.0	

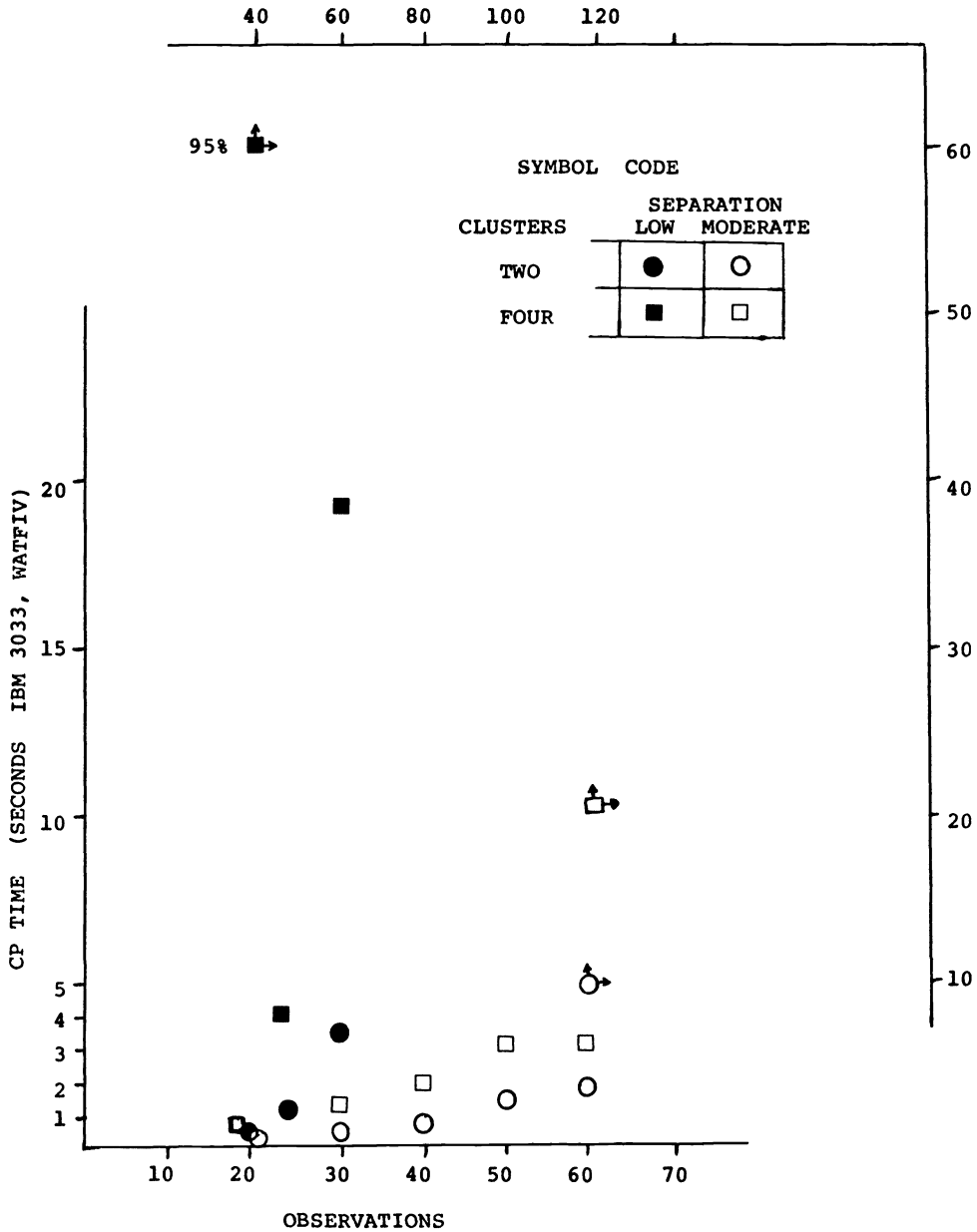


FIG. 5. Solution time versus problem size.

used for problems of two clusters, squares for four clusters; solid symbols represent problems generated with low separation, open symbols represent problems with moderate separation; the appropriate axes are the lower and left except for the several symbols pointing to the right and up. The notation "95%" by one symbol indicates that this is the best known lower bound to the best known approximate solution. (The problem was not solved in a time limit of 60 CPU seconds.)

Figure 5 makes it clear that problems of moderate separation (Types D2, D2K, D4) are reasonably easy to solve. Time increase (for fixed number of clusters) is only slightly greater than linear in number of observations. It seems safe to conclude that

such problems are indeed “easy” and the algorithm provides a practical approach to their solution.

Problems with low separation are another matter. Problems of up to 30 observations and two or four groups were all solved in less than 20 seconds but the time required is in the range of an order of magnitude greater than time required for comparable, moderately separated, problems. For example, problem Type D1 with 30 observations and four clusters required 19.5 seconds; problem D4 with 30 observations and four clusters required less than two seconds. One problem of 40 observations and four clusters was not solved in 60 CPU seconds, although its lower bound was within 5% of the best known heuristic solution.

One experiment with larger number of clusters and higher dimensions of the underlying distributions suggests that the algorithm performance significantly deteriorates on such problems. Problem Type D8, 64 observations, eight clusters, eight underlying distributions in three dimensions was not solved in 60 seconds. Furthermore, bounds obtained on subproblems were only within 22% of the optimum.

**5.3. Quality of lower bounds.** In many clustering problems the lower bounds obtained from clustering subproblems may be close enough to the heuristic solution to consider the full problem solved. Table 2 shows lower bounds for our problems at various levels of combination of subproblems. In the column “Subproblems,” 1, 2, 3, 4 indicate the four separate subproblems  $S_1$ ,  $S_2$ ,  $S_3$  and  $S_4$ . An entry (1, 2), (3, 4) indicates two subproblems formed from the union of  $S_1$  and  $S_2$  and the union of  $S_3$  and  $S_4$ . The column “Lower bound” is the sum of subproblem optimal solutions as a percentage of the heuristic solution.

The results suggest that if separation is at least moderate and the number of clusters is four or less the bounds are reasonably good to excellent. As examples: problem D4 with  $N=60$  and  $M=4$  has a lower bound for  $g=4$  (the four initial subproblems) of 95.4%; after combination into two subproblems the bound is 99.94%. Problem Type D4 with 120 observations into four clusters has bounds of 93.6%, 96.9% and 99.4% for eight, four, and two subproblems respectively. For problems of low separation and for greater number of clusters the bounds are poor to fair. For example, a problem from D1 with  $N=30$ ,  $M=4$  had bounds of 57.4% and 89.9% at four and two subproblems respectively. The moderately separated problem D8 provided a bound of 78% for four subproblems. Excessive computer time discouraged further experimentation with this problem.

**5.4. Selection of number of subproblems.** The timings in Table 1 and Fig. 5 are the minimum CPU times for various numbers of initial subproblems. Table 3 shows the sensitivity of time to number of subproblems. In almost all instances, it pays to start with a “large” number of subproblems—a number such that as few as two or three observations will be in each cluster formed for the initial subproblems. For example, the problem of 60 observations and four clusters was divided into 6 initial subproblems. With an average of 10 observations per subproblem and four clusters, each cluster has an average of only 2.5 observations. Note, however, that the lower bound is a surprising 96.2% of the optimum. The solution time is only 3.1 seconds versus four seconds using four subproblems.

The most extensive experimentation with number of subproblems was with the 50 observation problem, Type D4, partitioned into four clusters. Times for 2, 4, 5, and 6 subproblems were over 10, 6.4, 6.5, and 3.4 seconds respectively. We did not always do better with large numbers of subproblems. The difficult problem, D1 with 40 observations and four clusters was tried with both five and six subproblems. With



TABLE 2  
*Lower bounds.*

Problems with moderate separation: D2, D2K, D4, D8.			
Observations	Clusters	Subproblems	Bounds
30	4	1, 2	96%
40	2	1, 2, 3, 4 (1, 2)(3, 4)	98.6 99.4
40	4	1, 2, 3, 4 (1, 2)(3, 4)	90.3 98
50	2	1, 2, 3, 4 (1, 2)(3, 4)	99.2 99.96
50	4	1, 2, ..., 6 (1, 2)(3, 4)(5, 6) (1-4)(5, 6)	85 95.5 97.4
60	2	1, 2, ..., 6 (1, 2)(3, 4)(5, 6) (1-4)(5, 6)	96.1 99.1 99.6
120	2	1, 2, ..., 8 (1, 2) ... (7, 8) (1-4)(5-8)	98.8 99.9 99.98
120	4	1, 2, ..., 8 (1, 2) ... (7, 8) (1-4)(5-8)	93.6 96.9 99.4
Problems with low separation: D1			
24	2	1, 2, 3, 4 (1, 2)(3, 4)	90 96.8
24	4	1, 2, 3, 4 (1, 2)(3, 4)	42 84
30	2	1, 2, 3, 4 (1, 2)(3, 4)	83 98.1
30	4	1, 2, 3, 4 (1, 2)(3, 4)	57.4 89.9
40	4	1, 2, 3, 4, 5 (1, 2)(3, 4), 5 (1-4), 5	61 88 95.4

five subproblems a bound of 95.4% was achieved in 60 seconds; with six subproblems the bound was only 69% in 60 seconds.

In most cases the cost of using too many subproblems will be a small amount of wasted time solving small subproblems.

**5.5. Importance of “quality” subproblems.** Our solution system relies heavily on heuristic methods—both to find good clusters and to generate representative subproblems. Several problems were run using random generation of subproblems to determine the impact on solution time and lower bounds. Results are also compared to KNF who use random subproblem generation. Table 4 summarizes these results showing both solution times and bounds for various subproblems. As an example of the value

TABLE 3  
Effect of number of subproblems.

Problems with moderate separation: D2, D2K, D4			
Observations	Clusters	No. of subproblems	Time
30	4	1	3.2
		2	1.2
40	4	2	6.0
		3	3.1
		4	1.9
50	4	2	>10
		4	6.4
		5	6.5
		6	3.4
60	4	4	4.0
		6	3.1
Problems with low separation: D1			
20	2	1	2.1
		2	0.34
24	2	1	>10
		2	1.1
		4	1.1
30	2	2	>10
		4	3.5

TABLE 4  
Comparison of guided vs. random subproblem generator.

Observations	Clusters	Type	Subproblems	Generation method			
				Random		Guided	
				bound	time	bound	time
30	2	D1	1, 2, 3, 4 (1, 2)(3, 4)	70%		83%	
				85		98.1	
30	4	D1	1, 2, 3, 4 (1, 2)(3, 4)	37		57	
				77		90	
					54		3.5
40	4	D4	1, 2, 3 (1, 2), 3	65		91	
				96.8		98.8	
50	4	D4	1, 2, 3, 4 (1, 2)(3, 4)		>60		19.5
					5.0		
				74		88	
120	2	D2K	1, 2, ..., 8 (1, 2) ... (7, 8) (1-4)(5-8)	83.4		98.9	
				96.2		99.8	
				99.4		99.96	
					*		10

\* Bound results are from Koontz, Narendra, Fukunaga (1975). They report 28 seconds on a CDC 6500. Due to differences in machine power (IBM 3033 faster than the CDC 6500), efficiency of code (the WATFIV compiler used on the IBM is not highly optimized), it is difficult to make any comparisons.

of quality subproblem generation, the bounds reported by Koontz are 83.4%, 96.2% and 99.4% for eight, four and two subproblems respectively. These may be compared to our bounds of 98.9%, 99.8% and 99.96% for similar number of subproblems. Thus, generation of quality subproblems might allow “verification” of a heuristic solution with the initial subproblems; random subproblems will probably require combination before bounds are sufficiently tight.

Solution time using nonrandom subproblem selection was always lower in our experiments than with random generation. Note that the time for either type of selection includes the subproblem generation time. In several cases the time was dramatically lower—in both cases these were problems of low separation. Times for 30 observations from D1 into two and four clusters was 54 seconds and over 60 seconds respectively using random generation; using heuristic subproblem generation the comparable times were 3.5 and 19.5 seconds.

These results suggest that in practice one should limit CPU time to a moderate amount and review the subproblem solution before proceeding to optimizations of combinations. Long solution times and poor bounds when the number of clusters is six or less suggest that the observations are not “naturally” clustered or possibly not well separated into the selected number of clusters. Prior use of heuristic algorithms should help to narrow the determination of the correct number of clusters in a data analysis problem. Experiments with one algorithm reported by Hartigan and Wong (1979) also indicates that solution times were faster for problems with well separated data.

**6. Conclusion.** The branch and bound algorithm presented here is a practical approach to optimal clustering if the following conditions hold:

1. The clusters are reasonably well separated. An indication of their separation is given by the difference in value of heuristic subproblem solution and heuristic full problem solution.
2. The number of clusters sought is limited to six or so. The limit interacts with degree of separation and problem size. In some situations problems of as few as two clusters may be very time consuming.
3. Problem size is not much over 120 observations.

These results make it clear that the algorithm has limitations. In situations where the algorithm performs well, heuristic methods are almost certain to find the optimal. In problems where heuristic methods are apt to miss the optimal, the *B-B* algorithm is slow. The increased time with number of clusters is a significant deficiency.

The problem size limitation is not, in itself, too severe a handicap. With good separation and few clusters the bounds are tight enough from solution of subproblems that in practice one would probably never need to find the global optimum in a problem of (say) 1000 observations. By careful subproblem generation the sum of optimal solutions from 10 subproblems of 100 observations each would be expected to yield a bound easily within one percent of the optimal.

**7. Suggestions for further research.** It is highly likely that solution time can be improved by utilizing better bounding techniques, particularly by improving on the LB1 bound used for the initial subproblems. Candidates for lower bounds include solutions based on *M*-medians of a graph, spanning trees, nearest neighbors, and the quadratic programming formulation.

**Acknowledgments.** I wish to thank Hugo Moortgat for many hours of helpful suggestions. My appreciation also to a referee for many helpful suggestions.

## REFERENCES

- M. R. ANDERBERG (1973), *Cluster Analysis for Applications*, Academic Press, New York.
- GEORGE DIEHR AND HUGO MOORTGAT (1980), *Mathematical programming in cluster analysis*, Proc. 12th Annual Meeting American Institute for Decision Sciences, pp. 226–228.
- PAULA DIEHR, GEORGE DIEHR, THOMAS KOEPEL, ROBERT WOOD, KIRK BEACH, BARRY WOLCOTT AND RICHARD K. TOMPKINS (1982), *Cluster analysis to determine headache types*, J. Chronic Disease, 35, pp. 623–633.
- W. D. FISHER (1969), *Clustering and Aggregation in Economics*, Johns Hopkins Press, Baltimore.
- , (1958), *On grouping for maximum homogeneity*, J. Amer. Statist. Assoc., 53, pp. 789–798.
- PIERRE HANSEN AND MICHEL DELATTRE (1978), *Complete-link cluster analysis by graph coloring*, J. Amer. Statist. Assoc., 73, pp. 397–403.
- J. A. HARTIGAN AND M. A. WONG (1979), *A K-means clustering algorithm*, Applied Statistics, 28, pp. 100–108.
- J. A. HARTIGAN (1975), *Clustering Algorithms*, John Wiley, New York.
- ROBERT E. JENSEN (1969), *A dynamic programming algorithm for cluster analysis*, Oper. Res., 17, pp. 1034–1057.
- WARREN L. G. KOONTZ, PATRENAHALLI M. NARENDRA AND KEINOSUKE FUKUNAGA (1975), *A branch and bound clustering algorithm*, IEEE Trans. Comput., C-24, pp. 908–915.
- E. L. LAWLER AND D. E. WOOD (1966), *Branch-and-bound methods: A survey*, Oper. Res., 14, pp. 699–719.
- JAMES B. MACQUEEN (1967), *Some methods of classification and analysis of multivariate observations*, Proc. 5th Berkeley Symposium on Mathematical Statistics and Probability, Univ. California Press, Berkeley, pp. 280–298.
- , (1979), *Notes on practical and mathematical experience with clustering*, Western Management Science Institute, Working Paper 295, Univ. California, Los Angeles.
- M. R. RAO (1971), *Cluster analysis and mathematical programming*, J. Amer. Statist. Assoc., 66, pp. 622–626.
- PETER H. A. SNEATH AND ROBERT R. SOKAL (1973), *Numerical Taxonomy*, W. H. Freeman, San Francisco.
- H. D. VINOD (1969), *Integer programming and the theory of grouping*, J. Amer. Statist. Assoc., 64, pp. 506–519.

## RADIATION OF SOUND FROM UNFLANGED CYLINDRICAL DUCTS\*

S. I. HARIHARAN† AND ALVIN BAYLISS‡

**Abstract.** In this paper we present calculations of sound radiated from unflanged cylindrical ducts. The numerical simulation models the problem of an aero-engine inlet. The time-dependent linearized Euler equations are solved from a state of rest until a time harmonic solution is attained. A fourth-order accurate finite difference scheme is used. Solutions are obtained from a fully vectorized Cyber-203 computer program. Cases of both plane waves and spin modes are treated. Spin modes model the sound generated by a turbofan engine. Boundary conditions for both plane waves and spin modes are treated. Solutions obtained are compared with experiments conducted at NASA Langley Research Center.

**Key words.** duct acoustics, sound radiation, finite difference, numerical boundary conditions

**1. Introduction.** In this paper we present a computational method to study sound radiated from an unflanged cylindrical duct. An incident field which is either a plane wave or a spinning mode (i.e., dependence on the azimuthal angle) propagates down the duct. At the open end of the duct, sound is radiated out into the farfield and a reflected wave traveling upstream in the duct is generated. This problem is of importance in the study of noise radiated from aero-engine inlets and in the development of effective duct liners.

A significant amount of work has been done on the computation of sound propagation in an infinitely long duct. A survey of such work may be found in [1]. The open end of the duct and the ensuing outward radiation of energy significantly complicates the problem. A further complication is the presence of the inlet flow about which little is known experimentally. In the procedure adapted here the solution is obtained by solving the Euler equations linearized about an arbitrary mean flow. Thus the method is general enough to permit computation of the linearized fluctuating field about an experimentally determined mean flow. In this paper, however, only numerical results for the case of no mean flow will be presented.

We will briefly discuss some work which has been done in the past and is relevant to our work. The earliest work in calculating the sound wave (pressure) radiated from cylindrical ducts is due to Levine and Schwinger [8]. They provided a method to predict sound from a semi-infinite thin pipe, when a plane wave is incident upstream in the pipe, using the Wiener-Hopf technique. This work motivated several other researchers in this field, in particular Savkar [10] provided a method to predict sound using Wiener-Hopf techniques for the case of an incident spinning mode. Ting and Keller [13] developed an asymptotic expansion valid for plane wave incidence and low frequencies. For higher frequencies asymptotic methods have not been successfully applied to this problem because there are different length scales inside the pipe and in the farfield. Though these methods provide some means to compute the sound radiated from engine inlets, they do not correspond to the entire physical situation

---

\* Received by the editors August 8, 1983, and in revised form November 14, 1983. This research was supported by the National Aeronautics and Space Administration under contract 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. Currently with University of Tennessee Space Institute, Tullahoma, Tennessee 37388.

‡ Exxon Corporate Research, Annandale, New Jersey 08801. The work of this author was also supported by the U.S. Department of Energy, under contract TE/AC02/76ER03077 and by the U.S. Air Force under contract AFOSR/81/0020.

due to the thickness of the inlets. With a given thickness of the duct and for smooth geometries, calculations are effectively handled by integral equation methods. One such work is due to Horowitz et al. [5] and includes the effect of a mean flow. This work is based on combining an exterior integral equation with a finite element discretization in the vicinity of the flow. A comparison of numerical and experimental results is given in [5].

**2. Formulation of the problem.** The equations for the fluctuating acoustic field will be obtained by linearizing the Euler equations for inviscid fluid motion.

$$(2.1) \quad \frac{\partial \rho}{\partial t} + \operatorname{div}(\rho \mathbf{v}) = 0, \quad \rho \left( \frac{\partial v_i}{\partial t} + v_j \frac{\partial v_i}{\partial x_j} \right) + \frac{\partial p}{\partial x_i} = 0.$$

For simplicity the equations are derived in Cartesian coordinates although the relevant geometry requires cylindrical coordinates. Here  $\rho$  is the density,  $\mathbf{v}$  the velocity, and  $p$  is the pressure. The flow variables are divided into a mean (denoted by a bar) and fluctuating field (denoted by a prime)

$$(2.2) \quad \rho = \bar{\rho} + \rho', \quad \mathbf{v} = \bar{\mathbf{U}} + \mathbf{u}', \quad p = \bar{p} + p',$$

and only first-order terms in the fluctuating field are retained. The fluctuating density  $\rho'$  is replaced by the fluctuating pressure  $p'$  by assuming that the flow is homentropic and has no mean temperature gradient so that

$$(2.3) \quad p = A\rho^\gamma,$$

or

$$(2.4) \quad \rho' = \frac{p'}{c_0^2} + O(p'^2),$$

where  $c_0$  is the ambient speed of sound. The resulting linearized system from (2.1) becomes

$$(2.5) \quad \frac{1}{c_0^2} \frac{\partial p'}{\partial t} + \frac{1}{c_0^2} \operatorname{div}(p' \bar{\mathbf{U}}) + \operatorname{div}(\bar{\rho} \mathbf{u}') = -\operatorname{div}(\bar{\rho} \bar{\mathbf{U}}),$$

$$\bar{\rho} \left( \frac{\partial u'_i}{\partial t} + U_j \frac{\partial u'_j}{\partial x_j} + u'_j \frac{\partial \bar{U}_i}{\partial x_j} \right) + \frac{p'}{c_0^2} \bar{U}_j \frac{\partial \bar{U}_i}{\partial x_j} + \frac{\partial p'}{\partial x_i} = -\frac{\partial \bar{p}}{\partial x_i} - \bar{\rho} \bar{U}_j \frac{\partial \bar{U}_i}{\partial x_j}.$$

The acoustic field thus satisfies a linear first-order hyperbolic system.

The numerical results presented here are for the case of no mean flow ( $U_j = 0$ ). However it is apparent from (2.5) that this does not cause major difficulties. In this case (2.5) reduces to

$$(2.6) \quad \frac{1}{c_0^2} \frac{\partial p'}{\partial t} + \rho_0 \operatorname{div} \mathbf{u}' = 0, \quad \rho_0 \frac{\partial \mathbf{u}'}{\partial t} + \nabla p' = 0,$$

where  $\rho_0$  is the density of the ambient fluid. We nondimensionalize these equations. Length is nondimensionalized by the diameter of the pipe ( $d$ ), time by  $c_0/d$ , pressure by  $\rho_0 c_0^2$  and the velocity by  $c_0$  to obtain

$$(2.7) \quad \frac{\partial p}{\partial t} + \operatorname{div} \mathbf{u} = 0, \quad \frac{\partial \mathbf{u}}{\partial t} + \nabla p = 0.$$

Note that in (2.7) the primes on the fluctuating quantities are dropped for convenience.

*Remark 2.1.* If  $p$  and  $u$  are time harmonic, that is

$$p(x, t) = \hat{p}(x) e^{-ikt}, \quad u(x, t) = \hat{u}(x) e^{-ikt},$$

where  $k$  is the wave number, then the system (2.7) reduces to

$$(2.8) \quad \Delta \hat{p} + k^2 \hat{p} = 0.$$

The problem discussed here is to solve the system (2.7) for  $p$  and  $u$  subject to appropriate boundary conditions which will be discussed later.

The technique here is to drive the system (2.7) with a time harmonic source which will yield the time harmonic solutions, namely the solution of (2.8). This technique is essentially the numerical implementation of an appropriate limiting amplitude principle. For exterior problems this method has been demonstrated by Kriegsmann and Morawetz [6] and Taflove and Umashankar [12] and for wave guide problems by Baumeister [1] and Kriegsmann [7]. For the case of no mean flow it will be seen that this is a computationally efficient method to obtain accurate results. The case of a nonzero mean flow can be handled analogously although the authors are not aware of theoretical results relating the mean flow to the decay rate of the transient.

**3. Solution procedures.** We take the origin at the open end of the duct so that its generators are parallel to the  $z$  axis (Fig. 1). We look for solutions of (2.7) of the form

$$(3.1) \quad \begin{aligned} p(r, \theta, z, t) &= P(r, z, t) e^{im\theta}, \\ u(r, \theta, z, t) &= \underline{U}(r, z, t) e^{im\theta}. \end{aligned}$$

In general solution will have the form

$$P = \sum_{m=0}^{\infty} P_m(r, z, t) e^{im\theta},$$

but we confine our solutions as in (3.1) for a single mode  $m$ . For  $m \geq 1$  the solution is referred to as the spin mode solution. In physical situations they correspond to the modes provided by a turbofan engine. For  $m = 0$  (3.1) describes a plane wave. The

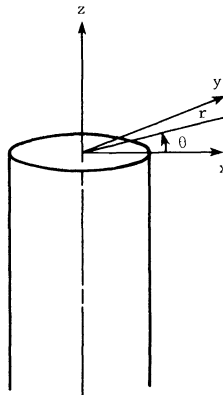


FIG. 1

more important case is when  $m > 0$  (a spinning mode). Then the system (2.7) becomes

$$\begin{aligned}
 \frac{\partial P}{\partial t} + u_z + v_r + \frac{v + imw}{r} &= 0, \\
 \frac{\partial u}{\partial t} + \frac{\partial P}{\partial z} &= 0, \\
 \frac{\partial v}{\partial t} + \frac{\partial P}{\partial r} &= 0, \\
 \frac{\partial w}{\partial t} + \frac{im}{r} P &= 0,
 \end{aligned}
 \tag{3.2}$$

and the problem reduces to solving the system (3.2) with appropriate boundary conditions.

The method we use to solve (3.2) is an explicit method which is fourth-order accurate in space and second-order accurate in time and is due to Gottlieb and Turkel [4]. This is a higher-order accurate version of the MacCormack scheme. The use of an explicit method has been recently advocated by Baumeister [1]. The major advantages are drastically reduced storage requirements and programming simplicity. Baumeister demonstrated the effectiveness of this approach for internal sound propagation with spinning modes [2]. The work in this paper extends this idea to the full radiation problem with a more accurate computational scheme.

A typical computational domain is depicted in Fig. 2. Referring to this figure, the computations are carried out in the rectangular region which is bounded by an inflow boundary and the farfield boundary. Thickness of the duct is allowed by a mesh size thickness in the  $r$  direction. The solution for large times is extremely sensitive to the inflow condition and also the farfield condition [9]. The solution is assumed to start from a state of rest, i.e.,  $P = u = v = w = 0$  at time  $t = 0$ . The system (3.2) can be written in the form

$$w_t + F_z + G_r = H,
 \tag{3.3}$$

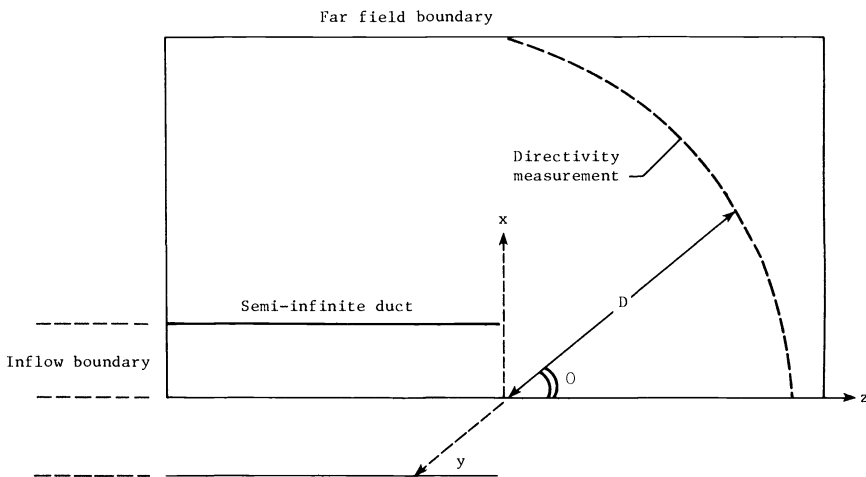


FIG. 2



where

$$(3.4) \quad w = \begin{bmatrix} P \\ u \\ v \\ w \end{bmatrix}, \quad F = \begin{bmatrix} u \\ P \\ 0 \\ 0 \end{bmatrix}, \quad G = \begin{bmatrix} v \\ 0 \\ P \\ 0 \end{bmatrix}, \quad H = \begin{bmatrix} -\frac{v+imw}{r} \\ r \\ 0 \\ 0 \\ -\frac{imP}{r} \end{bmatrix}.$$

We use the method of time splitting to advance the solution from time  $t$  to  $t+2\Delta t$ . If  $L_z(\Delta t)$  and  $L_r(\Delta t)$  denote symbolic solution operators to the one-dimensional equations

$$(3.5) \quad w_t^1 + F_z = H_1, \quad w_t^2 + G_r = H_2,$$

then the solution to (3.3) is advanced by the formula

$$(3.6) \quad w(t+2\Delta t) = L_z(\Delta t)L_r(\Delta t)L_r(\Delta t)L_z(\Delta t)w(t).$$

This procedure is second-order accurate in time. The fourth-order accuracy in space depends on formulating the difference scheme. For the one-dimensional system, (3.5), we have

$$(3.7) \quad \bar{w}_i(t+\Delta t) = w_i(t) + \frac{\Delta t}{6\Delta z}(7F_i - 8F_{i+1} + F_{i+2}) + \Delta t H_i,$$

$$w_i(t+\Delta t) = \frac{1}{2} \left[ w_i(t) + \bar{w}_i(t+\Delta t) + \frac{\Delta t}{6\Delta z}(-7\bar{F}_i + 8\bar{F}_{i-1} - \bar{F}_{i-2}) + \Delta t \bar{H}_i \right],$$

where  $\bar{F}_i$  denote  $F$  evaluated at  $\bar{w}_i$ , etc. This formula contains a forward predictor and a backward corrector. This is second-order accurate in space. One can formulate another variant which contains a backward predictor and a forward corrector which is also second-order accurate in space. In order to achieve fourth-order accuracy we alternate (3.7) and its variant at each time step. If there are  $N$  intervals with nodes at  $z_i$  ( $i=0, 1, \dots, N$ ), then the predictor in (3.7) cannot be used at  $i=N-1$  and at  $i=N$  and the corrector cannot be used at  $i=0$  and 1. Similar situations occur for the other variant too. At these points we extrapolate the fluxes using third-order extrapolations. For the right boundary we use the extrapolation formulae

$$(3.8) \quad F_{N+1} = 4F_N - 6F_{N-1} + 4F_{N-2} - F_{N-3},$$

$$F_{N+2} = 4F_{N+1} - 6F_N + 4F_{N-1} - F_{N-2},$$

and for the left boundary

$$(3.9) \quad F_0 = 4F_1 - 6F_2 + 4F_3 - F_4,$$

$$F_1 = 4F_0 - 6F_1 + 4F_2 - F_3.$$

Since we are interested in time harmonic solutions, the numerical solution is monitored until the transient has passed out of the computational domain and the solution achieves a steady time harmonic dependence.

**4. Boundary conditions.** A very important feature of our work is in obtaining appropriate boundary conditions. We derive our boundary conditions which are appropriate to an experiment carried out at NASA Langley Research Center [14]. The boundary conditions consist of two major parts. The first part is derivation of an inflow

condition which will correctly model the sound source. The second is an accurate farfield boundary condition which will simulate outgoing radiation.

*Inflow Conditions.* To derive inflow boundary conditions we consider the time harmonic case and, in particular, equation (2.8). We look for spinning mode solutions of the form

$$\hat{p} = \bar{P}(r, z) e^{im\theta}.$$

Inserting this into (2.8) yields

$$(4.1) \quad \frac{1}{r} \frac{\partial}{\partial r} \left( r \frac{\partial \bar{P}}{\partial r} \right) + \left( k^2 - \frac{m^2}{r^2} \right) \bar{P} + \frac{\partial^2 \bar{P}}{\partial z^2} = 0.$$

If we separate variables by setting

$$\bar{P}(r, z) = f(r)g(z),$$

we obtain

$$f(r) = J_m(\beta r) \quad \text{and} \quad g(z) = e^{\pm ilz},$$

where

$$(4.2) \quad \beta^2 = k^2 - l^2,$$

and  $l$  is to be determined.

If  $a$  is the radius of the duct (here  $a = \frac{1}{2}$  due to the nondimensionalization), then the usual boundary condition on the pipe is the hard wall condition

$$\frac{\partial \hat{p}}{\partial n} = 0 \quad \text{on } r = a.$$

This gives

$$\frac{d}{dr} (J_m(\beta r)) = 0 \quad \text{on } r = a,$$

or

$$\beta a = \lambda_{mn} \quad (n = 0, 1, 2, \dots),$$

where  $\lambda_{mn}$ 's are the zeros of the functions  $J'_m(z)$ . From (4.2), using the appropriate subscript corresponding to  $\lambda_{mn}$ , we have

$$(4.3) \quad (l_{mn}a)^2 = (ka)^2 - \lambda_{mn}^2.$$

DEFINITION. If  $ka > \lambda_{mn}$ , we say the mode  $(m, n)$  is propagating. Otherwise it is said to be cut-off.

We now consider only propagating modes. Then the solution of (4.1) has the form

$$(4.4) \quad \bar{P}(r, z) = \sum_{n=0}^{\infty} a_n e^{\pm iz\sqrt{(ka)^2 - \lambda_{mn}^2}} J_m\left(\frac{\lambda_{mn}}{a} r\right).$$

It is necessary to consider the case of a single propagating mode. In this situation  $n = 0$ . Dropping the corresponding zero subscripts in (4.3) we consider the values of  $l$  given by

$$l_m a = \sqrt{(ka)^2 - \lambda_m^2}.$$

Then the general solution inside the pipe can be written as a combination of an

incoming wave and a reflected wave. That is

$$(4.5) \quad \hat{p}(r, \theta, z) = (e^{il_m z} + R(k) e^{-il_m z}) J_m \left( \lambda_m \frac{r}{a} \right) e^{im\theta},$$

where  $R$  is the reflection coefficient and is also a function of the wave number  $k$ . Recalling that  $p(r, \theta, z, t) = \hat{p}(r, \theta, z) e^{-ikt}$  and  $p(r, \theta, z, t) = P(r, z, t) e^{im\theta}$ , we have

$$(4.6) \quad P(r, z, t) = (e^{il_m z} + R(k) e^{-il_m z}) J_m \left( \lambda_m \frac{r}{a} \right) e^{-ikt}.$$

The reflection coefficient  $R$  is unknown. Thus we must eliminate  $R$  in (4.6). This is accomplished by taking the time derivative of (4.6) and subtracting the  $k/l_m$  times the  $z$  derivative to get

$$P_t - \frac{k}{l_m} P_z = -2ik e^{il_m z} J_m \left( \lambda_m \frac{r}{a} \right) e^{-ikt},$$

but

$$P_z = -u_t \quad \text{from (3.2).}$$

Thus the above equation becomes

$$(4.7) \quad \left( P + \frac{k}{l_m} u \right)_t = -2ik e^{il_m z} J_m \left( \lambda_m \frac{r}{a} \right) e^{-ikt}.$$

We impose this boundary condition on the incoming boundary  $z = -L$ . We obtain a boundary condition on  $v$  at the inflow by using (3.2),

$$\frac{\partial v}{\partial t} + \frac{\partial P}{\partial r} = 0,$$

together with

$$\frac{\partial P}{\partial r} = \frac{\lambda_m}{a} \frac{J'_m(\lambda_m r/a)}{J_m(\lambda_m r/a)} P$$

to give

$$(4.8) \quad \frac{\partial v}{\partial t} + \frac{\lambda_m}{a} \frac{J'_m(\lambda_m r/a)}{J_m(\lambda_m r/a)} P = 0.$$

Note that the coefficient of  $P$  here contains a singular term at  $r = 0$ . However  $P$  contains a term (from (4.6)) of the form  $J_m(\lambda_m r/a)$ . Thus when  $r = 0$  (4.8) is simply replaced by  $v = 0$ .

*Conditions on the wall.* On the duct wall we consider the boundary condition  $v = 0$  (i.e. a rigid boundary). Using (3.2) this implies

$$(4.9) \quad \frac{\partial P}{\partial r} = 0.$$

We note that a general impedance condition simulating an acoustic liner can be handled without difficulty.

*Conditions on the axis.* When  $m = 0$  the system (3.2) has only three equations for  $p$ ,  $u$ , and  $v$ . The first equation of (3.2) contains a  $v/r$  term. Thus the boundary condition on the axis in this case is

$$(4.10) \quad v = 0 \quad \text{on } r = 0 \quad (m = 0).$$

When  $m = 1$ , the last equation of (3.2) gives

$$\frac{\partial w}{\partial t} + \frac{im}{r}P = 0.$$

Here  $P$  contains a term like  $J_1(\lambda r/a)$  for  $z$  close to  $-L$ . Thus  $\partial w/\partial t$  is nonzero. But from the first equation of (3.2) we have

$$(4.11) \quad v + iw = 0 \quad \text{on } r = 0 \quad (m = 1).$$

For  $m \geq 2$  the first and the last equations of (3.2) give

$$(4.12) \quad v = 0, w = 0 \quad \text{on } r = 0 \quad (m \geq 2).$$

*Far field conditions.* Radiation conditions are applied at the far field boundaries. The development follows that given in [3]. Let  $R$  be the distance ( $R = \sqrt{r^2 + z^2}$ ) from the origin to a point in the far field (see Fig. 2). The condition we impose here is the first member of a family of nonreflecting boundary conditions which are accurate as  $R \rightarrow \infty$ . This condition is

$$\frac{\partial P}{\partial t} + \frac{\partial P}{\partial R} + \frac{P}{R} = 0,$$

where

$$\frac{\partial P}{\partial R} = \frac{\partial P}{\partial z} \cos \alpha + \frac{\partial P}{\partial r} \sin \alpha,$$

where  $\alpha$  is the angle from the  $z$  axis to the far field point. Using the second and third equations of (3.2) we have

$$\frac{\partial P}{\partial R} = -u_t \cos \alpha - v_t \sin \alpha.$$

Thus the radiation condition becomes

$$(4.13) \quad \frac{\partial P}{\partial t} - (u \cos \alpha + v \sin \alpha)_t + \frac{P}{R} = 0.$$

The conditions (4.7) through (4.13) were used to obtain the results discussed in the next section.

**5. Numerical results.** We computed the solutions with the details given in §§ 3 and 4 on a CDC Corp. Cyber-203 machine. The algorithm described above is almost totally vectorizable. For a very low frequency plane wave case the typical number of grid points in the  $(r, z)$ -plane were  $80 \times 100$ . The incoming boundary was kept at  $z = -10d$  (10 diameters) and the radiation boundary was chosen so as to enclose a circle of radius  $10d$ . For high frequencies the typical grid sizes were  $115 \times 135$  and the inflow boundary as varied from  $z = -10d$  to  $z = -8d$ .

To verify the effectiveness of the code we compared our results with asymptotic expansion obtained by Ting and Keller [13] for a low frequency plane wave. To make comparisons we computed the solutions in the duct and on the axis at various stations for a nondimensional frequency  $ka = 0.2$ . Here  $k = 2\pi/\omega$  is the wave number. Results are presented in Table 1.

For high frequencies we compared our results with the Weiner-Hopf results of Savkar and Edelfelt [11] and the experiment done at NASA Langley Research Center [14]. In this experiment the directivity patterns were measured on a circle at 10 diameters

TABLE I  
Comparison with Ting and Keller solution,  
 $ka = 0.2$ .

$Z$	Ting and Keller $ P $	Numerical $ P $
-10	1.5026	1.5054
-9	1.0984	1.1113
-8	.3873	.3544
-7	.4355	.3933
-6	1.1133	1.0874
-5	1.7603	1.7196
-4	1.9280	1.9583
-3	1.8933	1.9097
-2	1.5495	1.5477
-1	1.0279	1.0076

from the open end of the pipe. The test facility has a spin mode synthesizer which can produce both plane and spinning mode wave. Since only the forward radiation pattern is of interest and experimental results are only available from  $0$  to  $90^\circ$  the comparison is restricted to this region. These results are not sensitive to the placement of the outer boundary validating the boundary conditions (4.13). The code produces a directivity pattern which is essentially flat up to  $180^\circ$  although the accuracy of (4.13) would be expected to degrade as  $\theta \rightarrow 180^\circ$ .

The first comparison was made for the plane wave case ( $m=0$ ) and a non-dimensional frequency of  $ka = 3.76$ . Since the experimental results were obtained only in the farfield the sound pressure level was plotted as a function of angle measured from  $z$  axis. The results are presented in Fig. 3 and show good agreement with the experiment.

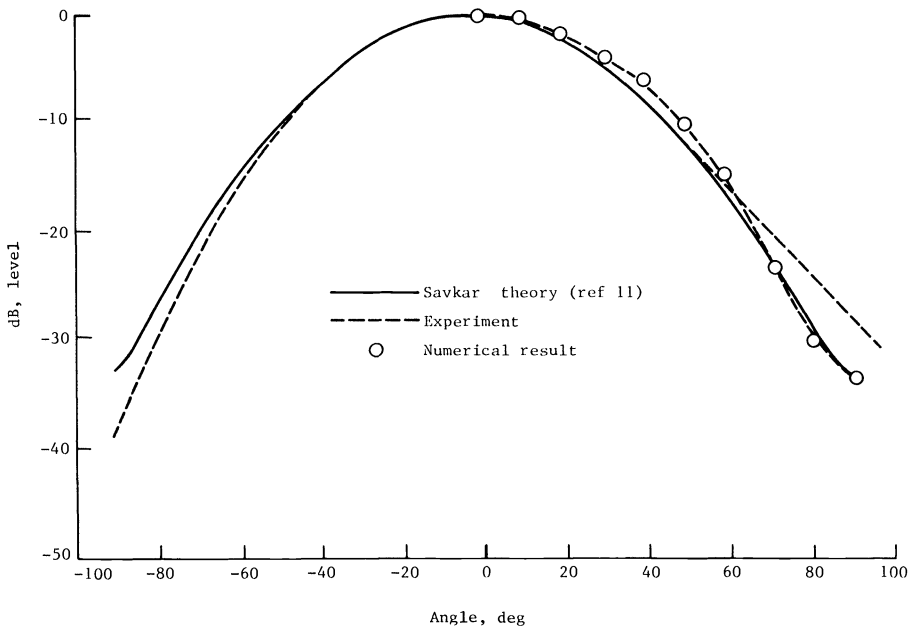


FIG. 3

Figures 4 and 5 show typical comparisons of the spinning mode case with  $m = 2$  and for frequency values  $ka = 3.37$  and  $4.40$ . As in these figures, except the plane wave case, the computed results agree within 5 dB levels. Clearly our results show better comparison than Weiner-Hopf results [11] due to allowance of thickness. In these

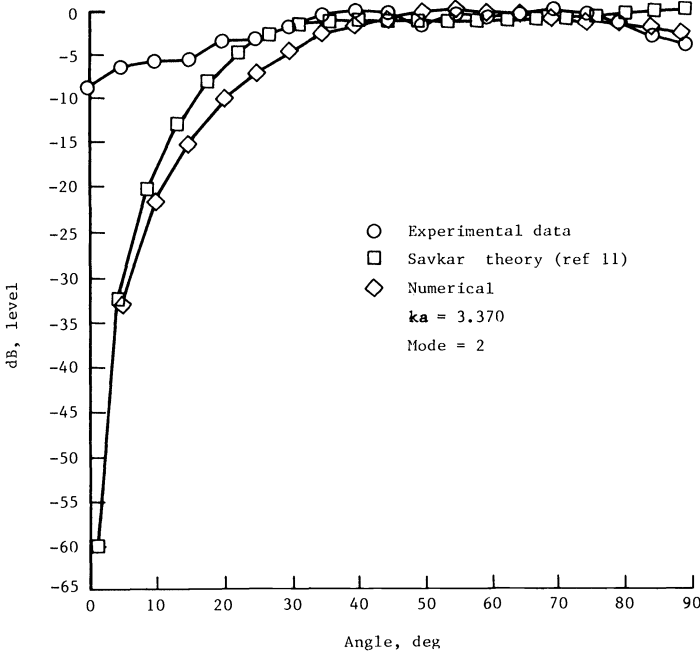


FIG. 4

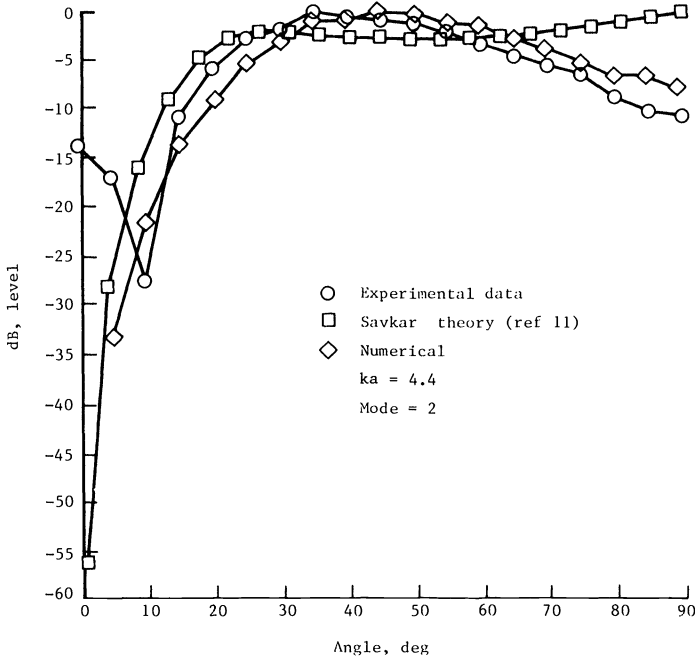


FIG. 5

cases the results near the axis do not compare very well. This is due to the fact that in the experiment it is difficult to completely control other modes and plane waves. This is particularly true for this frequency  $ka = 4.40$  which is close to the next cut-on mode. In the plane wave case the results were unexpectedly good.

**6. Variable geometry ducts.** We consider ducts with a local variable geometry cross section. The duct is assumed to be straight as  $z \rightarrow -\infty$  (see Fig. 6). Thus the inflow boundary conditions previously formulated are still valid. The variable duct is incorporated in the numerical scheme by mapping it into a straight duct. This slightly changes the coefficients in the final system (3.3) but does not degrade the convergence to the time harmonic solution.

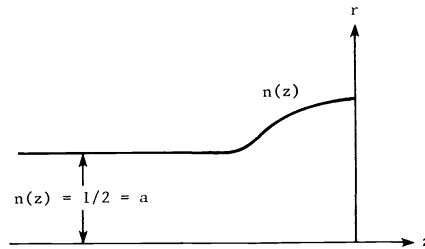


FIG. 6

Suppose the duct configuration is as in Fig. 6. It has a curved boundary near  $z = 0$  and has straight extension everywhere else. This allows us to have the same inflow boundary conditions and the conditions on axis and also the radiation condition. But the boundary conditions on the wall will be changed.

The Euler equations have the form

$$(6.1) \quad w_t + f_z + g_r + h = 0.$$

We introduce the following change of variables:

$$(6.2) \quad z_1 = z, \quad r_1 = \frac{r}{an(z)}.$$

We then use the chain rule to compute  $f_z, g_r$  in terms of  $f_{z_1}$  and  $g_{r_1}$ , etc. where  $r = n(z)$  is the geometry of the duct. This yields

$$(6.3) \quad w_t + f_{z_1} + \left( \frac{1}{an(z)} g - \frac{rn'(z)}{an(z)^2} f \right)_{r_1} + f \frac{n'(z)}{n(z)} + h = 0.$$

This has the same form as (6.1). Thus very minor changes in the difference scheme and in the radiation boundary conditions are required. The boundary condition  $v = 0$  on  $r = a$  is replaced by the vanishing of the normal velocity on the wall. On the surface of the pipe a normal vector is  $(1, an'(z))$  in  $(r, z)$  coordinates. Thus the above condition reduces to

$$(6.4) \quad v - an'(z)u = 0.$$

We simulated a duct where  $n(z)$  has the form

$$n(z) = \begin{cases} .5, & z < -1, \\ .5 - \varepsilon(2z-1)(z+1)^2, & -1 \leq z \leq 0 \end{cases}$$

(see Fig. 6). The grids of the computational domain follow the same geometry. For  $\varepsilon = .15$  the results we obtained are shown in Fig. 7 compared with the straight duct

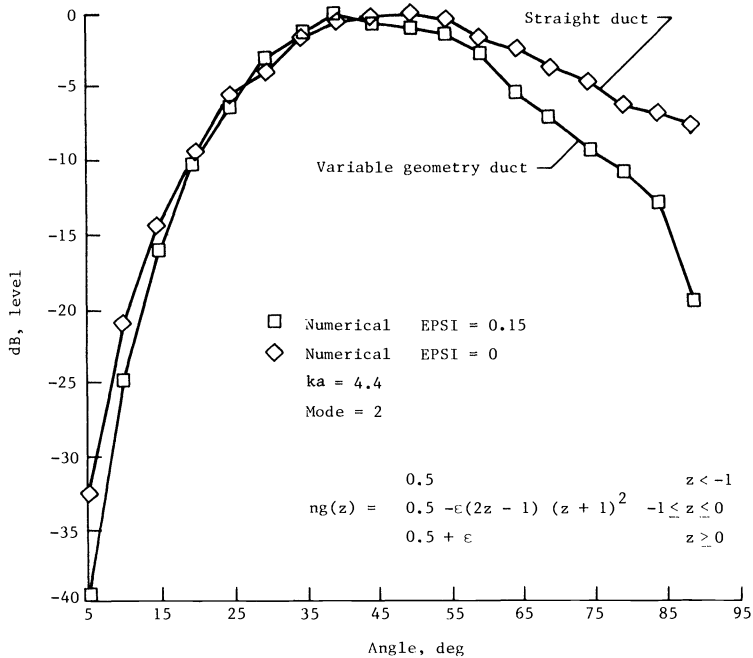


FIG. 7

situation. The dB level reduces at 90° by about 10 dB. This indicates the importance of the nozzle geometry in determining the far field radiation pattern.

## REFERENCES

- [1] K. J. BAUMEISTER, *Numerical techniques in linear duct acoustics*, NASA TM-82730, 1981.
- [2] ———, *Influence of exit impedance on finite difference solutions of transient acoustic mode propagation in ducts*, ASME Paper No. 81-WA/NCA-13, 1981.
- [3] A. BAYLISS AND E. TURKEL, *Radiation boundary conditions for wave-like equations*, *Comm. Pure Appl. Math.*, 33, 6 (1980), pp. 707-725.
- [4] D. GOTTLIEB AND E. TURKEL, *Dissipative two-four methods for time dependent problems*, *Math. Comp.*, 30 (1976), pp. 703-723.
- [5] S. J. HOROWITZ, R. K. SIGMANN AND B. T. ZINN, *An iterative finite element-integral technique for predicting sound radiation from turbofan inlets in steady flight*, AIAA Paper 82-0124, 1982.
- [6] G. A. KRIEGSMANN AND C. S. MORAWETZ, *Solving the Helmholtz equation for exterior problems with a variable index of refraction*, *this Journal*, 1 (1980), pp. 371-385.
- [7] G. A. KRIEGSMANN, *Radiation conditions for wave guide problems*, *this Journal*, 3 (1982), pp. 318-326.
- [8] H. LEVINE AND J. SCHWINGER, *On the radiation of sound from an unflanged circular pipe*, *Phys. Rev.*, 73 (1948), pp. 383-406.
- [9] L. MAESTRELLO, A. BAYLISS AND E. TURKEL, *On the interaction between a sound pulse with shear layer of an axisymmetric jet*, *J. Sound Vib.*, 74 (1981), pp. 281-301.
- [10] S. D. SAVKAR, *Radiation of cylindrical duct acoustics modes with flow mismatch*, *J. Sound Vib.*, 42 (1975), pp. 363-386.
- [11] S. D. SAVKAR AND I. H. EDELFEIT, *Radiation of cylindrical duct acoustic modes with flow mismatch*, NASA CR-132613, 1975.
- [12] A. TAFLOVE AND K. R. UMASHANKAR, *Solution of Complex Electromagnetic Penetration and Scattering Problems in Unbounded Regions*, in *Computational Methods for Infinite Domain Media-Structure Interaction*, A. J. Dalinowski, ed., 1981, pp. 83-114.
- [13] L. TING AND J. B. KELLER, *Radiation from the open end of a cylindrical or conical pipe and scattering from the end of a rod or slab*, *J. Acoust. Soc. Amer.*, 61 (1977), pp. 1439-1444.
- [14] J. M. VILLE AND R. J. SILCOX, *Experimental investigation of the radiation of sound from an unflanged duct and a bellmouth including the flow effect*, NASA TP-1697, 1980.



## EXPERIMENTS WITH QUASI-NEWTON METHODS IN SOLVING STIFF ODE SYSTEMS\*

PETER N. BROWN†, ALAN C. HINDMARSH‡ AND HOMER F. WALKER†

**Abstract.** A nonlinear algebraic system must be solved at each step of the integration of a stiff system of ordinary differential equations by methods based on backward differentiation formulas. Quasi-Newton methods are of potential benefit in solving these algebraic problems. Three types of quasi-Newton methods are studied for this purpose—Doolittle LU updates, and Broyden's first and second methods performed implicitly. Detailed algorithms are given. Tests on some large stiff systems show that significant benefits can be obtained for some problems.

**Key words.** ordinary differential equations, stiff systems, quasi-Newton methods

**1. Introduction.** The numerical solution of stiff systems of ordinary differential equations (ODE's) relies heavily on methods for solving systems of algebraic equations. If the ODE system is nonlinear, then so are the algebraic systems that one must solve. Quasi-Newton methods, by which we mean primarily those which are in some sense generalizations of the one-dimensional secant method, have been found to be very successful methods for solving nonlinear algebraic systems. Over the last decade, a great deal of progress has been made in determining very effective quasi-Newton methods, especially for classes of problems which have in common some special structure which can be exploited.

Recent developments in quasi-Newton methods have a potential for application in the context of solving stiff ODE's. The most challenging ODE problems, for which the need for efficient algebraic system-solving methods is usually greatest, are generally those for which the algebraic systems to be solved are very large, have a Jacobian matrix which is sparse (e.g., banded), and have significant expense associated with function and Jacobian evaluations. A major source of such ODE problems is the solution of time-dependent partial differential equations by the method of lines (discretizing in space only) [15], [16]. Most quasi-Newton methods require relatively few Jacobian evaluations (or function evaluations if Jacobians are being approximated by difference quotients) and can often be implemented to offer savings on arithmetic as well. Furthermore, algebraic systems with sparse Jacobians have special structure which can be exploited in quasi-Newton methods. Preliminary studies of the use of quasi-Newton methods in a stiff ODE method were done by Hindmarsh and Byrne [17] and Alfeld [1].

In the following, we consider the application of three particular quasi-Newton methods to the solution of stiff ODE's. These methods are intended primarily for use on large algebraic systems with sparse Jacobians. The focus here is on ODE's for which the associated algebraic systems have sparse Jacobians and are so large that not only function and Jacobian evaluations but also storage and the cost of arithmetic are major concerns. The remainder of this introduction provides a very brief background on stiff ODE's, certain procedures for solving them numerically, and quasi-Newton methods.

---

\* Received by the editors December 29, 1982, and in revised form October 12, 1983. This work was performed under the auspices of the U.S. Department of Energy by the Lawrence Livermore National Laboratory under contract No. W-7405-Eng-48, and supported by the DOE Office of Basic Energy Sciences, Mathematical Sciences Branch.

† Department of Mathematics, University of Houston, Houston, Texas 77004.

‡ Mathematics and Statistics Division, L-316, Lawrence Livermore National Laboratory, University of California, Livermore, California 94550.

In the sequel, we describe the specific quasi-Newton methods of interest, their implementation in a particular algorithm for solving stiff ODE's, and the outcome of applying the resulting procedures to several test problems. We conclude with a summary of overall results and an outline of future areas of investigation.

**1.1. Stiff ODE's.** We consider an initial value problem for an ODE system,

$$(1.1) \quad \dot{y} = f(t, y), \quad y(t_0) = y_0$$

where the dot denotes  $d/dt$  and  $y$  is a vector of size  $N$ . The ODE in (1.1) is stiff if, roughly speaking, it contains a rapid decay process—i.e. rapid by comparison to the time scale of interest for the whole problem. Precise definitions of stiffness vary [22], but usually make reference to the Jacobian matrix,

$$(1.2) \quad J = \partial f / \partial y = J(t, y).$$

Stiffness means that at least one eigenvalue of  $J(t, y)$  has a very large negative real part, when evaluated on the solution curve.

A large and very popular class of numerical methods for ODE initial value problems is that of linear multistep methods. These have the form

$$(1.3) \quad y_n = \sum_{j=1}^{K_1} \alpha_j y_{n-j} + h \sum_{j=0}^{K_2} \beta_j \dot{y}_{n-j}$$

where  $y_k$  denotes the numerical approximation to  $y(t_k)$ ,  $h = t_k - t_{k-1}$  is considered fixed, and  $\dot{y}_k$  denotes  $f(t_k, y_k)$ . (Variable-step analogues of (1.3) also exist but will not be considered here.) The popular BDF (backward differentiation formula) method corresponds to the case  $K_2 = 0$ ,  $K_1 = q =$  method order, and this method has been used extensively for stiff systems [11], [13], [16], [24]. We shall restrict our attention here to the context of a general purpose initial value ODE solver called LSODE [14], [16], which uses the BDF method in the stiff case. Thus at each time step, LSODE must solve an algebraic system

$$(1.4) \quad \begin{aligned} 0 &= F_n(y_n) \equiv y_n - h\beta_0 f(t_n, y_n) - a_n, \\ a_n &\equiv \sum_{j=1}^q \alpha_j y_{n-j} \end{aligned}$$

in which  $\beta_0 > 0$ .

In its unmodified form, LSODE solves (1.4) by modified Newton iteration, in which a prediction  $y_n(0)$  is formed using an explicit formula of the type (1.3) and corrected by iterations

$$(1.5) \quad P_n(y_n(m+1) - y_n(m)) = -F_n(y_n(m)).$$

Here the iteration matrix  $P_n$  is an approximation to  $\partial F_n / \partial y_n$ , i.e.,

$$(1.6) \quad P_n \approx \frac{\partial F_n}{\partial y_n} = I - h\beta_0 J(t_n, y_n),$$

that is held fixed for the iterations, and is usually also held fixed over several time steps. The LSODE user has the option of specifying  $J$  as either a full or a banded matrix, and as either supplied by a user subroutine or computed internally by difference quotient approximations. In all cases, the linear system (1.5) is solved by doing an LU decomposition of  $P_n$  (at the time it is formed) and using that for all iterations (on all steps) until a decision is made to reevaluate  $P_n$ . A convergence test is made on the basis of iterate differences. The time step itself must also pass a test on estimated local

truncation error, and the step size  $h$  and order  $q$  are adjusted dynamically on the basis of such tests. A change of step size from  $h$  to  $h'$  is accomplished, in effect, by interpolating in the existing solution history (the  $y_{n-j}$  spaced at  $h$ ) to get history at the new spacing  $h'$ .

**1.2. Quasi-Newton methods.** By a quasi-Newton method for solving an algebraic system  $F(y) = 0$ ,  $F: R^N \rightarrow R^N$ , we mean here any method which generates a sequence of approximate solutions  $\{y(k)\}_{k=1,2,\dots}$  from an initial approximation  $y(0)$  by means of an iteration of the form

$$(1.7) \quad y(k+1) = y(k) - B_k^{-1}F(y(k)).$$

Such methods are regarded as variants of Newton's method, in which  $B_k$  is the Jacobian matrix  $\partial F/\partial y(y(k))$ , and so  $B_k$  is considered to be an approximation to  $\partial F/\partial y(y(k))$ .

Our particular interest is in quasi-Newton methods which are generalizations of the one-dimensional secant method. There one obtains  $B_{k+1}$  from  $B_k$  for some  $k$  by updating  $B_k$  so that for  $s(k) = y(k+1) - y(k)$  and  $z(k) = F(y(k+1)) - F(y(k))$ , the secant equation

$$(1.8) \quad B_{k+1}s(k) = z(k)$$

is satisfied as nearly as possible (in some sense) among all matrices satisfying any auxiliary conditions which might be imposed on  $B_{k+1}$ . Auxiliary conditions on  $B_{k+1}$  usually take the form of a requirement that  $B_{k+1}$  reflect some special structure of  $\partial F/\partial y$  such as symmetry or a particular pattern of sparsity. The qualifier "as nearly as possible" is necessary because there may not always exist a matrix satisfying both (1.8) and the auxiliary conditions. For a full discussion of these methods, see Dennis and Moré [6] or Dennis and Schnabel [7].

As indicated at the outset, the focus here is on ODE's (1.1) which are stiff and for which  $N$  is large and the Jacobian  $J$  given by (1.2) is sparse. Our interest is in considering secant-update quasi-Newton methods as alternatives to the modified Newton algorithm (1.5) for solving the algebraic systems (1.4) which arise in applying BDF methods to such ODE's. In the remainder of this introduction, we touch on several quasi-Newton methods which can potentially use the sparsity of  $J$  to advantage. More specific descriptions of the methods of particular interest follow in the next section.

The sparse Broyden update given by Schubert [21] and Broyden [4] determines a quasi-Newton method which takes sparsity into account. This update has the property that if  $B_k$  in (1.7) has a desired pattern of sparsity, then so does the updated matrix  $B_{k+1}$ . A similar update which preserves not only sparsity but also symmetry in determining  $B_{k+1}$  from  $B_k$  has been given by Marwil [18] and Toint [23]. Methods employing these updates have desirable local convergence properties but, unfortunately, require new Jacobian factorizations after each update if direct linear algebra methods are used to obtain each step  $-B_k^{-1}F(y(k))$ . Here, we assume that direct methods are used for solving linear systems and, in particular, that matrix factorizations are  $LU$  factorizations as in LSODE. It is further understood that the problems of interest here are of such size that  $LU$  factorizations involve considerable expense, and so we regard the sparse Broyden and sparse symmetric updates as unsuitable in the present context, although they may well prove useful in situations which warrant the use of iterative methods for solving linear systems.

The sparse Broyden update, however, is fundamental to the first quasi-Newton method considered in the sequel, that which employs the Doolittle  $LU$  updating

procedure of Dennis and Moré [5]. To describe this method briefly, let us suppose that in an iteration (1.7) one has obtained, for some value of  $k$ , a Doolittle decomposition

$$(1.9) \quad B_k = P_k L_k U_k,$$

using a partial pivoting strategy. In (1.9),  $P_k$  is a permutation matrix,  $L_k$  is a unit lower triangular matrix, and  $U_k$  is an upper triangular matrix. Then further approximate Jacobians are obtained by taking

$$(1.10) \quad B_{k+j} = P_k L_k U_{k+j}$$

for as many values of  $j$  as possible, where the sequence of upper triangular matrices  $\{U_{k+j}\}$  is generated through (essentially) sparse Broyden updates. These updates are done for some but not necessarily all values of  $j$ ; when done, they are determined by the secant equations

$$(1.11) \quad U_{k+j} s(k+j-1) = L_k^{-1} P_k^{-1} z(k+j-1).$$

If the successive matrices  $B_{k+j}$  are required to have a particular pattern of sparsity which is associated with certain corresponding patterns of sparsity of  $L_k$  and  $U_{k+j}$  (as is the case when the matrices  $B_{k+j}$  have a particular band structure), then the advantage of the Doolittle updating method in exploiting sparsity is clear: One maintains (sparse) factors of updated matrices having the desired pattern of sparsity without having to pay for additional factorizations or storage.

The other two quasi-Newton methods considered here take another approach to exploiting sparsity. In this approach, one obtains a factorization of  $B_k$  for some  $k$  and maintains subsequent approximate Jacobians implicitly, i.e., without explicitly updating  $B_k$ , its successors, or their factors, by creating (with  $B_k$ ) and storing certain auxiliary vectors which incorporate update information. (The number of auxiliary vectors needed for each implicit update is equal to twice the rank of the update.)

Since quasi-Newton methods employing implicit updating incur certain storage and arithmetic costs associated with the auxiliary vectors, such methods are most likely to be effective for problems in which the price of some additional storage and arithmetic might be outweighed by the use of low-rank updates which have proved to be highly successful in solving general algebraic systems with full Jacobians. Matthies and Strang [19], Engelman [9], Engelman, Strang, and Bathe [10], and Geradin, Idelsohn, and Hogge [12] report effective implementations of implicit updating methods which employ several generally successful rank one and rank two updates. Here, we consider the implicit implementation of two updates due to Broyden [3]. The first Broyden update is widely regarded as the most successful update for general systems of nonlinear equations. The second Broyden update is considered to be less effective on general systems than Broyden's first update; however, it has been conjectured (see Alfeld [1]) that Broyden's second update performs particularly well in the context of solving stiff ODE's.

**2. The quasi-Newton methods.** In this section, we describe more specifically the quasi-Newton methods of interest. It is intended here that these methods be applied to a sequence of problems (1.4) for many values of  $n$  and that useful information about these problems be carried over from one value of  $n$  to the next. For convenience, however, we describe these methods in the context of solving a single system  $F(y) = 0$ ,  $F: R^N \rightarrow R^N$ , with an iteration (1.7), beginning with an initial approximate solution  $y(0)$  and an initial approximate Jacobian  $B_0 \approx \partial F / \partial y(y(0))$ . The reader is safe in

assuming that  $F = F_n$ , that  $y(0)$  is an initial approximation of  $y_n$ , and that all discussion below refers to the same time step, step  $n$ .

In describing the quasi-Newton methods below, our principal interest is in the updating algorithms used in them. However, efficient implementations of the updating algorithms must be well coordinated with the algorithms for determining iteration steps, and so the algorithms given here are somewhat broader in scope than updating algorithms per se. All of our quasi-Newton methods assume that  $B_0$  is given in a form convenient for solving linear equations. They also depend on singling out particular values of  $k$  in (1.7) at which to update subsequent approximate Jacobians  $B_k$ . The rules for determining when to perform updates and when the iterates are sufficiently near the solution are outlined in the next section, in the discussion of our implementation of these methods in LSODE.

The updating algorithms described in the following are based on the well-known first and second updates of Broyden and the sparse Broyden update (see, for example, [6] or [7]). Here, we use variations of these updates which take into account an implicit rescaling of the independent and dependent variables by a nonsingular diagonal scaling matrix. Suppose that  $D$  is a given nonsingular diagonal scaling matrix such that  $\tilde{y} = Dy$  and the problem  $\tilde{F}(\tilde{y}) \equiv DF(D^{-1}\tilde{y}) = 0$  can be considered well-scaled. Such a scaling matrix is determined automatically by LSODE from user-supplied tolerance information. It is fixed throughout each time step, although it may vary from step to step. The manner of incorporating such a rescaling in an update is illustrated in [7, p. 187] for the first Broyden update.

We remark at this point that since our methods all take the full quasi-Newton step  $s(k) = -B_k^{-1}F(y(k))$  at each iteration of (1.7), one can save arithmetic by substituting  $F(y(k+1))$  for  $[z(k) - B_k s(k)]$  wherever the latter expression appears in an update formula. Since it is desirable to keep computational cost of updating as low as possible, we incorporate this labor-saving substitution throughout our descriptions and implementations of the methods of interest here, even though it is not always regarded as advisable in other settings.

**2.1. The Doolittle LU updating method.** Suppose that at the initial iteration of (1.7) one is given a Doolittle decomposition  $B_0 = P_0 L_0 U_0$ , and suppose that  $B_0$  and its successors are required to have a particular pattern of sparsity which in turn imposes a certain pattern of sparsity on their lower- and upper-triangular Doolittle factors. Denote by  $\mathcal{U}$  the subspace of  $R^{N \times N}$  consisting of all matrices having the pattern of sparsity required of these upper-triangular factors. For  $i = 1, \dots, N$ , let  $S_i$  indicate the "sparsity" projection operator on  $R^N$  which imposes the sparsity pattern of the  $i$ th row of matrices in  $\mathcal{U}$  on vectors in  $R^N$ , i.e., which for  $j = 1, \dots, N$  replaces the  $j$ th component of a vector in  $R^N$  by zero if the  $ij$ th entry of all matrices in  $\mathcal{U}$  must be zero and otherwise leaves it unchanged. Further denote the  $i$ th component of  $v \in R^N$  by  $v^{(i)}$ , the  $i$ th row of  $U \in \mathcal{U}$  by  $U^{(i)}$ , and the Euclidean norm on  $R^N$  by  $|\cdot|$ .

We are given a nonsingular diagonal scaling matrix  $D$  and a parameter  $\epsilon$ ,  $0 < \epsilon \leq 1$ . (The purpose of  $\epsilon$  is explained below.) The following is our algorithm for Doolittle LU updating.

**ALGORITHM 2.1.**

*At step 0 of (1.7).* Suppose that one has  $y(0)$  and  $B_0 = P_0 L_0 U_0$ . Then compute  $F(y(0))$ ,  $s(0) = -B_0^{-1}F(y(0))$ , and  $y(1) = y(0) + s(0)$ , and go on to step 1 if necessary.

*At step k of (1.7),  $k \geq 1$ .* Suppose that one has  $y(k)$ ,  $s(k-1)$ , and  $B_{k-1} = P_0 L_0 U_{k-1}$ . Then do the following:

- (1) Compute  $F(y(k))$  and  $s(k) = -L_0^{-1}P_0^{-1}F(y(k))$ .

(2) If no update is to be made, take  $U_k = U_{k-1}$ ; otherwise do the following for  $i = 1, \dots, N$ :

(a) If  $\varepsilon |Ds(k-1)| < |DS_i s(k-1)|$ , set

$$U_k^{(i)} = U_{k-1}^{(i)} - \frac{s(k)^{(i)}}{[S_i s(k-1)]^T D^2 [S_i s(k-1)]} [S_i s(k-1)]^T D^2;$$

(b) otherwise, take  $U_k^{(i)} = U_{k-1}^{(i)}$ .

(3) Compute  $s(k) \leftarrow U_k^{-1} s(k)$ ,  $y(k+1) = y(k) + s(k)$ , and go on to step  $k+1$  if necessary.

It is clear that  $U_k \in \mathcal{U}$  if  $U_{k-1} \in \mathcal{U}$  and that (1.11) is satisfied provided each test  $\varepsilon |Ds(k-1)| < |DS_i s(k-1)|$  in (2.a) above is passed for  $i = 1, \dots, N$  (in which case  $U_k$  is just the usual scaled sparse Broyden update of  $U_{k-1}$  in  $\mathcal{U}$ ). The purpose of these tests on  $\varepsilon$  is to insure that no correction is made in a row of  $U_{k-1}$  when the projected step  $S_i s(k-1)$  along that row is too small relative to the full step. These tests play an important role in the convergence analysis given in [5]. It can be argued heuristically that these tests are more than a theoretical convenience as follows: If  $B_{k-1} = P_0 L_0 U_{k-1}$  is a good approximation to  $\partial F / \partial y(y(k-1))$ , then  $w = L_0^{-1} P_0^{-1} z(k-1)$  is almost but not quite the result of operating on  $s(k-1)$  with an upper-triangular matrix. In light of the qualifier ‘‘not quite,’’ one sees that the ratios  $w^{(i)} / |S_i s(k-1)|$  can be well-defined but arbitrarily large; thus updating without these tests can do arbitrarily great violence to the approximate Jacobians. Note that large values of  $\varepsilon$  correspond to updating that is more conservative in that fewer row updates are likely to be done. In the experiments with Doolittle  $LU$  updating in LSODE reported in the sequel, we used a value of  $\varepsilon$  roughly equal to the unit roundoff. Such a small choice of  $\varepsilon$  implies that very few of the tests on  $\varepsilon$  will not be passed; a similarly small choice of  $\varepsilon$  is reported to be effective in the experiments in [5]. We also remark that there is a certain *restart* procedure for periodically obtaining a new Jacobian or approximate Jacobian which is included in the method of Dennis and Morfitt and which is necessary for their convergence analysis. Such a restart procedure is not necessary in the updating algorithm above because it is provided for elsewhere in our implementation of the algorithm in LSODE.

**2.2. The implicit Broyden updating methods.** To describe our implicit implementation of Broyden’s first update, we begin by recalling that if  $D$  is a nonsingular diagonal scaling matrix and  $B_{k+1}$  is obtained by a scaled first Broyden update of  $B_k$ , then

$$(2.1) \quad B_{k+1} = B_k + \frac{[z(k) - B_k s(k)]s(k)^T}{s(k)^T D^2 s(k)} D^2.$$

(See Dennis and Schnabel [7, p. 187, formula (8.3.1)].) It follows from the Sherman–Morrison–Woodbury formula (see Ortega and Rheinboldt [20]) that

$$B_{k+1}^{-1} = \left\{ I + \frac{[s(k) - B_k^{-1} z(k)]s(k)^T}{s(k)^T D^2 B_k^{-1} z(k)} D^2 \right\} B_k^{-1}.$$

By extension, if updated matrices  $B_1, \dots, B_l$  are generated from  $B_0$  by (2.1) in conjunction with an iteration (1.7), then one has

$$(2.2) \quad B_l^{-1} = [I + v(l)w(l)^T] \cdots [I + v(1)w(1)^T] B_0^{-1},$$

where the auxiliary vectors  $v(i)$  and  $w(i)$  are given by

$$(2.3) \quad v(i) = \frac{s(i-1) - B_{i-1}^{-1} z(i-1)}{s(i-1)^T D^2 B_{i-1}^{-1} z(i-1)},$$

and

$$(2.4) \quad w(i) = D^2s(i-1),$$

for  $i = 1, \dots, l$ .

Now suppose that at the initial iteration of (1.7) one is given  $B_0$  in factored form or in some other form convenient for the solution of linear equations. In the applications of interest here, for example,  $B_0$  is likely to be specified by its matrix factors together with a set of auxiliary vectors such as those appearing in (2.2). Suppose also that a nonsingular diagonal scaling matrix  $D$  is given. The following is our algorithm for implicitly implementing Broyden's first update.

ALGORITHM 2.2.

At step 0 of (1.7). Suppose that one has  $y(0)$  and  $B_0$  in a form convenient for the solution of linear equations. Then compute  $F(y(0))$ ,  $s(0) = -B_0^{-1}F(y(0))$ , and  $y(1) = y(0) + s(0)$ , and go on to step 1 if necessary.

At step  $k$  of (1.7),  $k \geq 1$ . Suppose that one has  $y(k)$ ,  $s(k-1)$ ,  $B_0$ , and also  $v(1), \dots, v(l)$  and  $w(1), \dots, w(l)$ , if  $k > 1$  and  $l$  updates have been made for some  $l$ ,  $1 \leq l \leq k-1$ . Then do the following:

(1) Compute  $F(y(k))$ ,  $s(k) = -B_0^{-1}F(y(k))$ , and if  $l$  updates ( $l > 0$ ) have been made, compute

$$s(k) \leftarrow [I + v(l)w(l)^T] \cdots [I + v(1)w(1)^T]s(k).$$

(2) If no update is to be made, then go to (3); otherwise, compute

$$w(l+1) = D^2s(k-1),$$

$$v(l+1) = \frac{s(k)}{w(l+1)^T[s(k-1) - s(k)]},$$

$$s(k) \leftarrow [I + v(l+1)w(l+1)^T]s(k).$$

(3) Compute  $y(k+1) = y(k) + s(k)$  and go on to step  $k+1$  if necessary.

We note that at the end of part (1) of the algorithm at step  $k$ ,  $k \geq 1$ , one has computed  $s(k) = -B_{k-1}^{-1}F(y(k))$ , where  $B_{k-1}^{-1}$  is implicitly taken to be either  $B_0^{-1}$  if no updating has been done since the initial step or  $[I + v(l)w(l)^T] \cdots [I + v(1)w(1)^T]B_0^{-1}$  if  $l$  updates have been made since the initial step. If no update is made at the current step, i.e., if  $B_k^{-1} = B_{k-1}^{-1}$  implicitly, then one accepts this  $s(k)$  as the iteration step. If an update is made, then one first computes  $v(l+1)$  and  $w(l+1)$  according to (2.3) and (2.4), respectively, so that  $B_k^{-1} = [I + v(l+1)w(l+1)^T] \cdots [I + v(1)w(1)^T]B_0^{-1}$  implicitly, and then updates  $s(k)$  to obtain  $s(k) = -B_k^{-1}F(y(k))$ .

It is evident that each update that is made requires the formation and storage of two vectors. If  $N$  is large, then storage and, hence, the number of updates that one can make, may be sharply limited. In any case, there is certainly a maximum number of updates that can be accommodated in practice. When this number is reached, one has a variety of options such as obtaining a new (approximate) Jacobian from scratch, discarding all update vectors and restarting the updating of  $B_0$  from scratch, replacing the early update vectors with current ones, or simply doing no additional updating. We chose the last option in our implementation with a maximum allowable number of updates equal to 5, since our intention was to update only very infrequently and, therefore, we felt it likely that the code would call for a new Jacobian more often than this maximum allowable number of updates would be reached.

It should also be mentioned that some arithmetic is incurred not only in forming the update vectors but also in using them to determine subsequent iteration steps.

However, most of the work of forming the update vectors is also applied to forming iteration steps concurrently. Furthermore, one sees from the algorithm that using the update vectors to form an iteration step or an additional update vector is unlikely to be regarded as costly, especially on a computer which performs vector operations efficiently.

To describe our implicit implementation of Broyden's second update, we first note that this update is most conveniently written in the form of an inverse analogue of (2.1), which is

$$\begin{aligned} B_{k+1}^{-1} &= B_k^{-1} + \frac{[s(k) - B_k^{-1}z(k)]z(k)^T}{z(k)^T D^2 z(k)} D^2 \\ &= B_k^{-1} \left\{ I - \frac{[z(k) - B_k s(k)]z(k)^T}{z(k)^T D^2 z(k)} D^2 \right\}. \end{aligned}$$

If updated matrices  $B_1, \dots, B_l$  are generated by this formula in conjunction with an iteration (1.7), then the counterpart of (2.2) is

$$B_l^{-1} = B_0^{-1} [I - t(1)u(1)^T] \cdots [I - t(l)u(l)^T],$$

where

$$t(i) = \frac{[z(i-1) - B_{i-1}s(i-1)]}{z(i-1)^T D^2 z(i-1)}$$

and

$$u(i) = D^2 z(i-1)$$

for  $i = 1, \dots, l$ .

Suppose that  $B_0$  is given in factored form or in some other form convenient for the solution of linear equations. Let  $D$  be a nonsingular diagonal scaling matrix. Algorithm 2.3 below is our algorithm for implicitly implementing Broyden's second update. We note that remarks similar to those following Algorithm 2.2 above are also appropriate for Algorithm 2.3.

#### ALGORITHM 2.3.

*At step 0 of (1.7).* Suppose that one has  $y(0)$  and  $B_0$  in a form convenient for the solution of linear equations. Then compute  $F(y(0))$ ,  $s(0) = -B_0^{-1}F(y(0))$ , and  $y(1) = y(0) + s(0)$ , and go on to step 1 if necessary.

*At step  $k$  of (1.7),  $k \geq 1$ .* Suppose that one has  $y(k)$ ,  $F(y(k-1))$ ,  $B_0$  and also  $t(1), \dots, t(l)$  and  $u(1), \dots, u(l)$ , if  $k > 1$  and  $l$  updates have been made for some  $l$ ,  $1 \leq l \leq k-1$ . Then do the following:

- (1) Compute  $F(y(k))$ .
- (2) If no update is to be made, compute

$$s(k) = \begin{cases} -B_0^{-1} [I - t(1)u(1)^T] \cdots [I - t(l)u(l)^T] F(y(k)) & \text{if } l > 0 \text{ updates have been made,} \\ -B_0^{-1} F(y(k)) & \text{otherwise} \end{cases}$$

and go to (3). If an update is to be made, compute

$$\begin{aligned} z(k-1) &= F(y(k)) - F(y(k-1)), \\ u(l+1) &= D^2 z(k-1), \quad t(l+1) = F(y(k)) / u(l+1)^T z(k-1), \\ s(k) &= -B_0^{-1} [I - t(1)u(1)^T] \cdots [I - t(l+1)u(l+1)^T] F(y(k)). \end{aligned}$$

- (3) Compute  $y(k+1) = y(k) + s(k)$  and go on to step  $k+1$  if necessary.



**3. Algorithmic implementation.** In implementing each of the three update algorithms described above, the LSODE package was modified so as to perform occasional quasi-Newton updates. In order to describe precisely the algorithm for this, we must first outline the structure and overall algorithm of LSODE, to the extent that this is relevant here.

**3.1. The unmodified algorithm.** Aside from several auxiliary routines of secondary importance, the structure of LSODE (unmodified) is shown in Fig. 1, with the dashed line connections ignored. Subroutine LSODE is a driver, and subroutine STODE performs a single step and associated error control. STODE calls PREPJ to evaluate and do an *LU* factorization of the matrix  $P_n$  of (1.6), and subsequently calls SOLSY to solve the linear system (1.5). (Recall that  $P_n$  approximates  $I - h\beta_0 J(t_n, y_n)$ .) Both of these routines call LINPACK routines [8] to do the matrix operations.

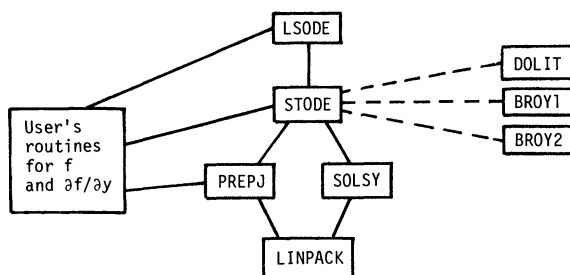


FIG. 1. Simplified overall structure of LSODE package.

Within STODE, the basic algorithm for step  $n$ , in its unmodified form, is as follows:

- (1) Set flag showing whether to reevaluate  $J$ .
- (2) Predict  $y_n(0)$ .
- (3) Compute  $f(t_n, y_n(0))$ ; set  $m = 0$ .
- (4) Call PREPJ if flag is on.
- (5) Form  $F_n(y_n(m))$ .
- (6) Call SOLSY and correct to get  $y_n(m + 1)$ .
- (7) Update estimate of convergence rate constant  $C$  if  $m \geq 1$ .
- (8) Test for convergence.
- (9) If convergence test failed:
  - (a) Set  $m \leftarrow m + 1$ .
  - (b) If  $m < 3$ , compute  $f(t_n, y_n(m))$  and go to 5.
  - (c) If  $m = 3$  and  $J$  is current, set  $h \leftarrow h/4$  and go to 1 (redo step).
  - (d) If  $m = 3$  and  $J$  is not current, set flag to reevaluate  $J$  and go to step (3) (redo step).
- (10) If the convergence test passed, update history, do error test, etc.

In algorithm step (1) above, the decision is made to reevaluate  $J$  (and redo the *LU* factorization of  $P = I - h\beta_0 J$ ) if either

- (a) 20 steps have been taken since the last evaluation of  $J$ , or
- (b) the value of  $h\beta_0$  has changed by more than 30% since  $J$  was last evaluated.

In algorithm step (7), the iterate difference  $s_n(m) = y_n(m + 1) - y_n(m)$  is used, together with  $s_n(m - 1)$  if  $m \geq 1$ , to form the ratio  $DELR = \|s_n(m)\| / \|s_n(m - 1)\|$ , and

$C$  is updated to be the larger of  $.2C$  and DELR.  $C$  is reset to  $.7$  whenever  $J$  is evaluated. The norm is a weighted root-mean-square norm, with weights determined by user-supplied relative and absolute tolerance parameters RTOL and ATOL. (These weights correspond to the diagonal scaling matrix  $D$  referred to in § 2.) The convergence test in step (8) requires the product  $\|s_n(m)\| \min(1, 1.5C)$  to be less than a constant which depends only on  $q$ . This is based on linear convergence, with the idea that  $C\|s_n(m)\|$  is a better estimate of the error in  $y_n(m+1)$  than  $\|s_n(m)\|$  is. Algorithm step (10) includes step and order selection for the next step (if the error test passed) or for redoing the current step (if it failed), but the details of that are not relevant here.

**3.2. The modified algorithm for updating.** There are three main additions to this structure, each of which is a call from STODE to one of the updating routines, shown by the dashed lines in Fig. 1. Subroutines DOLIT, BRO1, and BRO2 perform (respectively) Doolittle updates, Broyden's first update, and Broyden's second update.

Any implementation of an updating strategy in LSODE will necessarily have to include rules which decide when to reevaluate  $J$  and when to perform an update of  $P$ . Within LSODE, at any given step the only feasible measures of the quality of the current  $P$  are the following:

- (i) The ratio of the current value of  $h\beta_0$  to the value as of the last  $J$  evaluation.
- (ii) The number of steps taken since the last  $J$  evaluation.
- (iii)  $RCC = |(h\beta_0)_n / (h\beta_0)_{n-1} - 1|$ , where  $(h\beta_0)_k$  denotes the value of  $h\beta_0$  at step  $k$ .
- (iv)  $DELR = \|s_n(m)\| / \|s_n(m-1)\|$  = the ratio of iterate differences (when  $m \geq 1$ ).

The exact rules chosen for reevaluating  $J$  and updating  $P$  are based on these four quantities, as follows. In the course of the algorithm, a set of six flags is set according to the following rules:

- (i) Flag 1 is turned on if either
  - (a) 60 steps have been taken since the last evaluation of  $J$ ,
  - (b) the value of  $h\beta_0$  has changed by more than 30% since  $J$  was last evaluated,
  - or
  - (c) the value of  $h\beta_0$  has changed by more than 30% from the value on the previous step (i.e.,  $RCC > .3$ ).
- (ii) Flag 2 is turned on if  $.2 < RCC \leq .3$ .
- (iii) Flag 3 is turned on if  $.1 < RCC \leq .2$ .
- (iv) Flag 4 is turned on if  $m \geq 2$  and  $DELR \leq .1$ .
- (v) Flag 5 is turned on if  $m \geq 2$  and  $.1 < DELR \leq 1.0$ .
- (vi) Flag 6 is turned on if  $m \geq 2$  and  $DELR > 1.0$ .

Then time step  $n$  of the integration is given by substituting the following modified steps in the STODE algorithm given in § 3.1 (i.e., replacing step (1) by (1'), etc.):

- (1') (a) Set flag 1 showing whether to reevaluate  $J$ .
- (b) Set flag 2 and flag 3 showing whether to update  $P_n$ .
- (c) Set IUP = 1 if flag 2 or flag 3 is on; otherwise set IUP = 0.
- (6') Update the matrix  $P_n$  if IUP = 1 and  $m > 0$ ; call SOLSY as appropriate; correct to get  $y_n(m+1)$ .
- (8') (a) If flag 2 is on and  $m = 0$ , set  $m \leftarrow m + 1$ , compute  $f(t_n, y_n(m))$  and go to (5) (forcing at least two corrections).
- (b) Test for convergence.
- (9') If convergence test failed:
  - (a) Set  $m \leftarrow m + 1$ .
  - (b) If  $m < 2$ , compute  $f(t_n, y_n(m))$  and go to (5).

- (c) If  $m \geq 2$ , compute  $\text{DELR} = \|s_n(m)\| / \|s_n(m-1)\|$ . Set flag 4, flag 5, and flag 6 according to value of DELR.
- (d) If flag 6 is on, go to (h).
- (e) If  $m < 5$  and flag 4 is on, set  $\text{IUP} = 0$ , compute  $f(t_n, y_n(m))$ , and go to (5).
- (f) If  $m < 5$  and flag 5 is on, set  $\text{IUP} = 1$ , compute  $f(t_n, y_n(m))$ , and go to (5).
- (g) If  $m = 5$ , go to (h).
- (h) If  $J$  is current, set  $h \leftarrow h/4$  and go to (1') (redo step). Otherwise, set flag to reevaluate  $J$  and go to (3) (redo step with the same  $h$ ).

We note that updating is done when either  $.1 < \text{RCC} \leq .3$  or  $.1 < \text{DELR} \leq 1.0$  (with  $m \geq 2$ ), and that at least two corrections are performed when  $.2 < \text{RCC} \leq .3$ . If  $\text{DELR} \leq .1$  and  $m \geq 2$ , then no updating is done regardless of what the other strategies imply. Also, the maximum number of iterations allowed per step in the modification is 5 (compared to 3 in unmodified LSODE) to allow for steps in which the current  $P$  is initially somewhat out of date but after several updates are performed should be sufficiently good to complete that step and possibly several more. The structure of step (6') in the above algorithm depends significantly upon the particular updating scheme being employed, following the algorithm given in § 2. Further details are given in [2].

**4. Numerical tests.** The algorithms described above, and implemented in modified versions of the LSODE package, were tested on various ODE test problems. In this section we give, for each of four problems, a brief description of the problem, numerical results obtained, and some discussion. Three of the four test problems are obtained from time-dependent partial differential equation (PDE) systems solved by the method of lines. Further details on the problem specifications are available in [2]. All of the tests were done on a Cray-1 computer with the CFT compiler.

The algorithms tested included the unaltered LSODE package (as discussed in § 3.1) and versions modified to perform Doolittle and implicit Broyden updates of first and second kinds (as described in § 3.2). In addition, an algorithm was tested which uses the modified Newton strategy from the updating algorithms (of § 3.2) but which never performs matrix updates. This tests the value of the new corrector loop strategy as distinct from the updates themselves.

In what follows, we will use the following abbreviations for the various algorithms:

- LSODE: unaltered LSODE package;
- DOLIT: LSODE with Doolittle  $LU$  updating;
- IMPB1: Implicit updating by Broyden's first method;
- IMPB2: Implicit updating by Broyden's second method;
- LSODE\*: LSODE with new strategy but no updates.

Unless otherwise stated, the algorithms are as described in detail in § 3. However, the various heuristic parameters in the updating strategy were varied somewhat in many of the test runs. Where meaningful, results for altered parameter values are also given.

For each problem and each algorithm, a test run was made and yielded various statistics. Those of interest are defined as follows:

- R.T. = run time (CPU sec);
- NST = number of time steps;
- NFE = number of  $f$  evaluations;
- NJE = number of  $J$  evaluations (= number of  $LU$  decompositions);
- NUP = number of matrix updates.

In all cases, the  $J$  evaluations were done by a user-supplied subroutine.

**4.1. Test problem 1.** This problem is based on a pair of PDE's in two dimensions, representing a simple model of ozone production in the stratosphere with diurnal kinetics. (See also [16] for comparison tests on this problem.) There are two dependent variables  $c^i$  representing concentrations of  $O_1$  and  $O_3$  (ozone) in moles/cm<sup>3</sup>, which vary with altitude  $z$  and horizontal position  $x$ , both in km, with  $0 \leq x \leq 20$ ,  $30 \leq z \leq 50$ , and with time  $t$  in sec,  $0 \leq t \leq 86,400$  (one day). These obey a pair of coupled reaction-diffusion equations:

$$\frac{\partial c^i}{\partial t} = K_h \frac{\partial^2 c^i}{\partial x^2} + \frac{\partial}{\partial z} \left[ K_v(z) \frac{\partial c^i}{\partial z} \right] + R^i(c^1, c^2, t) \quad (i = 1, 2),$$

$$K_h = 4 \cdot 10^{-6}, \quad K_v(z) = 10^{-8} e^{z/5},$$

$$R^1(c^1, c^2, t) = -k_1 c^1 - k_2 c^1 c^2 + k_3(t) \cdot 7.4 \cdot 10^{16} + k_4(t) c^2,$$

$$R^2(c^1, c^2, t) = k_1 c^1 - k_2 c^1 c^2 - k_4(t) c^2,$$

$$k_1 = 6.031, \quad k_2 = 4.66 \cdot 10^{-16},$$

$$k_3(t) = \begin{cases} \exp[-22.62/\sin(\pi t/43,200)] & \text{for } t < 43,200, \\ 0 & \text{otherwise,} \end{cases}$$

$$k_4(t) = \begin{cases} \exp[-7.601/\sin(\pi t/43,200)] & \text{for } t < 43,200, \\ 0 & \text{otherwise.} \end{cases}$$

Homogeneous Neumann boundary conditions are posed. The initial condition functions are polynomials chosen to be slightly peaked in the center and consistent with the boundary conditions:

$$c^1(x, z, 0) = 10^6 \alpha(x) \beta(z), \quad c^2(x, z, 0) = 10^{12} \alpha(x) \beta(z),$$

$$\alpha(x) \equiv 1 - (.1x - 1)^2 + (.1x - 1)^4/2, \quad \beta(z) \equiv 1 - (.1z - 4)^2 + (.1z - 4)^4/2.$$

The PDE's are treated by central differencing, on a rectangular grid with uniform spacings,  $\Delta x = 20/(J - 1)$ ,  $\Delta z = 20/(K - 1)$ . The differencing for the vertical diffusion term is

$$(1/\Delta z^2)[K_v(z_{k+1/2})(c_{i,k+1}^i - c_{jk}^i) - K_v(z_{k-1/2})(c_{jk}^i - c_{i,k-1}^i)].$$

The boundary conditions are simulated by taking  $c_{0,k}^i = c_{2,k}^i$  (all  $k$ ), and similarly on the other boundary segments. The size of the ODE system is  $N = 2JK$ . The variables are indexed first by species, then by  $x$  position, and finally by  $z$  position. Thus in  $\dot{y} = f(t, y)$ , we have  $c_{jk}^i = y_m$ ,  $m = i + 2(j - 1) + 2J(k - 1)$ .

For these tests, we chose  $J = 20$  and  $K = 20$  ( $N = 800$ ). The problem is stiff because of the kinetics, and the Jacobian has half-bandwidths  $ML = MU = 2J = 40$ . (The diffusion terms are a potential cause of stiffness also, but are not in fact, for these choices of  $\Delta x$ ,  $\Delta z$ ,  $K_h$ ,  $K_v$ .) A mixed relative/absolute error tolerance was chosen, with  $RTOL = 10^{-5}$  and  $ATOL = 10^{-3}$ .

The results of testing the five algorithms on this problem are shown in Table 1. In this case, all three updating algorithms produced shorter run times than LSODE or LSODE\*, with IMPB2 being the fastest. By comparison with LSODE, IMPB2 trades 22 updates for a reduction in Jacobian evaluations by 21—a tradeoff that saves over 6 sec. (22%) of CPU time.

**4.2. Test problem 2.** This problem is based on a reaction-diffusion system arising from a Lotka-Volterra competition model, with diffusion effects in two space dimensions included. There are two species densities  $c^i$ , varying over the spatial habitat

TABLE 1  
Test results for problem 1.

Algorithm	R.T.	NST	NFE	NJE	NUP
LSODE	27.76	459	661	85	0
DOLIT	23.20	408	587	67	31
IMPB1	26.73	456	655	80	32
IMPB2	21.64	388	544	64	22
LSODE*	28.06	471	714	83	0

$\Omega = \{(x, z): 0 \leq x \leq 1, 0 \leq z \leq 1.8\}$ , and time  $t$  in sec,  $0 \leq t \leq 10$ . These obey

$$\frac{\partial c^i}{\partial t} = d_i \left( \frac{\partial^2 c^i}{\partial x^2} + \frac{\partial^2 c^i}{\partial z^2} \right) + f^i(c^1, c^2) \quad (i = 1, 2),$$

$$d_1 = .05, \quad d_2 = 1.0,$$

$$f^1(c^1, c^2) = c^1(b_1 - a_{11}c^1 - a_{12}c^2),$$

$$f^2(c^1, c^2) = c^2(b_2 - a_{21}c^1 - a_{22}c^2),$$

$$a_{11} = 10^6, \quad a_{12} = 1, \quad a_{21} = 10^6 - 1, \quad a_{22} = 10^6, \quad b_1 = b_2 = 10^6 - 1 + 10^{-6}.$$

Homogeneous Neumann boundary conditions are imposed. Initial conditions are chosen consistent with the boundary conditions:

$$c^1(x, z, 0) = 500 + 250 \cos(\pi x) \cos(10\pi z/1.8),$$

$$c^2(x, z, 0) = 200 + 150 \cos(10\pi x) \cos(\pi z/1.8).$$

Given the above parameter values and initial conditions, the solution of this reaction-diffusion system converges as  $t \rightarrow \infty$  to the equilibrium solution  $c^1 = c^1_* \equiv 1 - 10^{-6}$ ,  $c^2 = c^2_* \equiv 10^{-6}$ . The two partial differential equations are again treated by central differencing, on a rectangular  $J$  by  $K$  grid with uniform spacings, with boundary conditions treated as before. The size of the ODE system is  $N = 2JK$ , and the variables are indexed by species, then by  $x$  position, and finally by  $z$  position.

For this test, we chose  $J = 20$  and  $K = 20$  ( $N = 800$ ). The problem is stiff mainly because of the interaction terms, and the Jacobian has half-bandwidths  $ML = MU = 2J = 40$ . A mixed relative/absolute error tolerance was chosen, with  $RTOL = 10^{-6}$  and  $ATOL = 10^{-9}$ .

The test results on this problem are given in Table 2. Here, in all three updating algorithms, the updates seem to have had no beneficial effect, and simply increased the cost. The reason may be that for this problem the step size grows steadily throughout the problem, at a rate which forces reevaluations of both the Jacobian matrix and  $P$  every 8-10 steps, regardless of updating method.

TABLE 2  
Test results for problem 2.

Algorithm	R.T.	NST	NFE	NJE	NUP
LSODE	23.92	582	665	66	0
DOLIT	25.20	583	696	69	30
IMPB1	25.20	583	696	69	30
IMPB2	25.33	588	708	70	31
LSODE*	25.63	583	700	72	0

**4.3. Test problem 3.** This problem is an ODE system in population biology which models the interaction of  $N$  species competing for the same limited resource. It is also of Lotka–Volterra type and has the form, with time  $t$  in seconds,

$$\frac{du_i}{dt} = u_i \left( b_i - \sum_{j=1}^N a_{ij} u_j \right) \quad (i = 1, \dots, N).$$

The matrix  $A = (a_{ij})$  is taken to be symmetric and banded. For test purposes, we chose  $N = 100$ , half-bandwidths  $ML = MU = 20$ , and coefficients

$$\begin{aligned} a_{11} = \dots = a_{25,25} &= 1, & a_{26,26} = \dots = a_{50,50} &= 10^6, \\ a_{51,51} = \dots = a_{75,75} &= 10^{10}, & a_{75,75} = \dots = a_{100,100} &= 10^{11}, \\ a_{ij} &= .0002 a_{ii} \text{ for } i \neq j, \quad |i - j| \leq 20. \end{aligned}$$

We define  $b = (b_i)$  by setting  $u^* = (1, \dots, 1)$  and  $b = Au^*$ . Then  $u(t) \equiv u^*$  is an equilibrium solution of the ODE system, to which the solution of the ODE system converges as  $t \rightarrow \infty$ . We chose initial conditions  $u_i(0) = 1.5 i$ , and took  $0 \leq t \leq 10$ . The problem is stiff for the parameters chosen. A mixed relative/absolute error tolerance was chosen with  $RTOL = ATOL = 10^{-6}$ .

The test results on this problem are given in Table 3. Here only IMPB1 and IMPB2 were competitive with LSODE, while DOLIT was not. The explanation may be as offered for Problem 2—steadily and rapidly growing step sizes.

TABLE 3  
Test results for problem 3.

Algorithm	R.T.	NST	NFE	NJE	NUP
LSODE	5.32	652	770	80	0
DOLIT	6.22	694	871	100	36
IMPB1	5.39	646	773	83	24
IMPB2	5.31	642	765	81	29
LSODE*	5.66	670	820	85	0

**4.4. Test problem 4.** Like Problem 2, this problem is based on a reaction-diffusion system arising from a Lotka–Volterra predator–prey model with diffusion effects in two space dimensions. Here the prey and predator species densities vary over  $\Omega = \{(x, z): 0 \leq x \leq 1, 0 \leq z \leq 1\}$  and  $0 \leq t \leq 3$ . The equations are the same as in Problem 2 except with

$$\begin{aligned} f^1(c^1, c^2) &= c^1(b_1 - a_{12}c^2), & f^2(c^1, c^2) &= c^2(-b_2 + a_{21}c^1), \\ b_1 &= 1, & a_{12} &= .1, & a_{21} &= 100, & b_2 &= 1000, \end{aligned}$$

and initial conditions

$$\begin{aligned} c^1(x, z, 0) &= 10 - 5 \cos(\pi x) \cos(10\pi z), \\ c^2(x, z, 0) &= 17 + 5 \cos(10\pi x) \cos(\pi z). \end{aligned}$$

Here the solution becomes spatially homogeneous as  $t \rightarrow \infty$ , and tends to a time-periodic solution of the Lotka–Volterra ODE system modeling the predator–prey interaction without spatial effects, namely  $dc^i/dt = f^i$  ( $i = 1, 2$ ). This last system is alternately stiff and nonstiff depending on the position of the solution in phase space.

The two PDE's are differenced just as in Problem 2, except that the mesh dimensions are  $J = K = 10$  ( $N = 200$ ), and hence the Jacobian has half-bandwidths  $ML = MU = 2J = 20$ . The tolerance parameters used were  $RTOL = 10^{-6}$  and  $ATOL = 10^{-4}$ .

The test results on this problem are given in Table 4. Here IMPB2 gave an 8% reduction in run time, trading 50 updates for 16 fewer Jacobian evaluations. But DOLIT and IMPB1 gave little or no overall cost reduction for this particular algorithm. However, runs were also made with slightly different parameter values in the updating strategy (50% in place of 30% in criterion (b) for flag 1, and .4 in place of .3 in the setting of flag 1 (criterion (c)) and of flag 2; see the detailed algorithm in § 3.2). In these runs, all three updating algorithms ran faster than LSODE, from 15% faster (IMPB2) to 9.5% faster (DOLIT). However, LSODE\* ran 12% faster than LSODE also.

TABLE 4  
Test results for problem 4.

Algorithm	R.T.	NST	NFE	NJE	NUP
LSODE	8.84	1,248	1,635	129	0
DOLIT	8.80	1,125	1,630	125	53
IMPB1	8.86	1,270	1,677	129	58
IMPB2	8.14	1,180	1,560	113	50
LSODE*	8.79	1,242	1,632	127	0

**5. Discussion.** We have presented three quasi-Newton methods and discussed their application to solving the nonlinear algebraic equations arising in the solution of stiff ODE systems by BDF methods. Our focus has been on the case in which the ODE system is very large and the Jacobian of the system is sparse, and the quasi-Newton methods considered here were chosen because of their potential for exploiting sparsity. This investigation has not been exhaustive. For one thing, the testing of the methods chosen here has been somewhat limited; for another, there are other quasi-Newton methods, as well as variations of those considered here, which might also be appropriate for this setting.

It must be recognized, however, that the area under investigation is broad and largely unexplored. From the point of view of solving algebraic equations, the ODE setting is markedly different from that of a fixed algebraic problem. In particular, it is clear that for best results a great deal of interaction should take place between the ODE integration algorithm (the step and order selection and its various heuristic decision rules) and the algorithm implementing any given quasi-Newton method (and its heuristics). We do not claim to have achieved an optimal merge of the two, but we believe that we have made the most serious attempt to date at doing so.

Our test results show that, *for some problems*, the combined ODE and quasi-Newton algorithms considered here can offer significant improvements over the unmodified algorithm. We found further that, in our tests, the implicit updates by Broyden's second method came the closest to being consistently beneficial (when updates of any kind were beneficial). The Doolittle method and updates based on Broyden's first method usually (but not always) did a poorer job.

The test results also suggest that, *for some problems*, the quasi-Newton methods studied here may not be capable of reducing the total costs. Specifically, the potential helpfulness of the updates used here seems to be precluded for problems in which the

order and step size values vary rather rapidly during the integration. (However, other quasi-Newton updates, which are applicable only in the small-system case, offer hope of dealing effectively with such rapid variations.) The most favorable results with updates seem to occur when the updated values of  $(\partial F_n/\partial y_n)^{-1}$  (actual or virtual) are as accurate (or produce the same speed of convergence) as those that would be gotten by reevaluating  $\partial F_n/\partial y_n$  from scratch, but are obtained at much lower cost. With the updates considered here, this fortunate situation seems most likely to occur when sizeable numbers of consecutive integration steps are taken over which the step sizes and Jacobian values change only relatively little. In a general purpose solver like LSODE, a natural approach to the design of the algorithm is to try to detect when such conditions hold and when they do not, and attempt to restrict the use of updates in a dynamic way accordingly. This idea has been a guiding principle in our work, but there is certainly more to be done towards that end.

## REFERENCES

- [1] P. ALFELD, *Two devices for improving the efficiency of stiff ODE solvers*, in Proc. 1979 SIGNUM Meeting on Numerical Ordinary Differential Equations, Univ. Illinois, Dept. Computer Science, Report 79-1710, Urbana, pp. 24-1 to 24-3.
- [2] P. N. BROWN, A. C. HINDMARSH AND H. F. WALKER, *Experiments with quasi-Newton methods in solving stiff ODE systems*, Lawrence Livermore National Laboratory Report UCRL-88470, December 1982.
- [3] C. G. BROYDEN, *A class of methods for solving nonlinear simultaneous equations*, Math. Comp., 19 (1965), pp. 577-593.
- [4] ———, *The convergence of an algorithm for solving sparse nonlinear systems*, Math. Comp., 25 (1971), pp. 285-294.
- [5] J. E. DENNIS, JR. AND E. S. MARWIL, *Direct secant updates of matrix factorizations*, Math. Comp., 38 (1982), pp. 459-474.
- [6] J. E. DENNIS, JR. AND J. J. MORÉ, *Quasi-Newton methods, motivation and theory*, SIAM Rev., 19 (1977), pp. 46-89.
- [7] J. E. DENNIS, JR. AND R. B. SCHNABEL, *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*, Prentice-Hall, Englewood Cliffs, NJ, 1983.
- [8] J. J. DONGARRA, J. R. BUNCH, C. B. MOLER AND G. W. STEWART, *LINPACK Users' Guide*, Society for Industrial and Applied Mathematics, Philadelphia, 1979.
- [9] M. S. ENGELMAN, *Quasi-Newton methods in fluid dynamics*, in Proc. 4th Conference on Mathematics of Finite Elements and Applications, Brunel University, Uxbridge, England, April 27-May 1, 1981.
- [10] M. S. ENGELMAN, G. STRANG AND K. J. BATHE, *The application of quasi-Newton methods in fluid dynamics*, Int. J. Numer. Meth. Eng., 17 (1981), pp. 707-718.
- [11] C. W. GEAR, *Numerical Initial Value Problems in Ordinary Differential Equations*, Prentice-Hall, Englewood Cliffs, NJ, 1971, pp. 158-166.
- [12] M. GERADIN, S. IDELSOHN AND M. HOGGE, *Computational strategies for the solution of large nonlinear problems via quasi-Newton methods*, Computers and Structures, 13 (1981), pp. 73-81.
- [13] A. C. HINDMARSH, *Linear multistep methods for ordinary differential equations: method formulations, stability, and the methods of Nordsieck and Gear*, Lawrence Livermore National Laboratory Report UCRL-51186, Rev. 1, March 1972.
- [14] ———, *LSODE and LSODI, two new initial value ordinary differential equation solvers*, in ACM SIGNUM Newsletter, vol. 15, no. 4 (December 1980), pp. 10-11.
- [15] ———, *ODE solvers for use with the method of lines*, in Advances in Computer Methods for Partial Differential Equations—IV, R. Vichnevetsky and R. S. Stepleman eds., IMACS, New Brunswick, NJ, 1981, pp. 312-316.
- [16] ———, *ODEPACK, a systematized collection of ODE solvers*, in Scientific Computing, R. S. Stepleman et al. eds., North-Holland, Amsterdam, 1983, pp. 53-64.
- [17] A. C. HINDMARSH AND G. D. BYRNE, *On the use of rank-one updates in the solution of stiff systems of ordinary differential equations*, ACM SIGNUM Newsletter, vol. 11, no. 3 (October 1976), pp. 23-27.



- [18] E. S. MARWIL, *Exploiting sparsity in Newton-like methods*, Ph.D. thesis, Cornell University, Ithaca, NY, 1978.
- [19] H. MATTHIES AND G. STRANG, *The solution of nonlinear finite-element equations*, Int. J. Numer. Meth. Eng., 14 (1979), pp. 1613–1626.
- [20] J. M. ORTEGA AND W. C. RHEINOLDT, *Iterative Solution of Nonlinear Equations in Several Variables*, Academic Press, New York, 1970.
- [21] L. K. SCHUBERT, *Modification of a quasi-Newton method for nonlinear equations with a sparse Jacobian*, Math. Comp., 24 (1970), pp. 27–30.
- [22] L. F. SHAMPINE AND C. W. GEAR, *A user's view of solving stiff ordinary differential equations*, SIAM Rev., 21 (1979), pp. 1–17.
- [23] PH. L. TOINT, *On sparse and symmetric matrix updating subject to a linear equation*, Math. Comp., 31 (1977), pp. 954–961.
- [24] R. A. WILLOUGHBY ed., *Stiff Differential Systems*, Plenum Press, New York, 1974.

## TWO STRATEGIES FOR ROOT FINDING ON MULTIPROCESSOR SYSTEMS\*

I. NORMAN KATZ† AND MARK A. FRANKLIN‡

**Abstract.** The problem of finding the roots of a single nonlinear equation on an MIMD computer system is considered. Two globally convergent strategies for exploiting the parallelism in the computer system are investigated and compared. In one strategy a serial root finding algorithm is used, with the parallel computation capabilities of the system being used to speed up the necessary function evaluations. For a given function this increase in speed is determined by a known speed-up function  $S_p$  which relates the number of processors available to function evaluation time. Linear and logarithmic speed-up functions are investigated. In the second strategy a parallel root finding algorithm is used. Function evaluations are performed serially, one function evaluation being performed on each processor. Two new parallel root finding algorithms based on contraction mapping and Halley's method are presented and their convergence properties examined. Expressions for comparing the two strategies are derived and decision curves presented.

**Key words.** parallel numerical methods, parallel processing, root finding, multiprocessor systems

**1. Introduction.** Recent advances in computer technology have made it practical to construct systems in which many computers can perform different computations simultaneously, and then communicate the results to all (or some) of the other computers. The increased computational power provided by such systems operating in parallel may make it possible to solve large scale engineering and scientific problems which, because of their complexity, were considered intractable until now. In order to fully utilize these computer systems it is necessary, of course, to properly exploit the parallelism of computation.

This paper investigates the problem of finding the roots of a single nonlinear equation on a computer system of the MIMD (multiple instruction stream-multiple data stream) type. Two strategies for applying parallel computations are studied. In one strategy a serial root finding algorithm is used, with the parallel computation capabilities of the system being used to speed up the necessary function evaluations. For a given function this increase in speed is determined by a known speed-up function  $S_p$  ( $S_p$  is the ratio of the function evaluation speed in the single computer case to the evaluation speed in the multiple computer case). We refer to this strategy as SAPE (Serial Algorithm Parallel function Evaluation). In the second strategy, each function is evaluated on a separate computer, but the root finding algorithm is parallel in nature, i.e., the algorithm is such that different function evaluations can be performed simultaneously on the separate computers. We call this strategy PASE (Parallel Algorithm Serial Function Evaluation). In order to have an unbiased comparison, the same root-finding algorithm is used in both strategies; in the first strategy it is applied serially, and in the second in a parallel manner. Both strategies result in globally convergent algorithms.

Parallel algorithms for root finding have been considered by several authors e.g. [1]-[11] in the sense of the PASE strategy. No comparisons have been made, however, between PASE and SAPE strategies. The parallel algorithms used in our PASE strategy are conceptually similar to some algorithms presented in [10], [11]. However, we use

---

\* Received by the editors July 15, 1981, and in final revised form December 8, 1983. This work was supported in part by the National Science Foundation under grant MCS78-20731.

† Department of Systems Science and Mathematics, Washington University, St. Louis, Missouri 63130.

‡ Department of Electrical Engineering, Washington University, St. Louis, Missouri 63130.

the iteration functions for a contraction mapping and for Halley's method instead of the various iterations used in [10]. The analytic forms of the iteration functions used here lead to an error analysis which makes it possible to compare the relative efficiencies of the two strategies.

Both the SAPE and PASE strategies combine a direct search procedure and an iteration scheme (contraction mapping or Halley's method). The search procedure has the advantage that it converges even when the initial guesses are far from the root (provided that they bracket the root), and the iteration scheme has the advantage that in the neighborhood of the root, convergence becomes very rapid. By combining the two procedures it is possible to gain the advantages of both approaches.

The two general strategies presented here have been considered in [12] for the problem of one-dimensional optimization of a unimodal function on a multiprocessor system. However there two search strategies are compared: a serial Fibonacci search with parallelism used in function evaluation, and a  $n$ -ary parallel search with parallelism implicit in the search algorithm but no parallelism in function evaluation.

We begin by defining the PASE and SAPE strategies and by describing a property for higher order iteration functions which assures continued bracketing of a root once it has been bracketed initially. Then we analyze the behavior of the two strategies. Using this analysis we show how to determine the relative efficiencies of the two strategies. Sample decision curves are then given. If the values of certain parameters are known, these curves can be used to decide which strategy is more efficient.

**2. The strategies.** Suppose we wish to solve

$$f(x) = 0$$

where  $f: R^1 \rightarrow R^1$ , by an iterative method. We combine a search procedure together with an iterative procedure of the form

$$(1) \quad x^{(m+1)} = g(x^{(m)}), \quad m = 0, 1, 2, \dots$$

where  $g(x)$  is a suitably formulated iteration function such that  $\alpha = g(\alpha)$  if and only if  $f(\alpha) = 0$ . In this paper we assume that the root  $\alpha$  is simple, that is  $f'(\alpha) \neq 0$ .

Assume that  $N$  computers are available in an MIMD system in which there is negligible overhead associated with transferring information between computers. Associated with parallel function evaluation procedures there is a speed-up function,  $S_p(N)$ . If  $t_p(N)$  is the time it takes to evaluate the function with  $N$  computers, then the speed-up function is defined as:  $S_p(N) = t_p(1)/t_p(N)$ . Later we consider two particular forms for  $S_p(N)$ : linear speed-up and logarithmic speed-up.

Assume that  $\alpha \in [a, b]$  and let  $a^1 = g(a)$ ,  $b^1 = g(b)$ . In order to assure that the root  $\alpha$  continues to be bracketed after iteration (i.e.  $\alpha \in [a^1, b^1]$ ) we require the following property for  $x, y$  in a sufficiently small neighborhood of  $\alpha$ :

$$(P) \quad (g(x) - \alpha)(g(y) - \alpha) < 0 \quad \text{if } (x - \alpha)(y - \alpha) < 0.$$

For an iteration function  $g(x)$  of integral order  $n > 0$  it is well-known (see [13]) that for  $x$  sufficiently close to  $\alpha$

$$(2) \quad g(x) - \alpha \approx \lambda_n(x - \alpha)^n$$

where  $\lambda_n$  is independent of  $x$ . In order for the property (P) to be satisfied  $n$  must be odd. In particular, Newton's method for which  $n = 2$  (see [13]) does *not* have property (P). We consider the cases  $n = 1, 3$ . The case  $n = 1$  is that of a contraction mapping. For the case  $n = 3$  there are many schemes that are known (see [14]). Here we choose Halley's method (see [14]) to illustrate our strategy computationally, but we also give formulas for the methods of Chebyshev and Cauchy (see [14]).

Our method consists of combining a search procedure with the iterative procedure given in (1). Parallelism is used in one of two ways. In the first way the combined search-iteration is implemented in a serial algorithm, but the functional evaluations required by the algorithm at each step are performed in parallel. We call this approach SAPE (Serial Algorithm-Parallel function Evaluation). In the second way the combined search-iteration is implemented in a parallel algorithm, but the functional evaluations are performed serially. We call this approach PASE (Parallel Algorithm-Serial functional Evaluation). Assume that  $a < b$ ,  $\alpha \in (a, b)$ .

### 2.1. The SAPE strategy.

( $n = 1$ ) Define  $r(x) = x - g(x)$ , suppose that  $g(a)$ ,  $r(a)$ ,  $g(b)$ ,  $r(b)$  have been computed and that  $r(a)r(b) < 0$ .

*Step 1.* Compute  $x = \frac{1}{2}(a + b)$ ,  $g(x)$ ,  $r(x)$   
then if  $r(a)r(x) < 0$  let  $a_0 = a$ ,  $b_0 = x$   
otherwise let  $a_0 = x$ ,  $b_0 = b$ .

*Step 2.* Compute  $g(g(a_0))$ ,  $g(g(b_0))$ ,  $r(g(a_0))$ ,  $r(g(b_0))$   
then if  $r(g(a_0))r(g(b_0)) > 0$  set  $a = a_0$ ,  $b = b_0$   
if  $r(g(a_0))r(g(b_0)) < 0$  let  $a_1 = \min(g(a_0), g(b_0))$   
 $b_1 = \max(g(a_0), g(b_0))$   
if  $b_1 - a_1 < b_0 - a_0$  set  $a = a_1$ ,  $b = b_1$   
otherwise set  $a = a_0$ ,  $b = b_0$ .

Step 1 requires one evaluation of the function  $g(x)$ , and step 2 requires two functional evaluations of  $g(x)$  (one at  $g(a_0)$  and the other at  $g(b_0)$ ).

( $n = 3$ ) In Halley's method the iteration function (see [14])

$$(2a) \quad g(x) = x - \left[ \frac{f/f'}{1 - (f/f')(f''/2f')} \right]_x$$

requires evaluation of  $f$ ,  $f'$ , and  $f''$ . Suppose that  $f(a)$ ,  $f(b)$  have been computed and that  $f(a)f(b) < 0$ .

*Step 1.* Compute  $x = \frac{1}{2}(a + b)$ ,  $f(x)$   
then if  $f(a)f(x) < 0$  let  $a_0 = a$ ,  $b_0 = x$   
otherwise let  $a_0 = x$ ,  $b_0 = b$ .  
*Step 2.* Compute  $f'(a_0)$ ,  $f''(a_0)$ ,  $g(a_0)$ ,  $f(g(a_0))$ ,  $f'(b_0)$ ,  $f''(b_0)$ ,  $g(b_0)$ ,  $f(g(b_0))$   
then if  $f(g(a_0))f(g(b_0)) > 0$  set  $a = a_0$ ,  $b = b_0$   
if  $f(g(a_0))f(g(b_0)) < 0$  let  $a_1 = \min(g(a_0), g(b_0))$   
 $b_1 = \max(g(a_0), g(b_0))$   
if  $b_1 - a_1 < b_0 - a_0$  set  $a = a_1$ ,  $b = b_1$   
otherwise set  $a = a_0$ ,  $b = b_0$ .

Step 1 requires 1 evaluation of the function  $f(x)$ , and step 2 requires 6 functional evaluations ( $f', f''$  at  $a_0$  and  $b_0$ ,  $f$  at  $g(a_0)$  and  $g(b_0)$ ). The procedures described for the SAPE strategy are globally convergent, because  $r(a)r(b) < 0$  when  $n = 1$ ,  $f(a)f(b) < 0$  when  $n = 3$  at every iteration. In the SAPE strategy the functional evaluations are performed sequentially, but each functional evaluation is implemented in parallel on  $N$  computers.

### 2.2. The PASE strategy.

( $n = 1$ ) Define  $r(x) = x - g(x)$ , suppose that  $g(a)$ ,  $r(a)$ ,  $g(b)$ ,  $r(b)$  have been computed, and that  $r(a)r(b) < 0$ . Assume that  $N \geq 3$ . Let  $h = (b - a)/(N - 1)$ ,  $x_1^{(0)} = a$ ,  $x_j^{(0)} = a + (j - 1)h$  for  $j = 2, \dots, N - 1$ ,  $x_N^{(0)} = b$ .

*Step 1.* Simultaneously compute  $g(x_j^{(0)})$ ,  $r(x_j^{(0)})$  for  $j = 2, \dots, N - 1$  and  $g(g(a))$ ,  $r(g(a))$ ,  $g(g(b))$ ,  $r(g(b))$  on the  $N$  parallel processors. There is at least one pair such that  $r(x_j^{(0)})r(x_{j+1}^{(0)}) < 0$  for some  $j = 1, \dots, N - 1$ . Choose the pair such that either  $|r(x_j^{(0)})|$  or  $|r(x_{j+1}^{(0)})|$  is smallest over all  $|r(x_i^{(0)})|$ ,  $i = 1, \dots, N - 1$ . Let  $a_0 = x_j^{(0)}$ ,  $b_0 = x_{j+1}^{(0)}$ .

*Step 2.* if  $r(g(a))r(g(b)) > 0$  set  $a = a_0, b = b_0$   
 if  $r(g(a))r(g(b)) < 0$  let  $a_1 = \min(g(a), g(b))$   
 $b_1 = \max(g(a), g(b))$   
 then if  $b_1 - a_1 < b_0 - a_0$  set  $a = a_1, b = b_1$   
 otherwise set  $a = a_0, b = b_0$ .

In step 1, the  $N$  functional evaluations  $g(x_j^{(0)})$  for  $j = 2, \dots, N - 1$ ,  $g(g(a))$ , and  $g(g(b))$ , are performed in parallel. It is assumed that the time needed for computing  $r(x)$  is negligible compared to the time needed to compute  $g(x)$ .

( $n = 3$ ) We again use Halley's iteration function  $g(x)$  given in (3). Assume that  $N \geq 4$ . Suppose that  $f(a), f(b)$  have been computed and that  $f(a)f(b) < 0$ . Let  $h = (b - a)/(2N - 5)$ ,  $x_1^{(0)} = a$ ,  $x_j^{(0)} = a + (j - 1)h$  for  $j = 2, \dots, 2N - 5$ ,  $x_{2N-4}^{(0)} = b$ .

*Step 1.1.* Simultaneously compute  $f(x_j^{(0)})$  for  $j = 2, \dots, N - 3$ , and  $f'(a)$ ,  $f''(a)$ ,  $f'(b)$ ,  $f''(b)$  needed for  $g(a), g(b)$ .

*Step 1.2.* Then simultaneously compute  $f(x_j^{(0)})$  for  $j = N - 2, \dots, 2N - 5$  and  $f(g(a))$ ,  $f(g(b))$ . There is at least one pair such that  $f(x_j^{(0)})f(x_{j+1}^{(0)}) < 0$  for some  $j = 1, \dots, 2N - 5$ . Choose the pair such that either  $|f(x_j^{(0)})|$  or  $|f(x_{j+1}^{(0)})|$  is smallest over all  $|f(x_i^{(0)})|$ ,  $i = 1, \dots, 2N - 5$ . Let  $a_0 = x_j^{(0)}$ ,  $b_0 = x_{j+1}^{(0)}$ .

*Step 2.* if  $f(g(a))f(g(b)) > 0$  set  $a = a_0, b = b_0$   
 if  $f(g(a))f(g(b)) < 0$  let  $a_1 = \min(g(a), g(b))$   
 $b_1 = \max(g(a), g(b))$   
 then if  $b_1 - a_1 < b_0 - a_0$  set  $a = a_1, b = b_1$   
 otherwise set  $a = a_0, b = b_0$ .

It is necessary to divide step 1 into two stages because  $f(g(x))$  cannot be calculated at the same time as  $f(x)$ ,  $f'(x)$ ,  $f''(x)$ . We therefore make use of both stages for the search procedure. In step 1.1 the  $N$  functional evaluations  $f(x_j^{(0)})$  for  $j = 2, \dots, N - 3$ ,  $f'(a)$ ,  $f''(a)$ ,  $f'(b)$ ,  $f''(b)$  are performed in parallel. In step 1.2 the  $N$  functional evaluations  $f(x_j^{(0)})$  for  $j = N - 2, \dots, 2N - 5$ ,  $f(g(a))$ ,  $f(g(b))$  are performed in parallel. Arithmetic operations in the calculation of  $g(x)$  are assumed to be negligible in comparison with a functional evaluation. The procedures described for the PASE strategy are globally convergent because  $r(a)r(b) < 0$  when  $n = 1$ ,  $f(a)f(b) < 0$  when  $n = 3$  at every iteration.

To summarize, both SAPE and PASE strategies combine an iteration procedure performed at the end points  $a$  and  $b$  with a search procedure performed at interior points. Both strategies are globally convergent. The search procedure guarantees global convergence for both strategies whereas the iteration procedure leads to rapid convergence locally when the bracketing interval  $(a, b)$  becomes sufficiently small.

Next, we consider the rates of convergence of SAPE and PASE. This leads to a comparison of the relative efficiencies of the two strategies.

**3. Convergence analysis.** Suppose  $\alpha \in (a, b)$  and  $f(a)f(b) < 0$ . We wish to determine a final interval  $(a_f, b_f)$  such that  $\alpha \in (a_f, b_f)$ ,  $f(a_f)f(b_f) < 0$  and  $b_f - a_f \leq 10^{-d}$ , where  $d$  is a prescribed precision. Given an interval  $(a, b)$  the next interval is determined either by the search procedure or by the iteration function  $g(x)$ . The next interval is

determined to be  $(a_1, b_1)$  by the iteration function  $g(x)$  if  $b_1 - a_1 < b_0 - a_0$  and if  $f(g(a))f(g(b)) < 0$ . This *must* occur if property (P) holds, if  $(a, b)$  is sufficiently small so that the local convergence properties of  $g(x)$  are valid, and if the reduction of the interval  $(a, b)$  determined by these local properties is smaller than the reduction determined by the search procedure. In our analysis we assume that these are the *only* circumstances in which  $g(x)$  determines the next interval; otherwise the next interval is determined by the search procedure. This means that for purposes of our analysis the iteration scheme is assumed to determine the next interval only when the bracketing interval is small enough. Also once the iteration scheme does determine the next interval, it continues to do so until the desired accuracy is achieved. There are, of course, cases where our assumptions will not hold, but these are assumed not to commonly occur.

**3.1. Analysis.** Initially  $(a, b)$  may be so large that the local convergence properties of the iteration function  $g(x)$  do not apply. Since the search procedure reduces the size of  $(a, b)$ , eventually these properties *must* apply. Observe that because of property (P) both the search procedure and the iteration function  $g(x)$  result in the next interval bracketing  $\alpha$ .

First we analyze the behavior of the PASE strategy. Let

$m_1$  = number of iterations in which the next interval is determined by the search procedure.

$m_2$  = number of iterations in which the next interval is determined by the iteration function  $g(x)$ .

If (2) is expanded to include another term, it becomes

$$\begin{aligned} g(x) - \alpha &= \lambda_n(x - \alpha)^n + \mu_n(x - \alpha)^{n+1} + \dots \\ &= \lambda_n(x - \alpha)^n \left( 1 + \frac{\mu_n}{\lambda_n}(x - \alpha) + \dots \right) \\ &\approx \lambda_n(x - \alpha)^n \left( 1 + \frac{\mu_n}{\lambda_n}(x - \alpha) \right). \end{aligned}$$

We define the range of effectiveness of  $g(x)$  (i.e. the interval in which local convergence properties hold) from

$$\left| \frac{\mu_n}{\lambda_n}(x - \alpha) \right| \ll 1,$$

or more specifically by

$$(3) \quad \left| \frac{\mu_n}{\lambda_n}(x - \alpha) \right| \leq 10^{-d_1},$$

where  $d_1$  is a prescribed precision,  $d_1 < d$ .

For any interval  $(a, b)$  the search procedure determines the next interval when the following conditions are met:

$$b - a > 10^{-d} \quad (\text{required accuracy is not met})$$

and at least one of the following hold:

$$b - a > |\lambda_n/\mu_n|10^{-d_1} \quad (g(x) \text{ is not in its range of effectiveness})$$

$$|\lambda_n|(b - a)^n > (b - a)/M_n \quad (g(x) \text{ is less effective than the search})$$

where

$$M_n = \begin{cases} N-1 & \text{if } n = 1, \\ 2N-5 & \text{if } n = 3. \end{cases}$$

Now let  $(a, b)$  be the initial interval. The search procedure continues until  $m_1$  is such that

$$(4) \quad \left(\frac{1}{M_n}\right)^{m_1} (b-a) \leq 10^{-d}$$

or both of the following hold:

$$(5) \quad \left(\frac{1}{M_n}\right)^{m_1} (b-a) \leq \left|\frac{\lambda_n}{\mu_n}\right| 10^{-d_1}$$

and

$$(6) \quad |\lambda_n| \left[ \left(\frac{1}{M_n}\right)^{m_1} (b-a) \right]^n < \frac{1}{M_n} \left[ \left(\frac{1}{M_n}\right)^{m_1} (b-a) \right].$$

From (2) it follows that when  $|x^{(0)} - \alpha| \leq |\lambda_n/\mu_n| 10^{-d_1}$ , then for  $\bar{m} \geq 1$

$$\begin{aligned} x^{(\bar{m})} - \alpha &= g(x^{(\bar{m}-1)}) - \alpha \approx \lambda_n (x^{(\bar{m}-1)} - \alpha)^n \approx \lambda_n (\lambda_n (x^{(\bar{m}-2)} - \alpha)^n)^n \\ &\approx \lambda_n^{1+n} (x^{(\bar{m}-2)} - \alpha)^{n^2} \approx \lambda_n^{1+n+n^2} (x^{(\bar{m}-3)} - \alpha)^{n^3} \\ &\approx \lambda^{1+n+n^2+\dots+n^{\bar{m}-1}} (x^{(0)} - \alpha)^{n^{\bar{m}}} \\ &= \begin{cases} \lambda \frac{3^{\bar{m}} - 1}{2} (x^{(0)} - \alpha)^{3^{\bar{m}}} & \text{if } n = 3, \\ \lambda^{\bar{m}} (x^{(0)} - \alpha) & \text{if } n = 1. \end{cases} \end{aligned}$$

Therefore if (4) does not hold and (5) and (6) do, then  $g(x)$  becomes effective in determining the next interval. Then, since

$$|x^{(0)} - \alpha| \leq (1/M_n)^{m_1} (b-a),$$

$m_2$  is such that

$$(7) \quad |\lambda_1|^{m_2} (1/(N-1))^{m_1} (b-a) \leq 10^{-d} \quad \text{if } n = 1,$$

$$(8) \quad |\lambda_3|^{(3^{m_2}-1)/2} (1/(2N-5))^{m_1} (b-a)^{3^{m_2}} \leq 10^{-d} \quad \text{if } n = 3.$$

Let  $m = m_1 + m_2$ , the total number of iterations required to satisfy the prescribed accuracy, and assume that  $|b-a| > 10^{-d}$ ; otherwise, stop.

It now follows from (6) that for the contraction mapping ( $n = 1$ ) if  $|\lambda_1| \geq 1/(N-1)$  then  $g(x)$  is never effective (independent of (5)) so  $m_2 = 0$ , and from (4) we have

$$m = m_1 = \frac{1}{\log(N-1)} (d + \log(b-a)).$$

If  $|\lambda_1| < 1/(N-1)$ , then  $m_1$  is determined from (5), and  $m_2$  is determined from (7). This leads to the following estimates for  $m$ .

( $n = 1$ ), contraction mapping, PASE strategy:

if  $|\lambda_1| \geq 1/(N-1)$ , then

$$m_1(\text{PASE}) = \frac{1}{\log(N-1)} (d + \log(b-a)),$$

$$m(\text{PASE}) = m_1(\text{PASE});$$

if  $|\lambda_1| < 1/(N-1)$ , then

$$m_1(\text{PASE}) = \frac{1}{\log(N-1)} \max \{d_1 - \log |\lambda_1/\mu_1| + \log(b-a), 0\},$$

$$(9) \quad m_2(\text{PASE}) = \frac{1}{(\log(1/|\lambda_1|))} \max \{d + \log(b-a) - m_1(\text{PASE}) \log(N-1), 0\},$$

$$m(\text{PASE}) = m_1(\text{PASE}) + m_2(\text{PASE}).$$

Similarly for the third order iteration ( $n=3$ )  $m_1$  is determined from (5) and (6), and  $m_2$  is then determined from (8). This leads to the following estimates for  $m$  ( $n=3$ ), third order iteration, PASE strategy:

$$m_1(\text{PASE}) = \frac{1}{\log(2N-5)} \max \{d_1 - \log |\lambda_3/\mu_3| + \log(b-a), 0, \frac{1}{2} \log |\lambda_3(2N-5)| + \log(b-a)\},$$

$$(10) \quad m_2(\text{PASE}) = \frac{1}{\log 3} \max \left\{ \log \left[ \frac{\log |\lambda_3|^{1/2} - d}{\log |\lambda_3|^{1/2} - m_1(\text{PASE}) \log(2N-5) + \log(b-a)} \right], 0 \right\},$$

$$m(\text{PASE}) = m_1(\text{PASE}) + m_2(\text{PASE}).$$

All logs are to the base 10.

In the SAPE strategy the search strategy consists of dividing the interval  $b-a$  by 2. The discussion above remains valid if  $N-1$  is replaced by 2 for  $n=1$ , and  $2N-5$  is replaced by 2 for  $n=3$ . This gives the following estimates for  $m$ .

( $n=1$ ), contraction mapping, SAPE strategy:

if  $|\lambda_1| \geq \frac{1}{2}$ , then

$$m_1(\text{SAPE}) = \frac{1}{\log 2} (d + \log(b-a)),$$

$$m(\text{SAPE}) = m_1(\text{SAPE});$$

if  $|\lambda_1| < \frac{1}{2}$ , then

$$m_1(\text{SAPE}) = \frac{1}{\log 2} \max \{d_1 - \log |\lambda_1/\mu_1| + \log(b-a), 0\},$$

$$(11) \quad m_2(\text{SAPE}) = \frac{1}{\log(1/|\lambda_1|)} \max \{d + \log(b-a) - m_1(\text{SAPE}) \log 2, 0\},$$

$$m(\text{SAPE}) = m_1(\text{SAPE}) + m_2(\text{SAPE}).$$

( $n=3$ ), third order iteration, SAPE strategy:

$$m_1(\text{SAPE}) = \frac{1}{\log 2} \max \{d_1 - \log |\lambda_3/\mu_3| + \log(b-a), 0, \frac{1}{2} \log |2\lambda_3| + \log(b-a)\},$$

$$(12) \quad m_2(\text{SAPE}) = \frac{1}{\log 3} \max \left\{ \log \left[ \frac{\log |\lambda_3|^{1/2} - d}{\log |\lambda_3|^{1/2} - m_1(\text{SAPE}) \log 2 + \log(b-a)} \right], 0 \right\},$$

$$m(\text{SAPE}) = m_1(\text{SAPE}) + m_2(\text{SAPE}).$$

All logs are to the base 10.



Two points regarding the use of the expressions in (9)-(12) require further discussion. First, the number  $d_1$  in (5) is used to determine whether the iteration function  $g(x)$  is in its range of effectiveness i.e. whether the second term in (3) is small compared to the first term. We have determined empirically through testing that  $d_1 = .3$  is a satisfactory value for the contraction mapping. In the case of Halley's method the range of effectiveness is smaller and we use either  $d_1 = 1$  or  $d_1 = 2$ . Second, the factors  $\lambda_n$  and  $\mu_n$  in (3) must be given. When  $n = 1$ , a Taylor series expansion easily shows that

$$\lambda_1 = g'(\alpha), \quad \mu_1 = g''(\alpha)/2.$$

When  $n = 3$ ,  $\lambda_3$  is a standard quantity given, for example, in [14] for various third order iteration functions.  $\mu_3$  is not given however. In the Appendix we derive  $\mu_3$  for three third order methods, those of Halley, Chebyshev, and Cauchy (see [14]). In our examples we have used only Halley's method, for which

$$\lambda_3 = [(f''/2f')^2 - (f'''/6f')]\alpha,$$

$$\mu_3 = [-(f^{iv}/8f') + (f''f'''/2f'^2) - (3f''/8f'^3)]\alpha.$$

**3.2. Numerical examples.** In order to test the accuracy estimates and convergence properties for the PASE algorithms presented in the previous sections, several nonlinear equations were solved and analyzed. The equations are:

$$f_1(x) = x - 2 + e^x = 0 \quad \text{(taken from [15]),}$$

$$f_2(x) = 5x - e^x - 3 = 0 \quad \text{(taken from [15]),}$$

$$f_3(x) = .5 + .5 e^{-x/2\pi} \sin x - D = 0 \quad \text{(taken from [16]).}$$

$f_1(x)$  has a unique root  $\alpha = .4428544010$ .  $f_2(x)$  has two roots  $\alpha_1 = 1.46882$ ,  $\alpha_2 = 1.74375189$ . For  $D = .75$   $f_3(x)$  has a root  $\alpha = .5804383646$ . When applying Halley's method the functions  $f_i(x)$  themselves are used in the definition of  $g_i(x)$ . However when applying the contraction mapping it is necessary to reformulate each equation in such a way that  $|\lambda_1| = |g'(\alpha)|$  satisfies  $0 < |\lambda_1| < 1$ . For each equation the definition of  $g_1(x)$ , and the numbers  $\lambda_1, \mu_1, \lambda_3, \mu_3$  are:

$g_1(x) = \log(2 - x),$	$\lambda_1 = .642, \mu_1 = .206,$	
	$\lambda_3 = -.008, \mu_3 = .025,$	
$g_2(x) = \frac{1}{5}(e^x + 3),$	$\lambda_1 = .869, \mu_1 = .434$	}
	$\lambda_3 = 12.07, \mu_3 = 131.7$	
$g_2(x) = \log(5x - 3),$	$\lambda_1 = .874, \mu_1 = .382$	}
	$\lambda_3 = 14.5, \mu_3 = -158.2$	
$g_3(x) = \arcsin(e^{x/2\pi}(2D - 1)),$	$\lambda_1 = .104, \mu_1 = .012$ for $D = .75,$	
	$\lambda_3 = .400, \mu_3 = .961.$	

For each of the two strategies, PASE and SAPE, and for each iteration function,  $n = 1$  and  $n = 3$ , we compared predicted values for  $m_1$  (the number of iterations determined by the search procedure) and  $m_2$  (the number of iterations determined by the iteration function  $g(x)$ ) with actual values. The predicted values are given by (9)-(12). For Halley's method, the range of effectiveness was smaller than for the contraction mapping.  $d_1$  was taken to be .3 as determined through extensive testing for the contraction mapping;  $d_1 = 1$  or  $d_1 = 2$  gives acceptable predictions for most cases when Halley's method is used.

TABLE 1  
Contraction mapping ( $n = 1$ ).

function	PASE STRATEGY					SAPE STRATEGY						
	N	initial interval	predicted $m_1$	actual $m_1$	predicted $m_2$	actual $m_2$	predicted $m_1$	actual $m_1$	predicted $m_2$	actual $m_2$	predicted $m$	actual $m$
$f_1(x)$	5	a=0.0, b=1.0	24.9	25	0	0	49.8	50	0	0	49.8	50
	7		19.2	20	0	0						
	9		16.6	17	0	0						
	11		15.1	15	0	0						
	13		13.9	14	0	0						
15		13.1	14	0	0							
$f_2(x)$ smaller root	5	a=0.0, b=1.7	25.3	26	0	0	50.6	51	0	0	50.6	50
	7		19.6	20	0	0						
	9		16.9	17	0	0						
	11		15.2	16	0	0						
	13		14.1	15	0	0						
15		13.3	14	0	0							
$f_2(x)$ larger root	5	a=1.5, b=2.4	48.8	49	0	0	48.8	49	0	0	48.8	49
	7		16.3	17	0	0						
	9		14.7	15	0	0						
	11		13.6	14	0	0						
	13		12.8	13	0	0						
$f_3(x)$	5	a=-4.3, b=4.3	24.4	25	0	0	.9	0	15.9	17	16.9	17
	7		18.9	19	0	0						
	9		16.3	17	0	0						
	11		14.7	15	0	0						
	13		13.6	14	0	0						
15		12.8	13	0	0							
$f_3(x)$	5	a=-4.3, b=4.3	.5	0	15.9	16						
	7		.4	0	15.9	16						
	9		.3	1	15.9	16						
	11		15.9	16	0	0						
	13		14.7	15	0	0						
15		13.9	14	0	0							

$N$  = number of processors  
 $m_1$  = number of iterations in which the search procedure determines the iterate  
 $m_2$  = number of iterations in which the contraction mapping determines the iterate  
 $m = m_1 + m_2$  = total number of iterations such that  
 $b_f - a_d \leq 10^{-d}$ ,  $d = 15$   
 $d_1$  determines range of effectiveness of contraction mapping [see equation (3)]  
 $|x - \alpha| \leq |\lambda_1/\mu_1| 10^{-d_1}$ ,  $d_1 = .3$

Table 1 compares PASE and SAPE for the contraction mapping and Table 2 does the same for Halley's method. In Table 1 the predicted values are very close to the actual values. In Table 2 predicted and actual values are close for the PASE strategy, and in most cases for the SAPE strategy. In one case, when the larger root for  $f_2(x)$  is solved for by SAPE, our assumption that the higher order iteration remains effective in determining the next iterate once it starts becoming effective, is violated.

**4. Comparison of strategies.**

**4.1. Computation times.** Denote by  $T_S$  the total time needed to compute the root  $\alpha$  using the SAPE strategy, in which the root finding algorithm is serial while the evaluation of  $g(x)$  is done in parallel with speed-up  $S_p(N)$ . Denote by  $T_P$  the total time needed using the PASE strategy, in which the root finding algorithm is parallel while the evaluation of  $g(x)$  is done serially on each of  $N$  processors.

Then we have

$$(13) \quad T_S = m(\text{SAPE})(t_p(N)m_F(\text{SAPE}) + t'_S),$$

$$(14) \quad T_P = m(\text{PASE})(t_p(1)m_F(\text{PASE}) + t'_P)$$

where

- $t_p(N)$  is the time that it takes to perform a function evaluation in parallel on  $N$  processors.
- $t'_S$  is the time that it takes to perform tests for convergence and for communication using the SAPE strategy.
- $t'_P$  is the time that it takes to compare residuals, perform tests for convergence and for communication using the PASE strategy.
- $m_F(\text{SAPE})$  is the number of functional evaluations per iteration using the SAPE strategy.
- $m_F(\text{PASE})$  is the number of steps which must be performed sequentially using the PASE strategy.

We assume that all functional evaluations are of comparable complexity and that they have similar speed-up characteristics. We also assume that  $t_p(N) \gg t'_S$  and  $t_p(1) \gg t'_P$ . Then since by definition  $S_p(N) = t_p(1)/(t_p(N))$ , we have

$$(15) \quad T_S = m(\text{SAPE})m_F(\text{SAPE})t_p(N) = m(\text{SAPE})m_F(\text{SAPE})t_p(1)/S_p(N)$$

$$(16) \quad T_P = m(\text{PASE})m_F(\text{PASE})t_p(1).$$

**4.2. Speed-up functions and decision regions.** Generally, for each iteration function (contraction mapping or Halley's method), there will be certain regions where the SAPE strategy will be advantageous, and certain regions where the PASE strategy will be best. In general this will depend on the error tolerance allowed, the number of computers available, the speed-up function, the characteristics of the nonlinear function itself (i.e.,  $\lambda_1, \mu_1$  and  $\lambda_3, \mu_3$ ), and the length of the initial interval  $(b - a)$ .

Consider next the speed-up function  $S_p(N)$ . This function will depend on the form of  $g(x)$  and on the parallel methods used to evaluate this function. For demonstration purposes we examine two sample  $S_p(N)$  functions, the first linear, and the second logarithmic. Both forms have been encountered in empirical studies involving parallel solution to various systems of equations ([17], [18], [19]). For linear speed-up:

$$(17) \quad S_p(N) = S_{pli}(N) = 1 + A(N - 1).$$

For logarithmic speed-up:

$$(18) \quad S_p(N) = S_{plo}(N) = 1 + B \log_{10} N.$$

TABLE 2  
Halley's method (n=3).

function	N	initial interval	PASE STRATEGY			SAPE STRATEGY								
			predicted $m_1$	actual $m_1$	predicted $m_2$	actual $m_2$	predicted $m_1$	actual $m_1$	predicted $m_2$	actual $m_2$				
$f_1(x)$ $d_1=1$	5	a=0.0, b=1.0	2.1	1	1.5	2	3.6	6	6	1.7	2	6.5	8	
	7	a=0.0, b=1.0	1.5	1	1.7	2	3.2	6	6	1.4	2	9.5	8	
	9	a=0.0, b=1.0	1.3	1	1.8	2	3.0	6	6	1.4	2	9.5	8	
	11	a=0.0, b=1.0	1.2	1	1.8	2	2.9	7	7	2.1	4	9.5	11	
	13	a=0.0, b=1.0	1.1	1	1.8	2	2.9	7	7	1.6	4	12.4	11	
	15	a=0.0, b=1.0	1.0	1	1.9	2	2.9	7	7	1.6	4	12.4	11	
	5	a=0.0, b=1.0	3.5	1	1.2	2	4.7	3	3					
	7	a=0.0, b=1.0	2.6	1	1.4	2	3.9	3	3					
	9	a=0.0, b=1.0	2.2	1	1.6	2	3.7	3	3					
	11	a=0.0, b=1.0	1.9	1	1.5	2	3.5	3	3					
	13	a=0.0, b=1.0	1.9	1	1.6	2	3.4	3	3					
	15	a=0.0, b=1.0	1.8	1	1.6	2	3.4	3	3					
	$f_2(x)$ smaller root	5	a=0.0, b=1.6	3.2	6	1.6	2	4.8	6	6	2.1	4	9.5	11
		7	a=0.0, b=1.6	2.4	2	2.0	3	4.4	0	0	2.1	5	8.6	5
		9	a=0.0, b=1.6	2.0	1	2.2	3	4.2	0	0	1.6	5	11.5	5
11		a=0.0, b=1.6	1.8	1	2.4	3	4.2	0	0	1.6	5	11.5	5	
13		a=0.0, b=1.6	1.7	2	2.5	4	4.2	0	0	1.6	5	11.5	5	
15		a=0.0, b=1.6	1.6	3	2.5	2	4.2	0	0	1.6	5	11.5	5	
5		a=0.0, b=1.6	4.6	6	1.2	2	5.8	12*	12*	2.9	3*	7.9	15	
7		a=0.0, b=1.6	3.4	2	1.5	3	4.9	12*	12*	1.6	3*	10.8	15	
9		a=0.0, b=1.6	2.9	1	1.7	3	4.6	12*	12*	1.6	3*	10.8	15	
11		a=0.0, b=1.6	2.6	1	1.9	3	4.5	12*	12*	1.6	3*	10.8	15	
13		a=0.0, b=1.6	2.4	2	1.9	2	4.4	12*	12*	1.6	3*	10.8	15	
15		a=0.0, b=1.6	2.3	3	2.0	2	4.3	12*	12*	1.6	3*	10.8	15	
$f_3(x)$ larger root		5	a=1.5, b=2.0	2.5	4	1.7	3	4.1	0	0	2.1	5	8.6	5
		7	a=1.5, b=2.0	1.8	1	2.0	3	3.9	0	0	2.1	5	8.6	5
		9	a=1.5, b=2.0	1.6	1	2.2	3	3.8	0	0	2.1	5	8.6	5
	11	a=1.5, b=2.0	1.4	1	2.3	3	3.7	0	0	2.1	5	8.6	5	
	13	a=1.5, b=2.0	1.3	1	2.4	3	3.7	0	0	2.1	5	8.6	5	
	15	a=1.5, b=2.0	1.2	1	2.5	3	3.7	0	0	2.1	5	8.6	5	
	5	a=1.5, b=2.0	3.9	4	1.2	3	5.2	0	0	2.1	5	8.6	5	
	7	a=1.5, b=2.0	2.9	1	1.6	3	4.4	0	0	2.1	5	8.6	5	
	9	a=1.5, b=2.0	2.5	1	1.7	3	4.2	0	0	2.1	5	8.6	5	
	11	a=1.5, b=2.0	2.2	1	1.8	3	4.0	0	0	2.1	5	8.6	5	
	13	a=1.5, b=2.0	2.1	1	1.9	3	3.9	0	0	2.1	5	8.6	5	
	15	a=1.5, b=2.0	1.9	1	1.9	3	3.9	0	0	2.1	5	8.6	5	
	$f_3(x)$ $d_1=1$	5	a=-2.0, b=2.0	2.8	2	1.7	3	4.5	0	0	2.1	5	8.6	5
		7	a=-2.0, b=2.0	2.1	1	2.0	3	4.1	0	0	2.1	5	8.6	5
		9	a=-2.0, b=2.0	1.8	1	2.2	3	4.0	0	0	2.1	5	8.6	5
11		a=-2.0, b=2.0	1.6	1	2.3	3	3.9	0	0	2.1	5	8.6	5	
13		a=-2.0, b=2.0	1.5	1	2.3	3	3.9	0	0	2.1	5	8.6	5	
15		a=-2.0, b=2.0	1.4	1	2.3	3	3.9	0	0	2.1	5	8.6	5	
5		a=-2.0, b=2.0	4.3	2	1.2	3	5.5	0	0	2.1	5	8.6	5	
7		a=-2.0, b=2.0	3.1	2	1.5	3	4.7	0	0	2.1	5	8.6	5	
9		a=-2.0, b=2.0	2.7	1	1.7	3	4.4	0	0	2.1	5	8.6	5	
11		a=-2.0, b=2.0	2.4	1	1.8	3	4.3	0	0	2.1	5	8.6	5	
13		a=-2.0, b=2.0	2.3	1	1.8	3	4.2	0	0	2.1	5	8.6	5	
15		a=-2.0, b=2.0	2.1	1	1.9	3	4.1	0	0	2.1	5	8.6	5	

N = number of processors  
 $m_1$  = number of iterations in which the search procedure determines the iterate  
 $m_2$  = number of iterations in which Halley's method determines the iterate  
 $m = m_1 + m_2$  = total number of iterations such that  
 $b_f - a_f \leq 10^{-d}$ ,  $d = 15$   
 $d_1$  determines the range of effectiveness of Halley's method [see equation (3)].  
 $|x - \alpha| \leq |\lambda_3/\mu_3| 10^{-d_1}$   
 \* in these cases Halley's method determined the first iterate, then the search method determines the next iterates, and then Halley's method determined the last iterate.  
 This situation is not included in our assumptions.

For each iteration function, contraction mapping or Halley's method, the SAPE strategy will be more efficient than the PASE strategy when  $T_S < T_P$ . Using (15) and (16), this condition will hold when

$$(19) \quad S_p(N) > m(\text{SAPE})m_F(\text{SAPE}) / (m(\text{PASE})m_F(\text{PASE})).$$

For each iteration function (contraction mapping or Halley's method) the linear and logarithmic speed-up functions in (17) and (18) can be substituted in (19). Depending upon  $S_p(N)$ ,  $\lambda_n$ ,  $N$  and the other parameters involved ( $\lambda_n/\mu_n$ ,  $(b-a)$ ,  $d$ ) the condition (19) can be tested and a decision can be made as to whether the SAPE or PASE strategy is preferable. We call the curves

$$(20) \quad S_p(N) = m(\text{SAPE})m_F(\text{SAPE}) / (m(\text{PASE})m_F(\text{PASE}))$$

decision curves.

Some conclusions can be drawn (particularly for large  $N$ ) without actually plotting the decision curves. Consider first the contraction mapping ( $n = 1$ ). For the contraction mapping  $m_F(\text{SAPE}) = 3$ ,  $m_F(\text{PASE}) = 1$ . If  $|\lambda_1| > \frac{1}{2}$  then  $|\lambda_1| \geq 1/(N-1)$  for all  $N$  since it is assumed  $N \geq 3$ . Therefore (19) becomes:

$$(21) \quad \begin{aligned} S_p(N) &= 1 + A(N-1) > 3 \log(N-1) / \log 2 && \text{(linear speed-up),} \\ S_p(N) &= 1 + B \log N > 3 \log(N-1) / \log 2 && \text{(logarithmic speed up)} \end{aligned}$$

independent of the values of all other parameters. For linear speed-up, then, SAPE is preferable if

$$A > \frac{1}{N-1} \left( \frac{3 \log(N-1)}{\log 2} - 1 \right) \quad \text{(linear speed-up),}$$

which is always true for sufficiently large  $N$ . For logarithmic speed-up, SAPE is preferable if

$$B > \frac{1}{\log N} \left( \frac{3 \log(N-1)}{\log 2} - 1 \right) \quad \text{(logarithmic speed-up).}$$

The right-hand side approaches  $3/\log 2 = 9.996$  as  $N \rightarrow \infty$ .

Suppose now that  $|\lambda_1| < \frac{1}{2}$ , and  $N$  is so large that  $1/(N-1) < |\lambda_1|$ . We consider two cases. First, assume that  $d_1 - \log |\lambda_1/\mu_1| + \log(b-a) < 0$ . Then (19) becomes

$$S_p(N) > \frac{3 \log(N-1)}{\log(1/|\lambda_1|)}.$$

For linear speed-up this leads to

$$A > \frac{1}{N-1} \left( \frac{3 \log(N-1)}{\log(1/|\lambda_1|)} - 1 \right)$$

which always holds for sufficiently large  $N$ ; therefore SAPE is more efficient for large  $N$ . For logarithmic speed-up, we obtain

$$B > \frac{1}{\log N} \left( \frac{3 \log(N-1)}{\log(1/|\lambda_1|)} - 1 \right).$$

For large  $N$ , then, SAPE is more efficient if  $B > 3/\log(1/|\lambda_1|)$ , and the line  $|\lambda_1| = 10^{-(3/B)}$  gives the asymptote of the boundary curve.

Now suppose that  $|\lambda_1| < \frac{1}{2}$ ,  $N$  is so large that  $1/(N-1) < |\lambda_1|$  but  $d_1 - \log |\lambda_1/\mu_1| + \log(b-a) > 0$ . Then for linear speed-up (19) becomes

$$S_p(N) = 1 + A(N-1) > \frac{3 \log(N-1)}{d + \log(b-a)} \left[ \frac{1}{\log 2} (d_1 - \log |\lambda_1/\mu_1| + \log(b-a)) + \frac{1}{\log(1/|\lambda_1|)} (d - d_1 + \log |\lambda_1/\mu_1|) \right] \tag{linear speed-up),}$$

which is always true for sufficiently large  $N$ . For logarithmic speed-up (19) is the same as above with the left-hand side replaced by  $S_p(N) = 1 + B \log N$ . Therefore for large  $N$ , SAPE is preferable if

$$B > \frac{3}{d + \log(b-a)} \left[ \frac{1}{\log 2} (d_1 - \log |\lambda_1/\mu_1| + \log(b-a)) + \frac{1}{\log(1/|\lambda_1|)} (d - d_1 + \log |\lambda_1/\mu_1|) \right] \tag{logarithmic speed-up).}$$

Now consider Halley’s method ( $n=3$ ). From (10) it follows that  $m_1(\text{PASE}) \rightarrow \frac{1}{2}$ ,  $m_2(\text{PASE}) \rightarrow 0$  as  $N \rightarrow \infty$ . Therefore  $m(\text{PASE}) \rightarrow \frac{1}{2}$  as  $N \rightarrow \infty$ . For Halley’s method  $m_F(\text{SAPE}) = 7$ ,  $m_F(\text{PASE}) = 2$  and (19) becomes

$$(22) \quad S_p(N) > 7m(\text{SAPE}).$$

Since for both linear and logarithmic speed-up  $S_p(N) \rightarrow \infty$ , whereas  $m(\text{SAPE})$  is independent of  $N$ , it follows that for large enough  $N$ , SAPE is preferable.

Not considered here is a speed-up function of the form  $S_p(N) = C[1 - \exp(-D(N-1))]$ . This represents speed-up saturation; that is, as the number of processors increases, the speed-up function asymptotically approaches a constant value  $C$ . From (20), it is clear that for large  $N$  the PASE strategy is preferable to the SAPE strategy with this  $S_p(N)$  when the contraction mapping is used. From (22) it is clear that SAPE is preferable for large  $N$  when Halley’s method is used if

$$C > 7m(\text{SAPE}).$$

**4.3. Sample decision curves.** In Figs. 1 and 2 we give sample plots of some decision curves ( $|\lambda_1|$  versus  $N$  for fixed values of  $d_1, d, |\lambda_1/\mu_1|, (b-a)$ ) for the contraction mapping. These are the curves

$$S_p(N) = 3m(\text{SAPE})/m(\text{PASE})$$

where we have substituted  $m_F(\text{SAPE}) = 3, m_F(\text{PASE}) = 1$  for the contraction mapping.  $S_p(N)$  is given either by (17) for linear speed-up or by (18) for logarithmic speed-up. For a specific parameter value  $A$ , or  $B$ , points on one side of the curve indicate that PASE is the better strategy, while points on the other side indicate that SAPE is better.

For example consider  $f_1(x)$  described in § 3.2. For this function  $\lambda_1 = .642$ . Say that  $S_p(N)$  is linear with  $A = .5$ . The ratio  $\lambda_1/\mu_1$  has been chosen to correspond to  $g_1(x)$ . Figure 1 indicates that for  $N \leq 29$ , use the PASE strategy, and for  $N \geq 30$  use the SAPE strategy. What this indicates is that with a linear speed-up function  $S_p(N)$ , a point will always be reached where the speed gains associated with parallel function evaluation (SAPE) outweigh gains associated with using a parallel root finding algorithm (PASE).

CONTRACTION MAPPING: LINEAR SPEEDUP

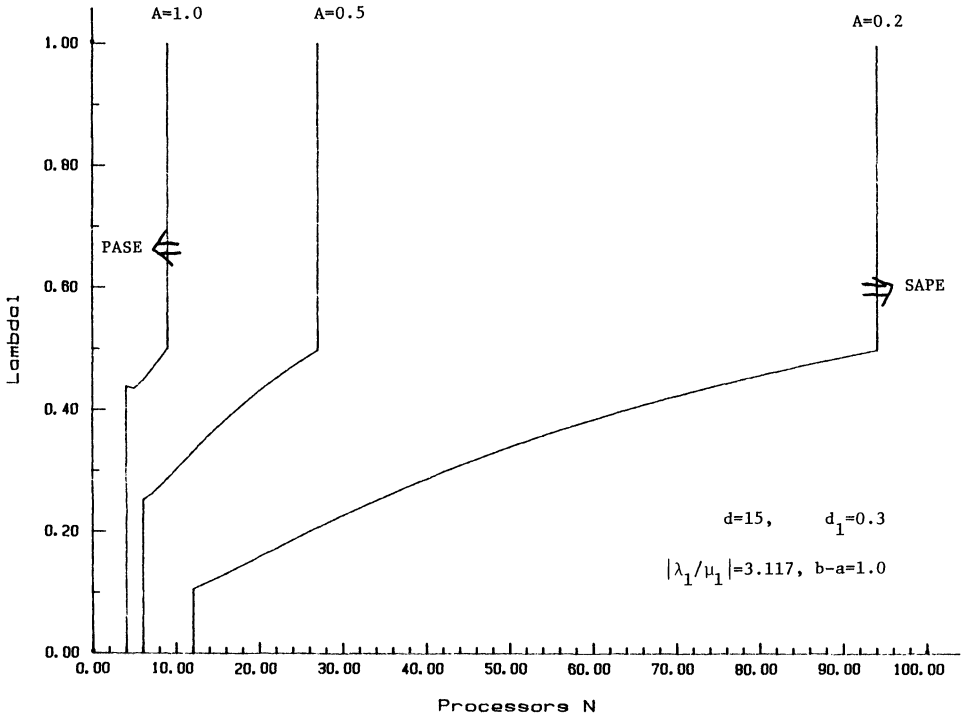


FIG. 1

CONTRACTION MAPPING: LOGARITHMIC SPEEDUP

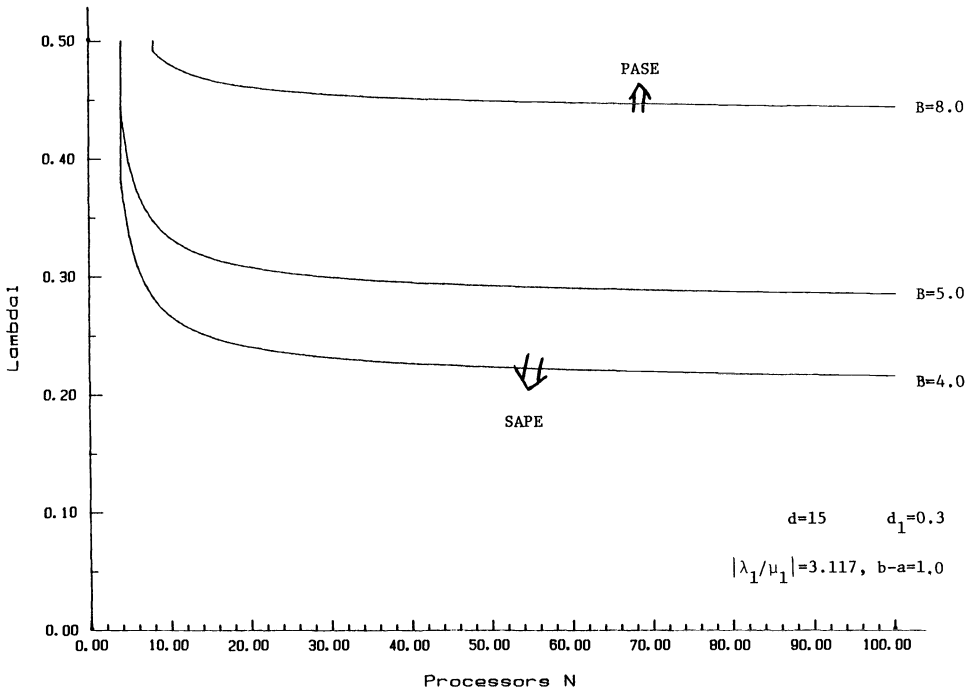


FIG. 2

HALLEY'S METHOD: LINEAR SPEEDUP

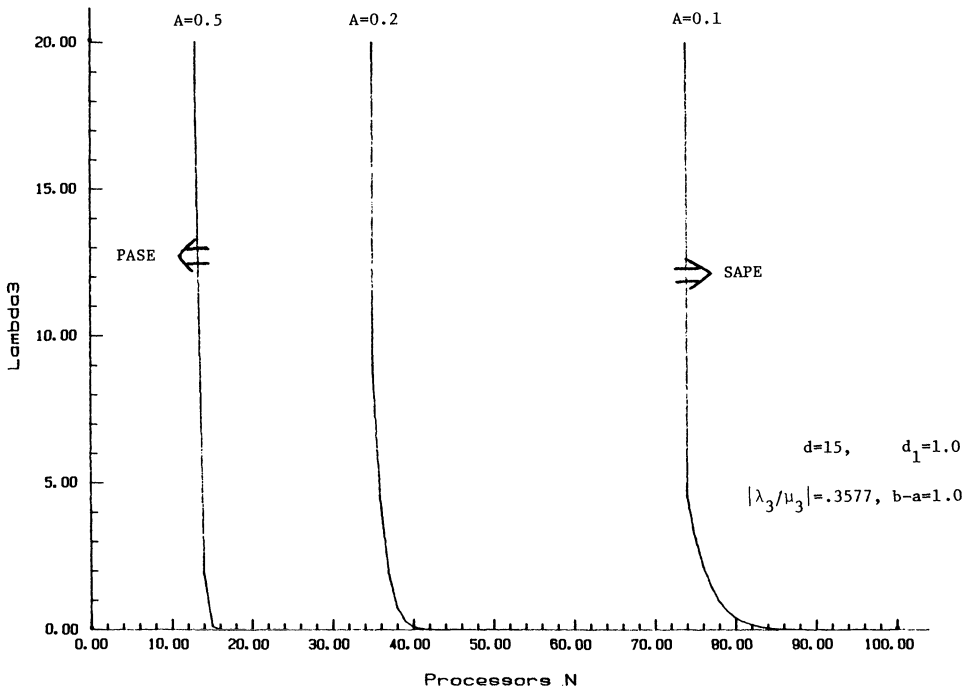


FIG. 3

In Fig. 2 the asymptotes of the decision curves are  $|\lambda_1| = 10^{-(3/B)}$ . For  $B = 4, 5, 8$ , the asymptotes are  $|\lambda_1| = .178, .251, .422$  respectively.

In Figs. 3 and 4 we give sample plots of some decision curves for Halley's method. These are the curves

$$S_p(N) = 7m(\text{SAPE}) / (2m(\text{PASE}))$$

where we have substituted  $m_F(\text{SAPE}) = 7$ ,  $m_F(\text{PASE}) = 2$  for Halley's method.  $S_p(N)$  is again given either by (17) or by (18).

**5. Summary and conclusions.** This paper has considered the problem of finding the roots of a single nonlinear equation on a multiprocessor system of the MIMD variety. Two strategies referred to as SAPE (Sequential Algorithm Parallel function Evaluation) and PASE (Parallel Algorithm Sequential function Evaluation) were investigated. Both strategies are globally convergent to a simple root which has initially been bracketed.

In the first strategy, two sequential root finding algorithms were used, the first a contraction mapping algorithm, and the second Halley's method. Speed-up for this approach results from a parallel evaluation of the function involved. Linear and logarithmic speed-up functions were investigated and expressions were derived for the number of iterations required to find the root for a given accuracy. Sample decision curves were given which allow determination of which strategy would be best when the number of computers available and certain parameters were known.

The results indicate that using either a contraction mapping method or Halley's method for a linear speed-up function, and a large number of processors, the SAPE strategy is best. With a logarithmic speed-up function the best strategy for a large



HALLEY'S METHOD: LOGARITHMIC SPEEDUP

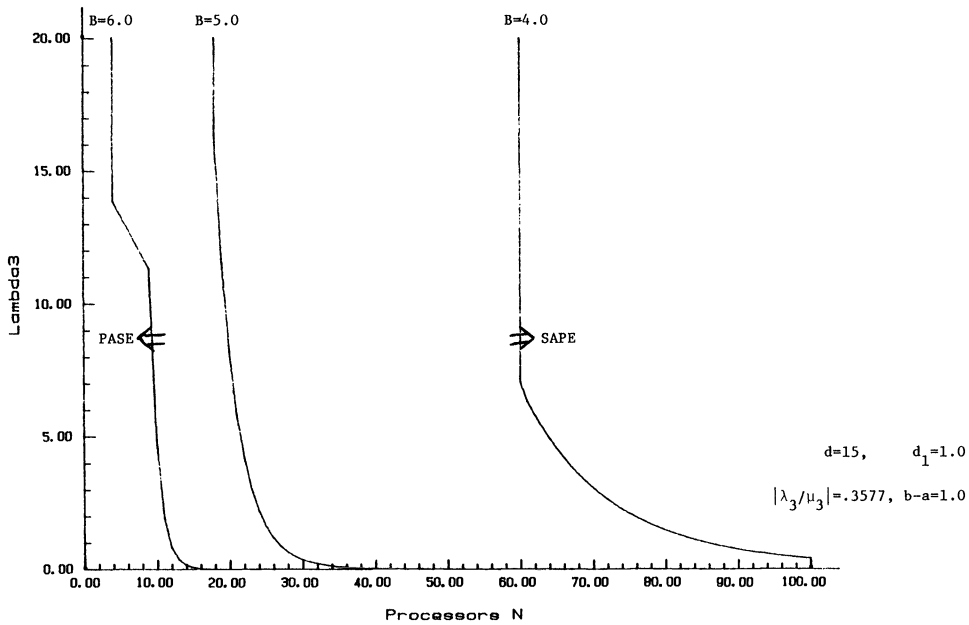


FIG. 4

number of processors will be dependent on the speed-up function parameter. If the speed-up function is of saturating type (i.e. reaches some limiting value as  $N$  becomes large) then for the contraction mapping and large  $N$  the PASE strategy is best whereas for Halley's method the best strategy depends upon the saturation parameter.

**Appendix.** Suppose that  $g(x)$  is an iteration function of third order, and suppose  $\alpha = g(\alpha)$ . In a sufficiently small neighborhood of  $\alpha$ , we have

$$g(x) - \alpha = \lambda_3(x - \alpha)^3 + \mu_3(x - \alpha)^4 + \dots$$

where  $\lambda_3$  and  $\mu_3$  are independent of  $x$  but depend upon  $f(x)$  and its derivatives (up to order 4) evaluated at  $\alpha$ .  $\lambda_3$  is a standard quantity given, for example, in [14]. We now derive expressions for  $\mu_3$  for the iteration functions of Halley, Chebyshev, and Cauchy.

**A.1. Halley's method.**  $g(x)$  is given by (2a). Assume for simplicity that  $\alpha = 0$ . It is shown in [14, p. 94] that  $g(x) = t$  is determined by solving for  $t$  in

$$(A1) \quad f(x) + (t - x)[f'(x) - \frac{1}{2}f''(x)u(x)] = 0$$

where

$$u(x) = f(x)/f'(x).$$

Writing

$$f(x) = f'x + \frac{f''}{2}x^2 + \frac{f'''}{6}x^3 + \frac{f^{iv}}{24}x^4 + \dots$$

where all derivatives are evaluated at 0, and retaining terms needed for a fourth order

approximation, we obtain

$$u(x) = \frac{x}{f'} \left( f'' - \frac{f'''}{2} x - \left( \frac{f''''}{3} - \frac{f''^2}{2f'} \right) x^2 + \dots \right),$$

$$f'(x) - \frac{1}{2} f''(x) u(x) = f' + \frac{f''}{2} x + \frac{f''^2}{4f'} x^2 + \left( -\frac{f''''}{12} + \frac{5}{12} \frac{f'' f'''}{f'} - \frac{f''^3}{4f'^2} \right) x^3 + \dots$$

Substituting in (A1) gives

$$x^3 \left( \frac{f''''}{6} - \frac{f''^2}{4f'} \right) + x^4 \left( \frac{f''''}{8} - \frac{5}{12} \frac{f'' f'''}{f'} + \frac{f''^3}{4f'^2} \right) + \dots + t \left( f' + \frac{f''}{2} x + \dots \right) = 0.$$

Finally solving for  $t$ , we obtain

$$\begin{aligned} t = g(x) &= \left[ x^3 \left( \frac{f''^2}{4f'} - \frac{f''''}{6} \right) + x^4 \left( -\frac{f''''}{8} + \frac{5}{12} \frac{f'' f'''}{f'} - \frac{f''^3}{4f'^2} \right) + \dots \right] \frac{1}{f'} \left[ 1 - \frac{f''}{2f'} x + \dots \right] \\ &= x^3 \left( \frac{f''^2}{4f'^2} - \frac{f''''}{6f'} \right) + x^4 \left( -\frac{f''''}{8f'} + \frac{1}{2} \frac{f'' f'''}{f'^2} - \frac{3f''^3}{8f'^3} \right) + \dots \\ &= \lambda_3 x^3 + \mu_3 x^4. \end{aligned}$$

**A.2. Chebyshev's method.** We use the notation in [14]:  $u(x) = f(x)/f'(x)$ ,  $Y_1 = 1$ ,  $Y_2 = A_2$ ,  $Y_3 = 2A_2^2 - A_3$ ,  $Y_4 = 5A_2^3 - 5A_2A_3 + A_4$ ,  $A_j(x) = f^{(j)}(x)/(j!f'(x))$  (see [14, pages 80, 84, 232 formula 2a]). Chebyshev's iteration function is given by

$$g(x) = x - Y_1(x)u(x) - Y_2(x)u^2 = x - \frac{f(x)}{f'(x)} - \frac{f''(x)}{2f'(x)} \left( \frac{f(x)}{f'(x)} \right)^2.$$

Also the fixed point  $\alpha = g(\alpha)$  satisfies (see [14], page 80)

$$\alpha = x - \sum_{j=1}^{\infty} Y_j u^j.$$

Therefore, it follows that

$$\frac{g(x) - \alpha}{(x - \alpha)^3} = Y_3(x) \left( \frac{u}{x - \alpha} \right)^3 + Y_4(x) \left( \frac{u}{x - \alpha} \right)^4 u + \dots$$

As  $x \rightarrow \infty$ ,  $u(x) = f(x)/f'(x) \rightarrow 0$ , and

$$\frac{u}{x - \alpha} = \frac{(f(x) - f(\alpha))/(x - \alpha)}{f'(x)} \rightarrow 1$$

so we obtain

$$(A2) \quad \lim_{x \rightarrow \infty} \frac{g(x) - \alpha}{(x - \alpha)^3} = Y_3(\alpha).$$

Also we have

$$\begin{aligned} (A3) \quad \frac{g(x) - \alpha}{(x - \alpha)^4} - \frac{Y_3(\alpha)}{(x - \alpha)} &= \frac{(g(x) - \alpha) - Y_3(\alpha)(x - \alpha)^3}{(x - \alpha)^4} \\ &= \frac{Y_3(x)[u(x)]^3 + Y_4(x)[u(x)]^4 + \dots - Y_3(\alpha)(x - \alpha)^3}{(x - \alpha)^4} \\ &= \frac{Y_3(\alpha)([u(x)]^3 - (x - \alpha)^3) + Y_4(x)[u(x)]^4 + (Y_3(x) - Y_3(\alpha))[u(x)]^3}{(x - \alpha)^4} \\ &= \frac{Y_3(\alpha) \left( \left( \frac{u(x)}{y - \alpha} \right)^3 - 1 \right) + Y_4(x) \left( \frac{u(x)}{x - \alpha} \right)^4 + \frac{Y_3(x) - Y_3(\alpha)}{x - \alpha} \left( \frac{u(x)}{x - \alpha} \right)^3}{(x - \alpha)^4}. \end{aligned}$$

Now since

$$\begin{aligned} \frac{u(x)}{x-\alpha} &= \frac{(f(x)-f(\alpha))/(x-\alpha)}{f'(x)} = \frac{f'(\alpha) + (1/2)f''(\alpha)(x-\alpha) + \dots}{f'(\alpha) + f''(\alpha)(x-\alpha) + \dots} \\ &= \left( f'(\alpha) + \frac{1}{2}f''(\alpha)(x-\alpha) + \dots \right) \frac{1}{f'(\alpha)} \left( 1 - \frac{f''(\alpha)}{f'(\alpha)}(x-\alpha) + \dots \right) \\ &= 1 - \frac{1}{2} \frac{f''(\alpha)}{f'(\alpha)}(x-\alpha) + \dots \end{aligned}$$

it follows that

$$\left( \frac{u(x)}{x-\alpha} \right)^3 = 1 - \frac{3}{2} \frac{f''(\alpha)}{f'(\alpha)}(x-\alpha) + \dots$$

Substituting into (A3), we obtain, as  $x \rightarrow \alpha$

$$\frac{g(x) - \alpha}{(x-\alpha)^4} - \frac{Y_3(\alpha)}{(x-\alpha)} \rightarrow -\frac{3}{2} \frac{f''(\alpha)}{f'(\alpha)} Y_3(\alpha) + Y_4(\alpha) - Y_3'(\alpha).$$

Therefore

$$\begin{aligned} g(x) - \alpha &= Y_3(\alpha)(x-\alpha)^3 + (-3Y_2(\alpha)Y_3(\alpha) + Y_4(\alpha) - Y_3'(\alpha))(x-\alpha)^4 + \dots \\ &= \lambda_3(x-\alpha)^3 + \mu_3(x-\alpha)^4 + \dots \end{aligned}$$

where

$$Y_3'(\alpha) = 4A_2(\alpha)A_2'(\alpha) - A_3(\alpha) = 2A_2(\alpha) \frac{(f'''(\alpha)f'(\alpha) - f''^2(\alpha))}{[f'(\alpha)]^2} - A_3(\alpha).$$

**A.3. Cauchy's method.** Cauchy's iteration function is given by (see [14, pp. 93, 232, formula 7a])

$$g(x) = x - \frac{2u(x)}{1 + (1 - 4u(x)A_2(x))^{1/2}}$$

where  $u(x) = f(x)/f'(x)$  and  $A_2(x) = f''(x)/(2f'(x))$ .  $g(x)$  is obtained by solving the following quadratic equation for  $t$  as a function of  $x$  (see [14, p. 93]):

$$P_{0,2}(t) \equiv f(x) + f'(x)(t-x) + \frac{f''(x)}{2}(t-x)^2 = 0.$$

We also have

$$\begin{aligned} 0 = f(\alpha) &= f(x) + f'(x)(\alpha-x) + \frac{f''(x)}{2}(\alpha-x)^2 + \frac{f'''(x)}{6}(\alpha-x)^3 + \frac{f^{iv}(x)}{24}(\alpha-x)^4 + \dots \\ \text{(A4)} \quad &= P_{0,2}(\alpha) + \frac{f''(x)}{6}(\alpha-x)^3 + \frac{f^{iv}(x)}{24}(\alpha-x)^4, \end{aligned}$$

and

$$\begin{aligned} P_{0,2}(\alpha) &= P_{02}(t) + P'_{02}(t)(t-\alpha) + \frac{P''_{02}(t)}{2}(t-\alpha)^2 + \dots \\ &= P'_{02}(t)(\alpha-t) + O[(x-\alpha)^6] \end{aligned}$$

since  $g(x)$  is of third order. Substituting into (A4), we obtain

$$\begin{aligned}
 P'_{02}(t)(t-\alpha) &= (f'(x) + f''(x)(t-x))(t-\alpha) \\
 (A5) \qquad &= -\frac{f'''(x)}{6}(x-\alpha)^3 + \frac{f^{iv}(x)}{24}(x-\alpha)^4 \\
 &= -\frac{f'''(\alpha)}{6}(x-\alpha)^3 - \frac{f^{iv}(\alpha)}{8}(x-\alpha)^4 + \dots
 \end{aligned}$$

Now for the first expression on the right we obtain

$$\begin{aligned}
 (f'(x) + f''(x)(t-x))(t-\alpha) &= (f'(\alpha) + f''(\alpha)(x-\alpha) + f'''(\alpha)(t-x) \\
 &\qquad\qquad\qquad + f^{iv}(\alpha)(x-\alpha)(t-\alpha))(t-\alpha) \\
 &= (f'(\alpha) + f''(\alpha)(t-\alpha) + f'''(\alpha)(x-\alpha)(t-\alpha))(t-\alpha) \\
 &= f'(\alpha)(t-\alpha) + O[(x-\alpha)^6].
 \end{aligned}$$

Therefore substituting into (A5) this gives

$$\begin{aligned}
 t-\alpha = g(x) - \alpha &= -\frac{f'''(\alpha)}{6f'(\alpha)}(x-\alpha)^3 - \frac{f^{iv}(\alpha)}{8f'(\alpha)}(x-\alpha)^4 \\
 &= \lambda_3(x-\alpha)^3 + \mu_3(x-\alpha)^4.
 \end{aligned}$$

**Acknowledgment.** We appreciatively acknowledge the assistance of Sanjay Dhar of the Department of Electrical Engineering, Washington University, for writing and executing the programs which lead to the results in the tables and the figures.

#### REFERENCES

- [1] D. CHAZAN AND W. L. MIRANKER, *A nongradient and parallel algorithm for unconstrained minimization*, SIAM J. Control, 8 (1970), pp. 207-217.
- [2] D. HELLER, *A survey of parallel algorithms in numerical linear algebra*, SIAM Rev., 20 (1978), pp. 740-777.
- [3] W. L. MIRANKER, *A survey of parallelism in numerical analysis*, SIAM Rev., 13 (1971), pp. 523-547.
- [4] ———, *Parallel methods for solving equations*, in *Parallel Computers—Parallel Mathematics*, M. Feilmeir, ed., 1977, pp. 9-15; also *Mathematics and Computers in Simulation*, 20 (1978), pp. 93-101.
- [5] H. MUKAI, *Parallel algorithms for unconstrained optimization*, Proc. of the IEEE Conference on Decision and Control, Fort Lauderdale, FL, 1979.
- [6] ———, *Parallel algorithms for solving nonlinear equations*, Report, Dept. Systems Science and Mathematics, Washington University, St. Louis, 1979.
- [7] W. G. POOLE, JR. AND R. G. VOIGT, *Numerical algorithms for parallel and vector computers: an Annotated Bibliography*, Computing Reviews, October 1974.
- [8] A. H. SAMEH, *Numerical parallel algorithms—A survey*, in *High Speed Computer and Algorithm Organization*, D. J. Kuck, D. H. Lawrie and A. H. Sameh, eds., Academic Press, New York, 1977, pp. 207-228.
- [9] H. S. STONE, *Problems of parallel computation*, in *Complexity of Sequential and Parallel Numerical Algorithms*, Proceedings of a Symposium of Sequential and Parallel Numerical Algorithms, May 1973, J. F. Traub, ed., Academic Press, New York, 1973, pp. 1-16.
- [10] G. S. SHEDLER, *Parallel Numerical Methods for the Solution of Equations*, Communications ACM, 10 (1967), pp. 286-291.
- [11] G. S. SHEDLER AND M. M. LEHRMAN, *Evaluation of redundancy in a parallel algorithm*, IBM Systems J., 6 (1967), pp. 142-149.
- [12] M. A. FRANKLIN AND N. SOONG, *One dimensional optimization of multiprocessor systems*, IEEE Trans. Comput., C-30 (1981), pp. 61-66.
- [13] E. ISAACSON AND H. B. KELLER, *Analysis of Numerical Methods*, John Wiley, New York, 1966.
- [14] J. F. TRAUB, *Iterative Methods for the Solution of Equations*, Prentice-Hall, Englewood Cliffs, NJ, 1964.

- [15] W. H. KAHAN, *Personal calculator adds key to solve any equation  $f(x) = 0$* , Hewlett-Packard J. (December 1979), pp. 20-26.
- [16] B. CARNAHAN, H. A. LUTHER AND J. O. WILKES, *Applied Numerical Methods*, John Wiley, New York, 1969, p. 190.
- [17] M. A. FRANKLIN, *Parallel solution of ordinary differential equations*, IEEE Trans. Comput., C-27 (1978), pp. 413-420.
- [18] L. RASKIN, *Performance evaluation of multiple processor systems*, Ph.D. thesis, Dept. Computer Science, Carnegie-Mellon Univ., Pittsburgh, August 1978.
- [19] G. M. BAUDET, *The design and analysis of algorithms for asynchronous multiprocessors*, Ph.D. thesis, Dept. Computer Science, Carnegie-Mellon Univ., Pittsburgh, April 1978.

## THE NUMERICAL SOLUTION OF HIGHER INDEX LINEAR TIME VARYING SINGULAR SYSTEMS OF DIFFERENTIAL EQUATIONS\*

STEPHEN L. CAMPBELL†

**Abstract.** A family of Taylor-type methods is given for solving numerically a large class of singular systems of differential equations of the form  $A(t)x'(t) + B(t)x(t) = f(t)$ . The advantages, disadvantages, and implementation of these methods are discussed and several analytical and numerical examples are given.

**Key words.** Taylor series methods, singular systems, linear differential equations

**1. Introduction.** Singular linear systems of differential equations of the form

$$(1.1) \quad A(t)x'(t) + B(t)x(t) = f(t)$$

where  $A, B$  are  $n \times n$  matrices and  $A(t)$  is singular for all  $t$ , arise in a wide variety of circuit, control and economic models [2], [3], [12], [13]. They occur both as the natural way in which (circuit) equations are formalized, as reduced order models, in cheap control problems, and in constrained path control problems. See [2], [3] for a more careful development of applications. The numerical and analytic solution of (1.1), if  $A, B$  are constant, is reasonably well-understood [2], [3], [14], [18], [19]. The situation when  $A, B$  depend on  $t$  is quite different. Recent work by Petzold, Gear, [10], [14], [16] and ourselves has shed some light on the structure of (1.1); however, only index one (to be defined in the next section) and some index two systems can be reliably solved numerically [16] without restrictive assumptions on  $A, B$ . The same (implicit) methods do not always work on higher index systems and there is no known easy way to determine when they should work [16]. In fact, no general numerical method has yet been given for (1.1). The outline for the rest of this paper is as follows.

§ 2. Develops necessary notation and terminology concerning (1.1).

§ 3. Motivates and introduces our family of methods.

§ 4. Shows that the restrictive assumptions made in § 3 are actually not very restrictive.

§ 5. Discusses the implementation of our methods and gives the results of some numerical tests.

§ 6. Additional comments.

§ 7. Conclusions.

The purpose of this paper is to present the methods and discuss their general properties. The proofs, accordingly, use "soft" rather than "hard" analysis. Much work developing more precise error bounds and examining the stability of the different variations possible remains to be done.

**2. Notation and terminology.** We shall assume that  $A, B, f$  are sufficiently differentiable functions of  $t$  defined on an interval  $[a, b]$ . Following [8], [14], (1.1) is called *solvable* if for every sufficiently smooth  $f$ , in particular for every infinitely differentiable  $f$ , there is a smooth solution to (1.1) defined on all of  $[a, b]$  and the solutions are uniquely determined by their value at any point in  $[a, b]$ . This definition of solvability is *not* the same as that given in [2], [3].

---

\* Received by the editors June 7, 1983, and in revised form October 28, 1983. This research was sponsored by the Air Force Office of Scientific Research, Air Force System Command, under grant AFOSR-81-0052B.

† Department of Mathematics, North Carolina State University, Raleigh, North Carolina 27650.

Given a pair of  $n \times n$  matrices  $E, F$  such that they form a regular pencil, that is  $\lambda E + F$  is nonsingular for some scalar  $\lambda$ , there exists nonsingular  $P, Q$  so that

$$(2.1) \quad PEQ = \begin{bmatrix} C & 0 \\ 0 & N \end{bmatrix}, \quad PFQ = \begin{bmatrix} G & 0 \\ 0 & I \end{bmatrix},$$

$C$  is invertible,  $CG = GC$ , and  $N$  is nilpotent of index  $k$ . That is  $N^k = 0, N^{k-1} \neq 0$ . (If  $N = 0$ , we consider  $k = 1$ .) Then  $k$  is called the *index* of the pencil  $(E, F)$ , denoted  $k = \text{Index}(E, F)$ . The index of a matrix  $E$  is given by  $\text{Index}(E) = \text{Index}(E, I)$  and is just the index of nilpotency of the largest Jordan block corresponding to a zero eigenvalue in the Jordan canonical form of  $E$ . If  $E$  is invertible, then by definition  $\text{Index}(E) = \text{Index}(E, I) = 0$ . The form (2.1) is a weak version of the Kronecker canonical form of a pencil. See [7, p. 173] for a proof of (2.1).

The index of (1.1) at time  $t$  is the index of the pencil  $(A(t), B(t))$ . In the nomenclature of [16], this is the local index. We assume throughout the paper that  $A(t)$  is singular for all  $t$  so that the index of (1.1),  $k(t)$ , is greater than or equal to one at time  $t$ . If  $k(t) \equiv 1$  on  $[a, b]$ , then (1.1) will be called an *index one system*. If (1.1) is not an index one system, it will be called a higher index system.

**3. The methods.** the usual way [1], [9], [11], [18] of numerically solving (1.1) in either the constant coefficient or index one case is by implicit backward differences. (Note, however, [12], [17], [19].) For example, an implicit backward Euler method would be

$$(3.1) \quad A(t_m) \frac{x_m - x_{m-1}}{h} + B(t_m)x_m = f_m.$$

If  $A(t_m) + hB(t_m)$  is nonsingular, then (3.1) may be solved for  $x_m$  given  $x_{m-1}$ . This method works for index one systems [3], [16], and higher index constant coefficient systems [18] provided the step size is held fixed [16]. In general, it does not work on (1.1) if (1.1) is higher index. (For exceptions see [6], [16].)

There are several problems with using implicit methods such as (3.1) on higher index problems.

(P1) Even in the constant coefficient case the usual step size strategies must be altered [10].

(P2) The method may not converge in practice because of numerical instability.

(P3) The method may not even converge theoretically under the assumption of exact arithmetic.

(P4) The coefficient of  $x_m, A(t_m) + h\beta B(t_m)$  ( $\beta$  depends on the method used) becomes progressively more ill-conditioned as  $h$  decreases.

(P5) Even if  $f$  is known exactly and the derivatives of  $f$  are known exactly, they are computed numerically which limits the accuracy attainable since round-off introduces an error term of the form  $O(\epsilon/h^{k-1})$ .

Our method, at some added overhead, will circumvent all of these difficulties to some extent for many higher index problems.

Examples in [3], [4] and the positive results on (3.1) in [5], [6], [8] show that whatever the method developed to solve (1.1), it must not only involve solving a new system of algebraic equations at each step, but also take into account the way in which  $A(t), B(t)$  change and compute derivatives of  $f$ . With these observations in mind, we now develop our methods.

We shall assume that (1.1) is solvable and we are trying to find the solution  $x(t)$ . Assume at time  $\hat{t}$  we know  $x(\hat{t})$ . The usual Euler method consists of estimating (or

finding  $x'(\hat{t})$  and then estimating  $x(\hat{t} + h)$  by  $\tilde{x}(t + h) = x(\hat{t}) + hx'(\hat{t})$ . We begin by trying to solve for  $x'(\hat{t})$ . Let

$$(3.2) \quad \begin{aligned} x(t) &= \sum_{i=0} x_i \delta^i, & A(t) &= \sum_{i=0} A_i \delta^i, & B(t) &= \sum_{i=0} B_i \delta^i, \\ f &= \sum_{i=0} f_i \delta^i \end{aligned}$$

where  $\delta = t - \hat{t}$ , so that  $c_i = (c^{(i)}(\hat{t}))/i!$  for  $c = x, A, B, f$ . The series will be considered infinite in what follows but it will be clear from the development to follow that we only need the first few terms and hence  $A, B, f, x$  need only be sufficiently differentiable.

Substituting the expansions (3.1) into (1.1) and equating like terms gives the system of equations

$$(3.3) \quad \begin{bmatrix} A_0 & 0 & 0 & 0 & 0 & \cdot \\ A_1 + B_0 & 2A_0 & 0 & 0 & 0 & \cdot \\ A_2 + B_1 & 2A_1 + B_0 & 3A_0 & 0 & 0 & \cdot \\ A_3 + B_2 & 2A_2 + B_1 & 3A_1 + B_0 & 4A_0 & 0 & \cdot \\ A_4 + B_3 & 2A_3 + B_2 & 3A_2 + B_1 & 4A_1 + B_0 & 5A_0 & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \cdot \\ \cdot \\ \cdot \end{bmatrix} = \begin{bmatrix} f_0 - B_0 x_0 \\ f_1 - B_1 x_0 \\ f_2 - B_2 x_0 \\ \cdot \\ \cdot \\ \cdot \end{bmatrix}$$

or

$$(3.4) \quad \mathcal{A}x = f.$$

Any solution of (3.3) must satisfy the  $nj$ -dimensional subsystem

$$(3.5) \quad \begin{bmatrix} A_0 & 0 & 0 \\ A_1 + B_0 & 0 & 0 \\ \vdots & \cdot & \cdot \\ A_{j-1} + B_{j-2} & \cdot \cdot \cdot & jA_0 \end{bmatrix} \begin{bmatrix} x_1 \\ \cdot \\ \cdot \\ x_j \end{bmatrix} = \begin{bmatrix} f_0 - B_0 x_0 \\ \cdot \\ \cdot \\ f_{j-1} - B_{j-1} x_0 \end{bmatrix}$$

or

$$(3.6) \quad \mathcal{A}_j x_j = f_j, \quad j \geq 0.$$

Notice that this equation is to hold exactly. Of course  $\mathcal{A}_j$  is a square singular matrix so that (3.6) does not uniquely determine  $x_j$ . However, it is possible that (3.6) determines  $x_1$  uniquely, or even  $x_1, \dots, x_s, s < j$  uniquely.

It follows from elementary linear algebra (the row echelon normal form) that the first  $z$  components of the consistent system of equations  $Eu = b$  are uniquely determined if and only if there is a nonsingular matrix  $R$  such that

$$(3.7) \quad RE = \begin{bmatrix} I_{zz} & 0 \\ 0 & H \end{bmatrix}.$$

Equivalently, the first  $z$  columns of  $E$  are linearly independent, and the first  $z$  columns are linearly independent from the remaining columns.

We shall say  $\mathcal{A}_j$  is  $s$ -full if (3.7) holds with  $z = sn$  where the  $A_i, B_i$  are  $n \times n$ . Thus (3.6) uniquely determines  $x_1, \dots, x_s$  if and only if it is  $r$ -full with  $r \geq s$ . The question of how often  $\mathcal{A}_j$  is  $s$ -full with  $s \geq 1$  will be left to the next section. In particular, the relationship between the fullness of  $\mathcal{A}_j$  and the traditional analytic methods of solving (1.1) by differentiations and coordinate changes will be discussed at the end of § 4 just prior to Theorem 4.5.



There are many ways to solve (3.6) since solutions are nonunique. We shall solve it in the least squares sense so that the solution may be denoted by  $\mathcal{A}_j^\dagger \mathbf{f}_j$  where  $\mathcal{A}_j^\dagger$  is the Moore–Penrose inverse [7]. This solution was chosen since its numerical behavior is well understood and numerous packages for its computation exist.

We are now ready to define our methods.

The *ith order, j-block method* ( $(i, j)$ -method) with step size  $h$  proceeds as follows: Given  $x_0 = x(\hat{t})$ , form the equations  $\mathcal{A}_j \mathbf{x}_j = \mathbf{f}_j$  and solve for  $\tilde{\mathbf{x}}_j = \mathcal{A}_j^\dagger \mathbf{f}_j$  denoted  $[\tilde{x}_1^T, \dots, \tilde{x}_i^T]^T$ . Then let the value of  $x(\hat{t} + h)$  be estimated by

$$(3.8) \quad \tilde{x}(\hat{t} + h) = x_0 + h\tilde{x}_1 + h^2\tilde{x}_2 + \dots + h^i\tilde{x}_i.$$

Provided  $\mathcal{A}_j$  is  $i$ -full, then (3.8) will be  $O(h^{i+1})$  accurate if  $x(\hat{t})$  was  $O(h^i)$  accurate. The local error is  $O(h^{i+1})$ . When  $A = I$ , this method reduces to a higher order Taylor method. Statements about the accuracy of the  $(i, j)$ -method are based on the analytic calculation of  $\mathcal{A}_j, \mathbf{f}_j$ . Using numerical estimates of  $\mathcal{A}_j, \mathbf{f}_j$  will be discussed in § 6.

Several possible advantages over implicit methods such as (3.1) immediately suggest themselves.

First, provided  $\mathcal{A}_j$  stays  $i$ -full and has constant rank as  $t$  varies, the method should result in a globally  $O(h^i)$  approximation.

Secondly, the conditioning of (3.6) is independent of the step size  $h$ , whereas  $A(t_m) + hB(t_m)$  in (3.1) becomes more and more ill conditioned as  $h \rightarrow 0^+$ .

Thirdly, the step size can be easily varied which is known to sometimes lead to problems for (3.1) even for higher index constant coefficient problems.

**4. The fullness assumption.** The applicability of the  $(i, j)$ -methods depends in large part on the generality of the fullness assumption on  $\mathcal{A}_j$  and the smoothness of its implementation.

**THEOREM 4.1.** *Suppose that  $\mathcal{A}_j$  is  $s$ -full for  $a \leq t \leq b$  and that there exists a continuous invertible  $R(t)$  so that*

$$(4.1) \quad R(t)\mathcal{A}_j(t) = \begin{bmatrix} I_{ns \times ns} & 0 \\ 0 & H(t) \end{bmatrix}.$$

Then the  $(s, j)$ -method with step size  $h$  is  $O(h^s)$  accurate on  $[a, b]$ .

*Proof.* Let

$$R(t) = \begin{bmatrix} R_1(t) & R_2(t) \\ R_3(t) & R_4(t) \end{bmatrix}$$

where  $R_1(t)$  is  $ns \times ns$ . Then applying  $R(t)$  to (3.5) yields

$$(4.2) \quad \begin{bmatrix} x_1 \\ \vdots \\ x_s \end{bmatrix} = [R_1(t) \quad R_2(t)]\mathbf{f}_j(t) \\ = [R_1(t) \quad R_2(t)] \begin{bmatrix} f_0 \\ \vdots \\ f_{j-1} \end{bmatrix} - [R_1(t) \quad R_2(t)] \begin{bmatrix} B_0 \\ \vdots \\ B_{j-1} \end{bmatrix} x_0.$$

Since  $R_1, R_2$  are continuous and hence uniformly bounded on  $[a, b]$ , we are done. This procedure allows us to solve for the first  $s$  derivatives of  $x$  if  $R_1$  and  $R_2$  are known.  $\square$

Note that (4.2) implies that  $R_1, R_2$ , are actually smoother than continuous. The expression (4.2) is the key to understanding why the  $(i, j)$ -methods work so well. If

(4.1) holds with  $R(t)$  which is invertible and  $l$ -times differentiable, we shall say  $\mathcal{A}_j$  is  $l$ -times smoothly  $s$ -full.

**THEOREM 4.2.** *If  $\mathcal{A}_j$  is 1-full for  $a \leq t \leq b$  and there exists a continuous invertible  $R(t)$  so that*

$$R(t)\mathcal{A}_j(t) = \begin{bmatrix} I_{n \times n} & 0 \\ 0 & H(t) \end{bmatrix},$$

let

$$R(t) = \begin{bmatrix} R_{11} & \cdots & R_{1j} \\ R_{21} & \cdots & R_{2j} \end{bmatrix} \quad \text{where each } R_{1r} \text{ is } n \times n.$$

Then every solution of (1.1) is a solution of

$$(4.3) \quad x' = Q(t)x + \bar{f}(t)$$

where

$$(4.4) \quad Q(t) = -\sum_{i=0}^{j-1} R_{1,i+1}(t)B_i(t), \quad \bar{f}(t) = \sum_{i=0}^{j-1} R_{1,i+1}(t)f_i(t).$$

The proof of Theorem 4.2 is trivial. Note that what the  $(1, j)$  method is really doing is solving (4.3) by an explicit Euler method. This has both advantages and disadvantages.

In the constant coefficient case, if an inconsistent initial condition is used, then the implicit backward differences numerically approximate a distribution [3]. This can cause problems if you do not want the distributional behavior. In the  $(i, j)$ -methods, under the assumption  $\mathcal{A}_j$  is continuously  $i$ -full a smooth solution of (4.3) results for every initial condition. However, if the initial condition is inconsistent for (1.1), the resulting solution of (4.3) is now a solution of (1.1).

There is a trade-off here. If we have a reasonable starting value for the initial value, the distributional solutions cause no difficulty. However, if distributional solutions are present and we have an initial condition that is not even close to consistent, we get no indication of that from an  $(i, j)$ -method.

In the  $(i, j)$ -method what is crucial is the continuity of the first  $ni$  rows of  $R$ .

**PROPOSITION 4.1.** *Suppose that  $\mathcal{A}_j$  is  $s$ -full for  $a \leq t \leq b$  and that  $\text{rank}(\mathcal{A}_j)$  is constant on  $[a, b]$ . Then the first  $ns$  rows of  $\mathcal{A}_j^\dagger(t)$ , denoted  $[\bar{R}_1(t), \bar{R}_2(t)]$  may be used for  $[R_1, R_2]$  in (4.2) and  $\bar{R}_1(t), \bar{R}_2(t)$  are as smooth as  $A(t), B(t)$  are.*

*Proof.* That  $\bar{R}_1, \bar{R}_2$  are as smooth as  $A, B$  follows from the assumption that  $\mathcal{A}_j$  has constant rank. As mentioned earlier, if  $\mathcal{A}_j$  is  $s$ -full, then the first  $ns$ -components of any solution to (3.5) are uniquely determined when the equation is consistent and  $\mathcal{A}_j^\dagger f_j$  is a solution of  $\mathcal{A}_j x_j = f_j$  when it is consistent because  $f_j$  is then in the range of  $\mathcal{A}_j$ .  $\square$

It is important to note that Proposition 4.1 does not assume  $A(t)$  has constant rank but rather that  $\text{rank}(\mathcal{A}_j(t))$  is constant.

Before proceeding further, it should also be pointed out that for a general system in the form (1.1), that  $\mathcal{A}_j$  being smooth and  $s$ -full does not imply it is smoothly  $s$ -full.

**Example 4.1.** Consider

$$(4.5) \quad \begin{bmatrix} 0 & 1 \\ 0 & \alpha t \end{bmatrix} \begin{bmatrix} x' \\ y' \end{bmatrix} + \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} f \\ g \end{bmatrix}, \quad t \geq 0,$$

so that

$$\mathcal{A}_3(\alpha, t) = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & \alpha t & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 2 & 0 & 0 \\ 0 & 1+\alpha & 0 & 2\alpha t & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 3 \\ 0 & 0 & 0 & 2\alpha+1 & 0 & 3\alpha t \end{bmatrix}.$$

Then  $\mathcal{A}_3(\alpha, t)$  is 1-full for all values of  $\alpha, t$ . Also  $\text{rank}(\mathcal{A}_3(\alpha, t)) = 4$  if  $\alpha = 0$  or  $t = 0$  while  $\text{rank}(\mathcal{A}_3(\alpha, t)) = 5$  if  $\alpha \neq 0$  and  $t \neq 0$ . It is a simple exercise to show that if

$$R(\alpha, t)\mathcal{A}_3(\alpha, t) = \begin{bmatrix} I_{2 \times 2} & 0 \\ 0 & H \end{bmatrix}, \quad \alpha \neq 0,$$

then the first two rows of  $R(\alpha, t)$  are discontinuous at  $t = 0$ . Note also that (4.5) is not a solvable system on  $[0 \ 1]$  if  $\alpha \neq 0$ .

All of the preceding discussion has yet to address the extent to which the  $(i, j)$ -methods are applicable. We shall now show that they work in almost all cases for which it is currently known that (1.1) is a solvable system.

We begin with the constant coefficient systems.

**THEOREM 4.3.** *Suppose that (1.1) is a solvable system with constant coefficient  $n \times n$  matrices  $A, B$  and the pencil  $(A, B)$  has index  $k$ . Then  $\mathcal{A}_{k+1}$  is 1-full so that the  $(1, k+1)$ -method gives an  $O(h)$  approximation on  $[a \ b]$ .*

*Proof.* For constant coefficient systems, solvability is equivalent to regularity of the pencil [2], so that there exists a constant invertible  $P, Q$  such that

$$(4.6) \quad PAQ = \begin{bmatrix} I & 0 \\ 0 & N \end{bmatrix}, \quad PBQ = \begin{bmatrix} G & 0 \\ 0 & I \end{bmatrix}$$

and  $N^k = 0$ . Thus  $\mathcal{A}_{k+1}$  is 1-full if and only if

$$(4.7) \quad \begin{bmatrix} I & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & N & 0 & 0 & 0 & 0 & 0 & 0 \\ G & 0 & 2I & 0 & 0 & 0 & & \\ 0 & I & 0 & 2N & 0 & 0 & & \\ & & G & 0 & & & & \\ \vdots & \vdots & 0 & I & \vdots & \vdots & \vdots & \vdots \\ & & \cdots & & G & 0 & (k+1)I & 0 \\ 0 & \cdots & 0 & I & 0 & 0 & (k+1)N \end{bmatrix}$$

is 1-full. Use the  $(1, 1)$  entry to eliminate the  $(3, 1)$  entry by a row operation. Similarly use the  $(3, 3)$  entry to eliminate the  $(5, 3)$  entry. Continuing in this manner, we get finally that it suffices to show

$$(4.8) \quad \begin{bmatrix} N & 0 & 0 & \cdots & 0 \\ I & 2N & 0 & & 0 \\ 0 & I & 3N & & \vdots \\ \cdot & & & & \vdots \\ 0 & \cdot & \cdot & I & (k+1)N \end{bmatrix}$$

is 1-full.

Starting at the bottom use each matrix  $I$  to eliminate the matrix  $sN$  above it. The result is

$$\begin{bmatrix} 0 & 0 & & 0 & 0 \\ I & 0 & \cdots & 0 & 0 \\ 0 & I & & \vdots & \vdots \\ \vdots & & & \vdots & \vdots \\ \vdots & & & 0 & -k(k+1)N^2 \\ 0 & \cdot & \cdots & I & (k+1)N \end{bmatrix}$$

which is 1-full.  $\square$

**COROLLARY 4.1.** *For the linear constant coefficient version of (1.1) where  $k = \text{Index}(A, B)$ , assume  $f$  is  $k + i$  times continuously differentiable. Then  $\mathcal{A}_{k+i}$  is  $i$ -full and the  $(i, k + i)$ -method gives  $O(h^i)$  accuracy on  $[a, b]$  if the starting value is  $O(h^i)$  accurate.*

As noted earlier, there already exist methods for the constant coefficient case [3], [17], [18], [19]. The main point of Theorem 4.1 is to assert the  $(i, j)$ -methods also work on these problems and to help in analyzing the general case.

We now turn to the only known procedures for the explicit solution of higher index systems [5], [12], [17]. The key points to determine are whether  $\mathcal{A}_j$  has constant rank and is smoothly  $s$ -full.

We begin with the case studied in [5]. Consider then

$$(4.9) \quad A(t)x'(t) + x(t) = f(t).$$

**THEOREM 4.4.** *Suppose that  $\text{Index } A(t) \leq 2$  on  $[a, b]$ ,  $\text{rank}(A(t)^2)$  is constant, and  $A(t)$  is differentiable. Define  $P, Q, N, C$  by  $P = A^D A, Q = I - P, C = AA^D A, N = (I - A^D A)A$  where  $A^D$  denotes the Drazin inverse of  $A$  [2]. If  $I - N'(t)$  is invertible on  $[a, b]$ , then it is shown in [5] that (4.9) is solvable and how to find the solutions. If, in addition,  $I \pm 2N'$  is invertible and  $\text{rank}(Q(I - N')Q) = \text{rank}(Q(-2N')Q) = \text{rank } Q$ , then  $\mathcal{A}_3$  has constant rank and is smoothly 1-full.*

Before proving this theorem, notice that it does not require  $A(t)$  to have constant rank. Later in this section Theorem 4.4 will be extended to cover (1.1) when  $B(t)$  is invertible and the local index is less than or equal to two.

*Proof of Theorem 4.4.* Since the proof gets somewhat messy, we shall first prove a special case. Consider then

$$Nx' + x = f$$

where  $N^2 = 0$ . Then at  $t$

$$(4.10) \quad \mathcal{A}_3(t) = \begin{bmatrix} N & 0 & 0 \\ I + N' & 2N & 0 \\ N''/2 & I + 2N' & 3N \end{bmatrix}.$$

Since  $N^2 = 0$ , we have  $NN' = -N'N$  and hence  $N(I \pm \alpha N')^{-1} = (I \mp \alpha N')^{-1}N$  when the inverses exist. Now multiply row three of  $\mathcal{A}_3$  by  $-2N(I + 2N')^{-1}$  to get the new second row of

$$(4.11) \quad [I + N' - N(I + 2N')^{-1}N'' \quad 0 \quad 0]$$

since  $-2N(I + 2N')^{-1}N = 2(I - 2N')^{-1}N^2 = 0$ . Multiply (4.11) by  $(I - 2N')$  to yield

$$(4.12) \quad [(I - 2N')(I + N') - NN'' \quad 0 \quad 0].$$

However,

$$NN'' = -N''N - 2(N')^2$$

so that (4.12) is

$$(4.13) \quad [(I - 2N')(I + N') + N''N + 2(N')^2 \quad 0 \quad 0].$$

The  $N''N$  may be eliminated by adding  $-N''$  times row one to (4.13) and row (4.13) is now

$$[I - N' \quad 0 \quad 0].$$

Since  $I - N'$  is invertible, we have that  $\mathcal{A}_3(t)$  is 1-full and has been row reduced to

$$\begin{bmatrix} N & 0 & 0 \\ I - N' & 0 & 0 \\ N''/2 & I + 2N' & 3N \end{bmatrix},$$

two additional row operations gives

$$\begin{bmatrix} 0 & 0 & 0 \\ I - N' & 0 & 0 \\ 0 & I + 2N' & 3N \end{bmatrix}$$

which has constant rank  $2n$ .

The proof of the general case is similar but more complicated.

We have

$$\mathcal{A}_3(t) = \begin{bmatrix} C + N & 0 & 0 \\ I + C' + N' & 2C + 2N & 0 \\ (C'' + N'')/2 & I + 2C' + 2N' & 3C + 3N \end{bmatrix}$$

where  $CN = NC = 0$ ,  $N^2 = 0$ , and  $\text{rank } C^2 = \text{rank } C$ . Note that  $CN' = -N'C$  and  $NN' = -N'N$ . Add  $-2N(I + 2N')^{-1}$  times row three to row two. Since

$$\begin{aligned} 2C + 2N - 2N(I + 2N')^{-1}(I + 2C' + 2N') &= 2C - 2N(I + 2N')^{-1}2C' \\ &= (I - 2N')^{-1}[(I - 2N')C - 2NC'] \\ &= (I - 2N')^{-1}[C - 2(NC)'] = (I - 2N')^{-1}C, \end{aligned}$$

this gives a new second row of

$$[I + C' + N' - N(I + 2N')^{-1}(C'' + N'') \quad (I - 2N')^{-1}C \quad 0].$$

Multiply this row by  $I - 2N'$  to yield

$$[(I - 2N')(I + C' + N') - N(C'' + N'') \quad C \quad 0].$$

Since  $N(C + N) = 0$ , it follows that

$$N(C'' + N'') = -2N'(C' + N') - N''(C + N)$$

so that row two is

$$[(I - 2N')(I + C' + N') + 2N'(C' + N') + N''(C + N) \quad C \quad 0].$$

The  $N''(C + N)$  term is eliminated using row one so that we have the new second row,

$$[I + C' - N' \quad C \quad 0].$$

Suppose for convenience that  $t = 0$  is the time of interest and take a constant similarity

transformation so that we may assume

$$C(0) = \begin{bmatrix} C_1(0) & 0 \\ 0 & 0 \end{bmatrix}, \quad N(0) = \begin{bmatrix} 0 & 0 \\ 0 & N_4(0) \end{bmatrix}$$

and  $C_1(0)$  is invertible. Thus

$$C = \begin{bmatrix} C_1 & tC_2 \\ tC_3 & tC_4 \end{bmatrix}, \quad N = \begin{bmatrix} tN_1 & tN_2 \\ tN_3 & N_4 \end{bmatrix}.$$

Now by assumption  $C_1$  is invertible for small  $t$  and  $\text{rank } C = \text{rank } C_1$  so that

$$C = \begin{bmatrix} C_1 & tC_2 \\ tC_3 & t^2 C_3 C_1^{-1} C_2 \end{bmatrix},$$

while  $CN = 0$  implies (new  $N_1$ )

$$N = \begin{bmatrix} t^2 N_1 & tN_2 \\ tN_3 & N_4 \end{bmatrix};$$

the first two rows at  $t=0$  are now

$$\begin{bmatrix} C_1(0) & 0 & 0 & 0 & 0 & 0 \\ 0 & N_4(0) & 0 & 0 & 0 & 0 \\ I + C_1'(0) & C_2(0) - N_2(0) & C_1(0) & 0 & 0 & 0 \\ C_3(0) - N_3(0) & I - N_4'(0) & 0 & 0 & 0 & 0 \end{bmatrix}.$$

Use the first row to zero out the  $I + C_1'(0)$ ,  $C_3(0) - N_3(0)$  and leave

$$\begin{bmatrix} C_1(0) & 0 & 0 & 0 & 0 & 0 \\ 0 & N_4(0) & 0 & 0 & 0 & 0 \\ 0 & C_2(0) - N_2(0) & C_1(0) & 0 & 0 & 0 \\ 0 & I - N_4'(0) & 0 & 0 & 0 & 0 \end{bmatrix}$$

which will imply  $\mathcal{A}_3(0)$  is 1-full since by assumption  $Q(0)(I - N'(0))Q(0) = I - N_4'(0)$  is invertible. At this point we have row reduced  $\mathcal{A}_3(0)$  to

$$\begin{bmatrix} C_1(0) & 0 & 0 & 0 & 0 & 0 \\ 0 & N_4(0) & 0 & 0 & 0 & 0 \\ 0 & C_2(0) - N_2(0) & C_1(0) & 0 & 0 & 0 \\ 0 & I - N_4'(0) & 0 & 0 & 0 & 0 \\ & & I + 2C_1'(0) & 2(C_2(0) - N_2(0)) & C_1(0) & 0 \\ \boxed{\frac{N''(0)}{2} + \frac{C''(0)}{2}} & & 2(C_2(0) - N_2(0)) & I - 2N_4'(0) & 0 & N_1(0) \end{bmatrix}.$$

Using the rows and columns with only one nonzero entry we see  $\mathcal{A}_3(0)$  has the same rank,  $2n + \text{rank } C$ , as

$$\begin{bmatrix} C_1(0) & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & C_1(0) & 0 & 0 & 0 \\ 0 & I - N_4'(0) & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & C_1(0) & 0 \\ 0 & 0 & 0 & I - 2N_4'(0) & 0 & N_4(0) \end{bmatrix}.$$

Thus under the assumptions of Theorem 4.4,  $\mathcal{A}_3$  has constant rank.  $\square$

The key point of this result is not so much the specific assumptions but rather that  $\mathcal{A}_3$  not being 1-full or having constant rank is the exceptional or special case.

In proving Theorem 4.4 we assumed in (4.9) that  $B(t) = I$ . The next propositions substantially relax that assumption:

PROPOSITION 4.2. *Suppose that  $P(t)$  is  $j$ -times continuously differentiable and invertible on  $[a, b]$ . Let  $\mathcal{A}_j$  be the matrix for (1.1) and  $\mathcal{A}_{jP}$  be the matrix for*

$$(4.14) \quad PAx' + PBx = Pf.$$

*Then there is a smooth invertible matrix  $R(t)$  such that  $R(t)\mathcal{A}_{jP} = \mathcal{A}_j$ . In particular  $\mathcal{A}_{jP}$  and  $\mathcal{A}_j$  have the same rank for all  $t$  and the same fullness properties.*

*Proof.* Let  $P(t) = \sum P_i \delta^i$  as in (3.2). Then

$$\mathcal{A}_{jP} = \begin{bmatrix} P_0 A_0 & 0 & 0 & 0 & \cdot \\ (P_0 A_1 + P_1 A_0) + P_0 B_0 & 2P_0 A_0 & 0 & 0 & \cdot \\ \sum_{i=0}^2 P_i A_{2-i} + (P_0 B_1 + P_1 B_0) & 2(P_0 A_1 + P_1 A_0) + P_0 B_0 & 3P_0 A_0 & 0 & \cdot \\ \sum_{i=0}^3 P_i A_{3-i} + \sum_{i=0}^2 P_i B_{2-i} & 2 \sum_{i=0}^2 P_i A_{2-i} + P_0 B_1 + P_1 B_0 & 3(P_0 A_1 + P_1 A_0) + P_0 B_0 & 4P_0 A_0 & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot \end{bmatrix}.$$

Starting at the top, multiply each row by  $P_0^{-1}$ , define  $\bar{P}_i = P_0^{-1} P_i$  and add  $-\bar{P}_1$  times each row to the one below to get

$$\begin{bmatrix} A_0 & 0 & 0 & 0 & \cdot \\ A_1 + B_0 & 2A_0 & 0 & 0 & \cdot \\ A_2 + \bar{P}_2 A_0 + B_1 & 2A_1 + B_0 & 3A_0 & 0 & \cdot \\ A_3 + \bar{P}_2 A_1 + \bar{P}_2 B_0 + B_2 & 2A_2 + 2\bar{P}_2 A_0 + B_1 & 3A_1 + B_0 & 4A_0 & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot \end{bmatrix}.$$

Now starting at the top add  $-\bar{P}_2$  times each row to the second row below:

$$\begin{bmatrix} A_0 & 0 & 0 & 0 & \cdot \\ A_1 + B_0 & 2A_0 & 0 & 0 & \cdot \\ A_2 + B_1 & 2A_1 + B_0 & 3A_0 & 0 & \cdot \\ A_3 + B_2 & 2A_2 + B_1 & 3A_1 + B_0 & 4A_0 & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot \end{bmatrix}.$$

Continuing this process  $j - 2$  times row reduces  $\mathcal{A}_{jP}$  to  $\mathcal{A}_j$ .  $\square$

The question of whether a given  $\mathcal{A}_j$  is  $s$ -full is unchanged by the adding of multiples of a column of  $\mathcal{A}_j$  to any column to the left of that column. Performing column operations starting at the right-most column of  $\mathcal{A}_j$ , we may prove the following Proposition in a manner analogous to the way Proposition 4.2 was proven.

PROPOSITION 4.3. *Suppose that  $Q(t)$  is  $j$ -times differentiable and invertible on  $[a, b]$ . Letting  $x = Qy$  changes (1.1) to*

$$(4.15) \quad AQy' + (AQ' + BQ)y = f.$$

*Let  $\mathcal{A}_j$  be the matrix for (1.1) and  $\mathcal{A}_{jQ}$  the matrix for (4.15). Then  $\mathcal{A}_j$  is  $s$ -full if and only if  $\mathcal{A}_{jQ}$  is  $s$ -full and  $\text{rank } \mathcal{A}_{jQ} = \text{rank } \mathcal{A}_j$  for all  $t$  in  $[a, b]$ .*

One of the earliest methods for solving (1.1), frequently used for finding higher order singular arcs in optimal control theory, consists of repeatedly differentiating and performing coordinate changes [8], [12], [17]. This technique and its variations require

constant rank assumptions and can be quite difficult to apply since, in general, the coordinate changes are time varying.

We shall now show that when these techniques work with  $j$ -differentiations, then  $\mathcal{A}_{j+1}$  is 1-full. Since the  $(1, j)$ -methods work directly with derivatives of the original coefficients  $A(t)$ ,  $B(t)$  instead of differentiating time varying coordinate changes applied to differentiated blocks, our method is probably easier to apply in most higher index problems if only one solution is sought.

Suppose then that we going to solve (1.1) by using differentiations and coordinate changes. Let  $\mathcal{A}_j$  be the associated matrix (3.5). As shown in Propositions 4.2, 4.3, performing the corresponding coordinate changes on  $A$ ,  $B$  to get a new  $\tilde{\mathcal{A}}_j$  does not alter the rank or fullness. Thus to show that  $\mathcal{A}_j$  is, say, 1-full and of constant rank we need only show the transformed  $\tilde{\mathcal{A}}_j$  has those properties. For notational convenience we shall work with  $\mathcal{A}_4$ .

The first assumption needed is that  $A(t)$  has constant rank. Then there exists smooth  $P$ ,  $Q$  such

$$P(t)A(t)Q(t) = \begin{bmatrix} I & 0 \\ 0 & 0 \end{bmatrix}.$$

Let  $x = Qy$  and multiply by  $P$  to transform  $Ax' + Bx = f$  to

$$(4.16) \quad \begin{aligned} x'_1 - B_{11}x_1 + B_{12}x_2 &= f_1, \\ B_{21}x_1 + B_{22}x_2 &= f_2. \end{aligned}$$

The corresponding  $\mathcal{A}_4$  is

$$(4.17) \quad \begin{bmatrix} I & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ B_{11}B_{12} & 2I & 0 & 0 & 0 & 0 \\ B_{21}B_{22} & 0 & 0 & 0 & 0 & 0 \\ B'_{11}B'_{12} & B_{11}B_{12} & 3I & 0 & 0 & 0 \\ B'_{21}B'_{22} & B_{21}B_{22} & 0 & 0 & 0 & 0 \\ \frac{1}{2} \begin{bmatrix} B''_{11}B''_{12} \\ B''_{21}B''_{22} \end{bmatrix} & B'_{11}B'_{12} & B_{11}B_{12} & 4I & 0 & 0 \\ & B'_{21}B'_{22} & B_{21}B_{22} & 0 & 0 & 0 \end{bmatrix}.$$

If  $B_{22}$  is invertible for all  $t$ , then (4.16) is index 1 and  $\mathcal{A}_4$  is 3-full. Suppose then  $B_{22}$  has constant rank and is always singular. Differentiate the second equation (4.16) to give

$$(4.18) \quad \begin{bmatrix} I & 0 \\ B_{21} & B_{22} \end{bmatrix} \begin{bmatrix} x'_1 \\ x'_2 \end{bmatrix} + \begin{bmatrix} B_{11} & B_{12} \\ B'_{21} & B'_{22} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} f_1 \\ f'_2 \end{bmatrix}.$$

The corresponding  $\tilde{\mathcal{A}}_4$  for (4.18) is

$$(4.19) \quad \begin{bmatrix} I & 0 & 0 & 0 & 0 & 0 \\ B_{21}B_{22} & 0 & 0 & 0 & 0 & 0 \\ B_{11}B_{12} & 2I & 0 & 0 & 0 & 0 \\ 2B'_{21}2B'_{22} & 2B_{21}2B_{22} & 0 & 0 & 0 & 0 \\ B'_{11}B'_{12} & B_{11}B_{12} & 3I & 0 & 0 & 0 \\ \frac{3}{2} \begin{bmatrix} B'_{21}B'_{12} \\ B'_{21}B'_{22} \end{bmatrix} & 3B'_{21}3B'_{22} & 3B_{21}3B_{22} & 0 & 0 & 0 \\ \frac{1}{2} \begin{bmatrix} B''_{11}B''_{12} \\ B''_{21}B''_{22} \end{bmatrix} & B'_{11}B'_{12} & B_{11}B_{12} & 4I & 0 & 0 \\ \frac{2}{3} \begin{bmatrix} B''_{21}B''_{12} \\ B''_{21}B''_{22} \end{bmatrix} & 2B'_{21}2B'_{22} & 4B'_{21}4B'_{22} & 4B_{21}4B_{22} & 0 & 0 \end{bmatrix}.$$



Notice that the nonzero rows of (4.17) appear as the first 7 rows of (4.19). Thus to show (4.17) is 1-full it suffices to show that  $\tilde{A}_3$  for (4.19) is 1-full.

There are two ways to proceed at this point. One is to transform  $B_{22}$  in (4.16), the other is to transform  $\begin{bmatrix} I & 0 \\ B_{21} & B_{22} \end{bmatrix}$  in (4.18). From our point of view it does not matter which is done so we shall do the second. Assume  $\begin{bmatrix} I & 0 \\ B_{21} & B_{22} \end{bmatrix}$  has constant rank. Then there exists  $P_1, Q_1$  such that

$$P_1 \begin{bmatrix} I_1 & 0 \\ B_{21} & B_{22} \end{bmatrix} Q_1 = \begin{bmatrix} \tilde{I}_2 & 0 \\ 0 & 0 \end{bmatrix},$$

where  $\tilde{I}_2$  is larger than  $I_1$ . Let  $\tilde{a}$  indicate new variables. Under this coordinate change (4.18) again has the form (4.16), but now the new  $\tilde{x}_1$  has larger dimension than the old  $x_1$ . We repeat the procedure. If  $\tilde{B}_{22}$  is invertible, then  $\tilde{\mathcal{A}}_3$  (and hence  $\mathcal{A}_4$ ) is 2-full. If  $\tilde{B}_{22}$  is still singular, the procedure is repeated and we consider  $\tilde{\mathcal{A}}_2$ . If  $\tilde{B}_{22}$  is invertible, then  $\tilde{\mathcal{A}}_{22}$  (and hence  $\tilde{\mathcal{A}}_3$  and hence  $\mathcal{A}_4$ ) is 1-full. We have the following.

**THEOREM 4.5.** *Suppose that (1.1) may be solved by coordinate changes and  $j$ -differentiations as just described. Then  $\mathcal{A}_{j+1}$  is 1-full.*

The number of differentiations needed to solve (1.1) is sometimes known on physical grounds since it is related to the number of times the input (or control)  $f$  is differentiated. In these cases Theorem 4.5 tells us that in using an  $(1, r)$ -method,  $r$  should be at least one more than the number of differentiations.

**5. Examples.** In this section we shall present the results from some simple, but interesting, test problems. More complicated examples have been run. All examples were run in double precision in APL ( $10^{-16}$ ).  $\mathcal{A}_j^\dagger$  was computed using a full rank factorization utilizing partial pivoting and then applying the APL domino operator. The first example is due to Petzold and Gear [10].

*Example 5.1.* Consider

$$(5.1) \quad \begin{bmatrix} 0 & 0 \\ 1 & \eta t \end{bmatrix} \begin{bmatrix} x' \\ y' \end{bmatrix} + \begin{bmatrix} 1 & \eta t \\ 0 & 1 + \eta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} e^t \\ t^2 \end{bmatrix}$$

on  $[-.5, .5]$ . The unique solution to (5.1) is  $x = e^t - \eta t(t^2 - e^t)$ ,  $y = t^2 - e^t$ .

This example has several interesting features. An implicit Euler method used on (5.1) is unstable if  $\eta < -1$ , stable and converges to the true solution if  $\eta > -1$ , and does not even make sense if  $\eta = -1$ . For  $\eta = -1 + \epsilon$ ,  $(A + hB)^{-1}$  involves not only  $1/\eta^2$  but  $1/\epsilon\eta^2$ . For  $\eta \neq -1$ , the index of (5.1) is identically two while for  $\eta = -1$ , the system is solvable but has no index since

$$\left( \begin{bmatrix} 0 & 0 \\ 1 & -t \end{bmatrix}, \begin{bmatrix} 1 & -t \\ 0 & 0 \end{bmatrix} \right)$$

is then a singular pencil.

Starting with the exact initial condition an implicit Euler method was run on (5.1.). The error at  $t = .5$  ( $\|e\| = \sup_{i=1,2} |e_i|$ ) is as follows.

TABLE 5.1  
Euler method error for (5.1).

$\eta =$	-3	-1	1	3
$h = .1$	30	×	.2	.5
.05	1000	×	.08	.2
.01	$10^{16}$	×	.02	.04

The (1, 3) and (2, 4) methods, however, worked well for all tested values of  $\eta$ , even  $\eta \leq 1$ .

TABLE 5.2  
(1, 3)-method error for (5.1). (2, 4)-method error for (5.1).

$\eta =$	-3	-1	1	3	$\eta =$	-3	-1	1	3
$h = .1$	.2	.05	.1	.3	$h = .1$	.01	.006	.002	.01
.05	.1	.02	.06	.14	.05	.004	.002	.0005	.003
.01	.02	.004	.01	.03	.01	.0001	.00006	.00002	.0002

Since (5.1) has local index two, we would expect to use an  $(i, i+2)$ -method for  $i$ th order accuracy. However given a (1, 3)-method is being used, a (2, 3)-method can be used at little added cost. Mathematically, the new approximation need be no more accurate but it should not be any worse. However, since the (2, 3)-method does utilize partial information on the second derivative of the solution, it is of interest to see whether in practice it tends to be any better than a (1, 3)-method.

TABLE 5.3  
(2, 3)-method error for (5.1).

$\eta =$	-3	-1	1	3
$h = .1$	.09	.02	.04	.1
.05	.04	.04	.02	.05
.01	.008	.007	.003	.004

Notice that for this example, the error tends to be less with a (2, 3)- method but occasionally ( $\eta = -1, h = .01$ ) is more.

*Example 5.2.* Now consider

$$(5.2) \quad \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} x' \\ y' \end{bmatrix} + \begin{bmatrix} 1 & t \\ t & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} f \\ g \end{bmatrix}$$

on  $[-1, .8]$  where  $f = t + te' + e'$ ,  $g = e' + t^2$ . For  $t \neq 0$ ,  $k(t) = 1$  while  $k(0) = 2$ . This system is not solvable as defined earlier on  $[-1, .8]$ ; however, it has a solution  $x = t, y = e'$  which is smooth.

If an implicit Euler method is applied to (5.2), it exhibits a jump at  $t = 0$ . For example, with  $h = .01$ , the error goes from  $-.02$  to  $50$  near zero.

TABLE 5.4  
Error for (5.2).

$h =$	.1	.05	.01
(1, 3)-method	.06	.03	.006
(2, 4)-method	.003	.001	.00003

A second order backward difference scheme does work on (5.3) as noted in [4]. A second order method,  $h = .1$ , gives a maximum error of .008 while a (2, 4)-method gives .003.

EXAMPLE 5.3. Now consider  $Nx' + x = f$  on  $[0, 1]$  where

$$(5.3) \quad N = \frac{1}{2} \begin{bmatrix} t & 1 \\ -t^2 & -t \end{bmatrix}, \quad f = \begin{bmatrix} e^t \\ t^2 \end{bmatrix}.$$

This is an index two problem as discussed in § 3 and thus the answer involves derivatives of  $f$  [5]. With a backward difference scheme this derivative is taken numerically whereas the  $(i, j)$ -methods utilize the derivative of  $f$  if known. Equation (5.3) was solved using both a  $(2, 4)$  and a second order backward difference scheme. The error for the implicit difference scheme using  $h = .1, .05, .01$  was  $.08, .02, .0009$  respectively for both  $x_1$  and  $x_2$ . The  $(2, 4)$ -method gave an error in  $x_2$  that was  $\frac{1}{2}$  as large and an error in  $x_1$  that was  $\frac{1}{10}$  as large. From [5], the solution of (5.3) is

$$(5.4) \quad x = f - [I - N']^{-1} N f' = f - \begin{bmatrix} t & 1 \\ -t^2 & -t \end{bmatrix} f'.$$

Notice that on  $[0, 1]$  that  $f'$  makes a somewhat larger contribution to  $x_1$  than to  $x_2$ . Thus one would expect greater accuracy in determining  $f'$  to possibly have a greater effect on the accuracy of  $x_1$ .

**6. Additional comments.** In many applications the explicit analytic computation of the  $A_i, B_i, f_i$  may be difficult, time consuming, or impossible. Suppose  $\mathcal{A}_j$  is  $s$ -full. If  $\hat{\mathcal{A}}_j = \mathcal{A}_j + O(\varepsilon)$  and  $\text{rank } \hat{\mathcal{A}}_j = \text{rank } \mathcal{A}_j$ , then  $\hat{\mathcal{A}}_j^\dagger = \mathcal{A}_j^\dagger + O(\varepsilon)$  also. Thus if  $\hat{A}_j^\dagger$  is to be used instead of  $\mathcal{A}_j^\dagger$  in an  $(s, j)$ -method, in order to maintain locally  $O(h^s)$  accuracy we need  $\varepsilon = h^s$ . For example, in a  $(1, j)$ -method the  $A_i, B_i, f_i$  should be estimated to at least  $O(h)$  accuracy. This, of course, assumes that  $\text{rank } \mathcal{A}_j = \text{rank } \hat{\mathcal{A}}_j$ . The simplest way to insure this rank condition holds numerically is for the estimates of  $A_i, B_i$  to be at the level chosen as zero in the numerical calculation of the  $R_i$ . This is in turn dependent on the conditioning of the least squares problem  $\mathcal{A}_j x_j = f_j$ . Ways to relax these rather stringent requirements on the accuracy with which  $\mathcal{A}_j$  must be computed are under investigation, but the problem is unresolved.

Notice that we needed to assume the smoothness of the solutions on  $[a, b]$  in order to derive (3.5). Some such assumption is necessary but it is possible to limit the amount of smoothness assumed as the next result shows.

**THEOREM 6.1.** *Suppose that (1.1) is a solvable system on  $[a, b]$  and  $\mathcal{A}_j$  is  $1$ -full on  $[a, b]$ . Suppose also that  $A(t), f(t)$  are  $2j+1$  times continuously differentiable and  $B(t)$  is  $2j+1-1$  times continuously differentiable on  $[a, b]$  where  $l \geq 0$ . If  $x(t)$  is a  $j$ -times continuously differentiable solution of (1.1), then it is  $l+j$ -times continuously differentiable on  $[a, b]$ .*

*Proof.* Assume  $l > 0$ , otherwise the result is trivial. Under the assumptions of Theorem 6.1, (3.5) holds for all  $t \in [a, b]$ . Thus  $x_1(t) = x'(t)$  may be written in terms of  $A^{(i)}(t), B^{(i)}(t), f^{(i)}(t)$ , for  $0 \leq i \leq j, 0 \leq r \leq j-1$ , and  $x_0(t) = x(t)$  using sums, products and the inverses of invertible combinations of the  $A^{(i)}, B^{(r)}$ . Thus  $x_1(t) = x'(t)$  is also  $j$ -times continuously differentiable. Hence  $x(t)$  is  $j+1$ -times continuously differentiable. The argument may be repeated until  $x(t)$  is  $l+j$  times differentiable since  $A^{(l)}$  is only  $l+j$  times differentiable.  $\square$

**7. Conclusions.** This paper has presented a family of methods for solving implicit differential equations of the form  $Ax' + Bx = f$ . Like all methods for these equations, it faces the problem of needing an initial value  $x(t_0)$ . The method works for many problems for which implicit difference schemes do not work. It also easily handles many higher index solvable systems with nonconstant ranks. In contrast to implicit

procedures matrix conditioning does not degrade the results when the step size is decreased, and variable step procedures can be more easily used.

## REFERENCES

- [1] R. K. BRAYTON, F. G. GUSTAVSON AND G. D. HACHTEL, *A new efficient algorithm for solving differential-algebraic systems using implicit backward differentiation formulas*, Proc. IEEE, 60 (1972), pp. 98-108.
- [2] S. L. CAMPBELL, *Singular Systems of Differential Equations*, Pitman, Marshfield, MA, 1980.
- [3] ———, *Singular Systems of Differential Equations II*, Pitman, Marshfield, MA, 1982.
- [4] ———, *Consistent initial conditions for singular nonlinear systems*, Circuits, Systems, and Signal Processing, 2 (1983), pp. 45-55.
- [5] ———, *Index two linear time-varying singular systems of differential equations*, SIAM J. Alg. Disc. Meth., 4 (1983), pp. 237-243.
- [6] ———, *One canonical form for higher index linear time varying singular systems of differential equations*, Circuits, Systems, and Signal Processing, 2 (1983), pp. 311-326.
- [7] S. L. CAMPBELL AND C. D. MEYER, JR., *Generalized Inverses of Linear Transformations*, Pitman, Marshfield, MA, 1979.
- [8] S. L. CAMPBELL AND L. PETZOLD, *Canonical forms and solvable singular systems of differential equations*, SIAM J. Alg. Discr. Meth. 4 (1983), pp. 517-521.
- [9] C. W. GEAR, *Simultaneous numerical solution of differential-algebraic equations*, IEEE Trans. Circuit Theory, CT-18 (1971), pp. 89-95.
- [10] C. W. GEAR AND L. R. PETZOLD, *Differential/algebraic systems and matrix pencils*, in Lecture Notes in Mathematics 973, Springer-Verlag, New York, 1983, pp. 75-89.
- [11] W. LINIGER, *Multistep and one-leg methods for implicit mixed differential algebraic systems*, IEEE Trans. Circuits and Systems, CAS-26 (1979), pp. 755-762.
- [12] D. G. LUENBERGER, *Dynamic equations in descriptor form*, IEEE Trans. Automat. Control, AC-22 (1977), pp. 312-321.
- [13] R. W. NEWCOMB, *The semistate description of nonlinear time-variable circuits*, IEEE Trans. Circuits and Systems, CAS-28 (1981), pp. 62-71.
- [14] L. R. PETZOLD, *Differential/algebraic equations are not ODE's*, this Journal, 3 (1983), pp. 367-384.
- [15] ———, *A description of DASSL: A differential/algebraic system solver*, Proc. IMACS World Congress, Montreal, Canada, Aug. 1982.
- [16] C. W. GEAR AND L. R. PETZOLD, *ODE methods for the solution of differential/algebraic systems*, SIAM J. Numer. Anal., 21 (1984), pp. 716-728.
- [17] L. M. SILVERMAN, *Inversion of multivariable linear systems*, IEEE Trans. Automat. Control, AC-14 (1969), pp. 270-276.
- [18] R. F. SINCOVEC, A. M. ERISMAN, E. L. YIP AND M. A. EPTON, *Analysis of descriptor systems using numerical algorithms*, IEEE Trans. Automat. Control, AC-26 (1981), pp. 139-147.
- [19] J. H. WILKINSON, *Note on the practical significance of the Drazin inverse*, in Recent Applications of Generalized Inverses, S. L. Campbell, ed., Pitman, Marshfield, MA, 1982.

## STABILITY OF METHODS FOR SOLVING TOEPLITZ SYSTEMS OF EQUATIONS\*

JAMES R. BUNCH†

**Abstract.** The stability properties of the classical and the new fast algorithms for solving Toeplitz systems are investigated.

**Key words.** Toeplitz matrices, linear equations

**1. Introduction.** A matrix  $T$  is *Toeplitz* if  $T_{ij} = t_{j-i}$ , i.e., the elements on each diagonal are all equal.

$$T = \begin{bmatrix} t_0 & t_1 & t_2 & \cdots & t_{n-1} \\ t_{-1} & t_0 & t_1 & & \vdots \\ t_{-2} & t_{-1} & t_0 & & t_2 \\ \vdots & & & & t_1 \\ t_{-n+1} & \cdots & t_{-2} & t_{-1} & t_0 \end{bmatrix}$$

is an  $n \times n$  Toeplitz matrix. In this paper we shall consider algorithms for solving Toeplitz systems of linear equations,  $Tx = b$ . We want to take advantage of the Toeplitz structure in order to solve  $Tx = b$  in  $O(n^2)$  or fewer operations instead of  $O(n^3)$  operations and with less than  $O(n^2)$  storage. On the other hand, we need our algorithms to be stable so that the solutions may be meaningful.

Another class of matrices related to Toeplitz matrices are Hankel matrices:  $H_{ij} = h_{i+j}$ , i.e., the elements on each cross-diagonal are equal.

$$H = \begin{bmatrix} h_0 & h_1 & h_2 & \cdots & h_{n-1} \\ h_1 & h_2 & & & h_n \\ h_2 & & & & \vdots \\ \vdots & & & & \vdots \\ h_{n-1} & h_n & \cdots & \cdots & h_{2n-2} \end{bmatrix}$$

is an  $n \times n$  Hankel matrix. Then  $T = JH$  and  $\tilde{T} = HJ$  are Toeplitz matrices, where

$$J = \begin{bmatrix} 0 & \cdots & 0 & 1 \\ \vdots & & & 0 \\ 0 & & & \vdots \\ 1 & 0 & \cdots & 0 \end{bmatrix}$$

In order to solve  $Hx = b$ , we may solve  $Tx = Jb$ , or  $\tilde{T}y = b$  and  $x = Jy$ .

Block-Toeplitz and block-Hankel systems also occur, i.e., the  $t_i$  and  $h_i$  are matrices [54], [67]. Furthermore, two-level (and three-level) systems can occur, i.e., the matrix is block-Toeplitz with Toeplitz or Hankel blocks (and block-Toeplitz or block-Hankel matrices whose blocks are block-Toeplitz or block-Hankel themselves). In this paper we shall only consider  $n \times n$  Toeplitz systems, i.e., point-Toeplitz matrices.

\* Received by the editors April 5, 1983, and in revised form November 3, 1983. This research was supported in part by the National Science Foundation under grants MCS 79-20491 and DMS 83-18412.

† Department of Mathematics, University of California at San Diego, La Jolla, California 92093.

In § 2 we shall discuss some of the sources of Toeplitz systems and explore the Padé approximation source in detail for later use in § 6. In § 3 we shall discuss the stability of algorithms, the condition of matrices, and the condition of the submatrices under partitioning; we shall show that if a matrix is nonsingular then its leading principal submatrices may be singular and that if a matrix is well-conditioned then its leading principal submatrices may be ill-conditioned, but that if the matrix is positive definite and well-conditioned then its leading principal submatrices must be well-conditioned.

In § 4 we shall discuss the “classical”  $O(n^2)$  algorithms based on Trench’s algorithm [59], [66], [67]. We shall see that these algorithms are unstable, in general, but are stable for positive definite Toeplitz matrices.

In §§ 5 and 6 we shall consider the new “fast”  $O(n \log^2 n)$  algorithms of Bitmead and Anderson [5] and of Brent, Gustavson and Yun [7]. We shall see that the Bitmead–Anderson algorithm is based on partitioning and the fast Fourier transform and is unstable, in general, but probably stable for positive definite Toeplitz matrices. The Brent–Gustavson–Yun algorithm is based on the fast Fourier transform, the fast extended Euclidean algorithm, and on the Gokhberg–Sementsul formulas [22], [23] and is unstable (at least for nonpositive-definite Toeplitz matrices). In § 7 we shall summarize our conclusions and present some conjectures.

**2. Origin of Toeplitz systems.** Toeplitz systems of linear equations arise from many sources [31], [54], [57]:

- (1) time series analysis:
  - (a) linear filtering [31], [39], [46], [61],
  - (b) maximum entropy spectral analysis [59],
  - (c) antennas and arrays [55],
  - (d) estimates of shaping and matching filters [62],
  - (e) recursive estimation [32];
- (2) image processing [1];
- (3) control theory:
  - (a) minimal realization problem [14], [25],
  - (b) minimal design problem [37], [38],
  - (c) adaptive estimation of control system parameters [3];
- (4) statistics:
  - (a) statistical communication and detection [50],
  - (b) stationary auto-regressive time series [28];
- (5) integral equations [16], [48]:
  - (a) convolution-type Volterra equations,
  - (b) convolution-type Fredholm equations;
- (6) orthogonal polynomials [27], [30], [34];
- (7) partial differential equations [42]:
  - (a) elliptic,
  - (b) parabolic;
- (8) Padé approximation [6], [7], [8], [9], [24], [26].

Let us examine Padé approximation more closely, as we shall need the results later in § 6. Let

$$a(x) = a_0 + a_1x + a_2x^2 + \cdots$$



for  $\hat{q}$ , and then  $\hat{p}$  is given by

$$\hat{p} = S\hat{q} + q_0\tilde{a}.$$

Hence, Padé approximation of a power series reduces to solving a Toeplitz system of linear equations. Frobenius [19] has shown that solutions to (†), (\*), (\*\*), always exist and the ratio  $p(x)/q(x)$  is unique.

**3. Stability, condition numbers, and partitioning.** An algorithm for solving systems of linear equations  $Mx = b$  is said to be *stable* if the solution  $x_c$  computed by the algorithm is the exact solution to a nearby system, i.e.,  $(M + E)x_c = b + f$  where  $E$  is small compared to  $M$  and  $f$  is small compared to  $b$ . This does not say that  $x_c$  is close to  $x$ .

The *condition number* of a nonsingular matrix  $M$  is  $\kappa(M) = \|M\| \|M^{-1}\|$  (relative to the matrix norm  $\| \cdot \|$ , e.g.,  $\|M\|_1 = \max_j \sum_i |m_{ij}| = \max$  absolute column sum,  $\|M\|_2 =$  largest singular value of  $M$ ,  $\|M\|_\infty = \max_i \sum_j |m_{ij}| = \max$  absolute row sum [63]). A matrix is *well-conditioned* (ill-conditioned) if  $\kappa(M)$  is “small” (“large”); since  $\kappa(M) \geq 1$ , by “small” and “large”, we mean relative to 1. Since all norms on a finite-dimensional space are equivalent [56], if  $\kappa(M)$  is “very large” for some norm it will also be for any norm. How large a condition number may be acceptable will be dependent on the particular problem. If  $M$  is singular, then we say  $\kappa(M) = \infty$ . If  $\kappa(M)$  is very large, then  $M$  is near a singular matrix [56]; thus, if  $M$  is ill-conditioned,  $M$  is nearly singular.

If  $Mx = b$  and  $\tilde{M}\tilde{x} = \tilde{b}$ , and if  $\|M^{-1}\| \|M - \tilde{M}\| < 1$  [63],

$$\frac{\|x - \tilde{x}\|}{\|x\|} \leq \left( \frac{\kappa(M)}{1 - \kappa(M)(\|M - \tilde{M}\|/\|M\|)} \right) \left\{ \frac{\|M - \tilde{M}\|}{\|M\|} + \frac{\|b - \tilde{b}\|}{\|b\|} \right\}.$$

If  $Mx = b$  is the problem in nature that we wish to solve, but  $\tilde{M}\tilde{x} = \tilde{b}$  is the problem that we have, then the above inequality roughly says that the relative error in  $\tilde{x}$  is approximately bounded by the condition number of the matrix times the relative error in the matrix and the right-hand side. How large a condition number will be acceptable depends on how much accuracy we require in the solution.

Thus, if we use a stable algorithm on a system of equations with a well-conditioned matrix, then our computed solution will be close to the solution of the original problem.

In many algorithms (such as those in §§ 4 and 5) the matrix is partitioned and smaller systems are solved using submatrices. For example, in block Gaussian elimination, we partition the system  $Mx = b$ , where  $M$  is nonsingular, into smaller systems (for simplicity let us use two):

$$Mx = \begin{bmatrix} A & B \\ C & D \end{bmatrix} \begin{bmatrix} \tilde{x} \\ \hat{x} \end{bmatrix} = \begin{bmatrix} \tilde{b} \\ \hat{b} \end{bmatrix} = b,$$

where  $M$  is  $n \times n$ ,  $x$  and  $b$  are  $n \times 1$ ,  $A$  is  $k \times k$ ,  $B$  is  $k \times (n - k)$ ,  $C$  is  $(n - k) \times k$ ,  $D$  is  $(n - k) \times (n - k)$ ,  $\tilde{x}$  and  $\tilde{b}$  are  $k \times 1$ , and  $\hat{x}$  and  $\hat{b}$  are  $(n - k) \times 1$ .

Using block Gaussian elimination and assuming  $A$  is nonsingular, we have

$$\begin{bmatrix} A & B \\ C & D \end{bmatrix} = \begin{bmatrix} I & 0 \\ CA^{-1} & I \end{bmatrix} \begin{bmatrix} A & B \\ 0 & \Delta \end{bmatrix},$$

where  $\Delta = D - CA^{-1}B$ , and

$$\begin{bmatrix} A & B \\ 0 & \Delta \end{bmatrix} \begin{bmatrix} \tilde{x} \\ \hat{x} \end{bmatrix} = \begin{bmatrix} I & 0 \\ -CA^{-1} & I \end{bmatrix} \begin{bmatrix} \tilde{b} \\ \hat{b} \end{bmatrix} = \begin{bmatrix} \tilde{b} \\ \hat{b} - CA^{-1}\tilde{b} \end{bmatrix}.$$



So we solve  $Mx = b$  by:

- (a) solving  $AX = C$  for  $X$ , where  $X$  is  $(n - k) \times k$ ,
- (b) forming  $\Delta = D - XB$ ,
- (c) forming  $\hat{c} = \hat{b} - X\tilde{b}$ ,
- (d) solving  $\Delta\hat{x} = \hat{c}$  for  $\hat{x}$ ,
- (e) forming  $\tilde{c} = \tilde{b} - B\hat{x}$ , and
- (f) solving  $A\tilde{x} = \tilde{c}$  for  $\tilde{x}$ .

If  $M$  is nonsingular then  $A$  need not be nonsingular, e.g.,

$$M = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \quad \text{with } A \text{ of order 1, 2, or 3.}$$

However, if  $M$  and  $A$  are nonsingular, then  $\Delta$  is guaranteed to be nonsingular since  $\det M = \det A \det \Delta$ .

But, in order to have meaningful answers on a computer, we also need to be guaranteed that if  $M$  is well-conditioned then  $A$  and  $\Delta$  are also well-conditioned so that the solutions in (a), (d), and (f) will be accurate. Unfortunately, this is not true, in general. We expect it not to be true since  $M$  can be nonsingular yet  $A$  can be singular, as in the example above. Whenever you have such a situation of singularity, then ill-conditioning (i.e., near singularity) must be lurking nearby.

Let us construct such examples. We shall do it for Toeplitz matrices so that we may use them in §§ 4 and 5. First, we will exhibit a  $3 \times 3$  Toeplitz matrix  $T_3$  which is nonsingular but with a singular upper left  $2 \times 2$  block  $T_2$ :

$$T_3 = \begin{bmatrix} 1 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 1 \end{bmatrix}, \quad T_2 = \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}.$$

Next, we shall perturb  $T_3$  slightly to obtain  $3 \times 3$  Toeplitz matrices  $\tilde{T}_3$  and  $\hat{T}_3$ , one being nonsymmetric and the other being symmetric but indefinite (i.e., having both positive and negative eigenvalues), such that  $\tilde{T}_3$  and  $\hat{T}_3$  are well-conditioned but with ill-conditioned upper left  $2 \times 2$  blocks  $\tilde{T}_2$  and  $\hat{T}_2$ , respectively.

$$\tilde{T}_3 = \begin{bmatrix} 1 & 1 + \varepsilon & 0 \\ 1 - \varepsilon & 1 & 1 + \varepsilon \\ 0 & 1 - \varepsilon & 1 \end{bmatrix} \quad \text{with } \tilde{T}_2 = \begin{bmatrix} 1 & 1 + \varepsilon \\ 1 - \varepsilon & 1 \end{bmatrix},$$

where  $0 < \varepsilon \ll 1$ , is a well-conditioned nonsymmetric matrix,

$$\kappa_\infty(\tilde{T}_3) \equiv \|\tilde{T}_3\|_\infty \|\tilde{T}_3^{-1}\|_\infty = \frac{9}{1 - 2\varepsilon^2} \cong 9,$$

but  $\tilde{T}_2$  is ill-conditioned,

$$\kappa_\infty(\tilde{T}_2) = \frac{(2 + \varepsilon)^2}{\varepsilon^2} \cong \frac{4}{\varepsilon^2}.$$

$$\hat{T}_3 = \begin{bmatrix} 1 & 1 - \varepsilon & 0 \\ 1 - \varepsilon & 1 & 1 - \varepsilon \\ 0 & 1 - \varepsilon & 1 \end{bmatrix} \quad \text{with } \hat{T}_2 = \begin{bmatrix} 1 & 1 - \varepsilon \\ 1 - \varepsilon & 1 \end{bmatrix},$$

where  $0 < \varepsilon \ll 1$ , is a well-conditioned symmetric indefinite matrix,

$$\kappa_\infty(\hat{T}_3) = \frac{(3 - 2\varepsilon)^2}{1 - 4\varepsilon + 2\varepsilon^2} \cong 9,$$

but  $\hat{T}_2$  is ill-conditioned,

$$\kappa_\infty(\hat{T}_2) = \frac{2 - \varepsilon}{\varepsilon} \cong \frac{2}{\varepsilon}.$$

(Since all norms are equivalent in finite-dimensional spaces, the above situation holds independent of the norm used to measure the conditioning.)

This situation will make algorithms based on partitioning, as in §§ 4 and 5, unstable even when used on well-conditioned matrices, at least for nonsymmetric and symmetric indefinite matrices. However, there is one class of matrices for which if  $M$  is well-conditioned then  $A$  must be well-conditioned, namely, the symmetric positive definite matrices:  $x^T M x > 0$  for all  $x \neq 0$ , or equivalently,  $M = M^T$  and all the eigenvalues of  $M$  are positive. (In the complex case, the class will be the Hermitian positive definite matrices—but we will restrict our discussion to the real case.)

Let

$$M = \begin{bmatrix} A & B \\ B^T & D \end{bmatrix}$$

be symmetric positive definite. Then  $A$ ,  $D$ , and  $\Delta \equiv D - B^T A^{-1} B$  are also symmetric positive definite [56]. For convenience, we shall use the 2-norm:  $\|M\|_2 =$  largest singular value of  $M$ . Thus,

$$\kappa_2(M) = \frac{\sigma_{\max}(M)}{\sigma_{\min}(M)} \quad \text{and} \quad \kappa_2(A) = \frac{\sigma_{\max}(A)}{\sigma_{\min}(A)},$$

where  $\sigma_{\max}$  is the largest singular value and  $\sigma_{\min}$  is the smallest. Since  $M$  and  $A$  are symmetric positive definite [56],  $\sigma_{\max}(M) = \lambda_{\max}(M)$ ,  $\sigma_{\min}(M) = \lambda_{\min}(M)$ ,  $\sigma_{\max}(A) = \lambda_{\max}(A)$ ,  $\sigma_{\min}(A) = \lambda_{\min}(A)$ , where  $\lambda_{\max}$  is the largest eigenvalue and  $\lambda_{\min}$  is the smallest eigenvalue. By the Cauchy interlace theorem [49, p. 186], [63, p. 104], since  $M$  is symmetric positive definite,

$$0 < \lambda_{\min}(M) \leq \lambda_{\min}(A) \leq \lambda_{\max}(A) \leq \lambda_{\max}(M).$$

Thus,

$$\kappa_2(A) = \frac{\sigma_{\max}(A)}{\sigma_{\min}(A)} = \frac{\lambda_{\max}(A)}{\lambda_{\min}(A)} \leq \frac{\lambda_{\max}(M)}{\lambda_{\min}(M)} = \frac{\sigma_{\max}(M)}{\sigma_{\min}(M)} = \kappa_2(M).$$

Hence, if  $M$  is well-conditioned then  $A$  is well-conditioned, or equivalently, if  $A$  is ill-conditioned then  $M$  is ill-conditioned.

It would be instructive to see why it is impossible to construct a  $3 \times 3$  symmetric positive definite well-conditioned Toeplitz matrix  $T_3$  with an ill-conditioned upper left  $2 \times 2$  block  $T_2$ . Let  $T_3$  be a  $3 \times 3$  symmetric positive definite Toeplitz matrix,

$$T_3 = \begin{bmatrix} t_0 & t_1 & t_2 \\ t_1 & t_0 & t_1 \\ t_2 & t_1 & t_0 \end{bmatrix},$$

with its upper left  $2 \times 2$  block,

$$T_2 = \begin{bmatrix} t_0 & t_1 \\ t_1 & t_0 \end{bmatrix},$$

being ill-conditioned. Let us see that  $T_3$  *must* be ill-conditioned.

Since  $T_3$  is positive definite,  $t_0 = e_1^T T_3 e_1 > 0$  where  $e_1^T = [1, 0, 0]$ . Without loss of generality, let  $t_0 = 1$ . But if  $M$  is positive definite then  $|M_{ij}| < \frac{1}{2}(M_{ii} + M_{jj})$  for  $i \neq j$  [56]. Hence,  $|t_1| < 1$  and  $|t_2| < 1$ . Since  $T_2$  is ill-conditioned, its rows (and columns) are almost dependent. Hence,  $t_1 = 1 - \epsilon$ , where  $0 < \epsilon \ll 1$ , and

$$T_2 = \begin{bmatrix} 1 & 1 - \epsilon \\ 1 - \epsilon & 1 \end{bmatrix}.$$

Since  $T_3$  is positive definite,  $\det T_3 > 0$  [56]. But

$$\det T_3 = 1 + 2t_2(1 - \epsilon)^2 - t_2^2 - 2(1 - \epsilon)^2 = (1 - t_2)[1 + t_2 - 2(1 - \epsilon)^2].$$

Since  $\det T_3 > 0$  and  $1 - t_2 > 0$ , we must have

$$1 + t_2 - 2(1 - \epsilon)^2 > 0,$$

or, equivalently,

$$t_2 > 1 - 4\epsilon + 2\epsilon^2.$$

So

$$T_3 = \begin{bmatrix} 1 & 1 - \epsilon & t_2 \\ 1 - \epsilon & 1 & 1 - \epsilon \\ t_2 & 1 - \epsilon & 1 \end{bmatrix},$$

where  $t_2$  cannot be outside the interval  $(1 - 4\epsilon + 2\epsilon^2, 1)$ . But then the rows (and columns) of  $T_3$  are almost dependent. Thus,  $T_3$  is ill-conditioned.

For further discussion of partitioning of matrices and its relation to rank- $r$  modification of matrices and to Kron's method of tearing, see [10].

**4. The "classical"  $O(n^2)$  algorithms.** In 1941, N. Wiener [61] and A. N. Kolmogorov [35] independently analyzed the continuous case for linear filtering. In 1947 N. Levinson [39] analyzed the discrete case of linear filtering; the discrete case yields a Toeplitz system of linear equations to solve. Levinson gave an  $O(n^2)$  algorithm for solving this Toeplitz system. In 1964, W. Trench [58] improved Levinson's algorithm (requiring only  $4n^2$  multiplications) and gave a matrix formulation for the method. In 1969, S. Zohar [66] improved Trench's algorithm to require  $3n^2$  multiplications. Zohar's formulation of Trench's algorithm is as follows.

Let  $T_{n+1}$  be a Toeplitz matrix of order  $n + 1$ ; let  $B_{n+1} = T_{n+1}^{-1}$ . We shall partition off the first row and column of  $T_{n+1}$  and  $B_{n+1}$ ; let us assume that the element in the upper left corner of  $T_{n+1}$  is 1:

$$T_{n+1} = \left[ \begin{array}{c|c} 1 & s_n^T \\ \hline r_n & T_n \end{array} \right], \quad B_{n+1} = \frac{1}{\lambda_n} \left[ \begin{array}{c|c} 1 & h_n^T \\ \hline g_n & M_n \end{array} \right].$$

Since  $T_{n+1} B_{n+1} = I_{n+1}$ , we have

$$\frac{1}{\lambda_n} (r_n + T_n g_n) = 0 \quad \text{and} \quad \frac{1}{\lambda_n} (r_n h_n^T + T_n M_n) = I_n.$$

Hence,

$$M_n = \lambda_n B_n - B_n r_n h_n^T = \lambda_n B_n + g_n h_n^T$$

where  $B_n = T_n^{-1}$ .

Thus, given  $B_n = T_n^{-1}$  and the first row and column of  $B_{n+1} = T_{n+1}^{-1}$ , we can compute the other elements of  $T_{n+1}^{-1}$ .

An  $n \times n$  matrix  $A$  is *persymmetric* if it is symmetric about the cross-diagonal, i.e., if  $A_{ij} = A_{n+1-j, n+1-i}$ , i.e., if  $A = JA^TJ$ , where

$$J = \begin{bmatrix} \mathbf{0} & \mathbf{1} \\ \mathbf{1} & \mathbf{0} \end{bmatrix}.$$

If  $T_{n+1}$  is Toeplitz, then  $T_{n+1}$ ,  $B_{n+1} = T_{n+1}^{-1}$ ,  $T_n$ , and  $B_n = T_n^{-1}$  are persymmetric. Let

$$x = \begin{bmatrix} x_1 \\ \vdots \\ x_k \end{bmatrix} \quad \text{and} \quad x^R = \begin{bmatrix} x_k \\ \vdots \\ x_1 \end{bmatrix};$$

then  $x^R = Jx$ . By persymmetry,

$$B_{n+1} = \frac{1}{\lambda_n} \left[ \begin{array}{c|c} 1 & h_n^T \\ \hline g_n & \lambda_n B_n + g_n h_n^T \end{array} \right] = \frac{1}{\lambda_n} \left[ \begin{array}{c|c} \lambda_n B_n + Jh_n g_n^T J & Jh_n \\ \hline g_n^T J & 1 \end{array} \right].$$

The first equality gives

$$(B_{n+1})_{i+1, j+1} = (B_n)_{ij} + \frac{1}{\lambda_n} (g_n h_n^T)_{ij} \quad \text{for } 1 \leq i, j \leq n.$$

The second equality gives

$$(B_{n+1})_{ij} = (B_n)_{ij} + \frac{1}{\lambda_n} (Jh_n g_n^T J)_{ij} \quad \text{for } 1 \leq i, j \leq n.$$

Subtracting

$$(B_{n+1})_{i+1, j+1} = (B_{n+1})_{ij} + \frac{1}{\lambda_n} (g_n h_n^T - Jh_n g_n^T J)_{ij}$$

for  $1 \leq i, j \leq n$ . Thus, from the first row and column of  $B_{n+1}$ , we can generate the other elements.

Now, we can write down an algorithm for computing  $\lambda_n$ ,  $g_n$ , and  $h_n$  and then, from them,  $B_{n+1}$ .

Initialize:  $(s_n)_{i1} = t_i, (r_n)_{i1} = t_{-i}$  for  $1 \leq i \leq n$ .

$$\lambda_1 = 1 - t_1 t_{-1}, \quad h_1 = -t_1, \quad g_1 = -t_{-1}.$$

Compute  $\lambda_n, g_n, h_n$ : For  $1 \leq i \leq n-1$ :

$$\eta_i = -(t_{i+1} + h_i^T J s_i),$$

$$\gamma_i = -(t_{-(i+1)} + r_i^T J g_i),$$

$$h_{i+1} = \begin{bmatrix} h_i + \frac{\eta_i}{\lambda_i} J g_i \\ \eta_i / \lambda_i \end{bmatrix},$$

$$J g_{i+1} = \begin{bmatrix} \gamma_i / \lambda_i \\ J g_i + \frac{\gamma_i}{\lambda_i} h_i \end{bmatrix},$$

$$\lambda_{i+1} = \lambda_i - \eta_i \gamma_i / \lambda_i.$$

This requires  $\sum_{i=1}^{n-1} (4i+3) = 2n^2 + O(n)$  multiplications.

Now we can form  $B_{n+1}$  as follows.

$$\begin{aligned} (B_{n+1})_{11} &= 1/\lambda_n, \\ (B_{n+1})_{1,j+1} &= (h_n)_{j1}/\lambda_n \quad \text{for } 1 \leq j \leq n, \\ (B_{n+1})_{i+1,1} &= (g_n)_{i1}/\lambda_n \quad \text{for } 1 \leq i \leq n, \\ (B_{n+1})_{i+1,j+1} &= (B_{n+1})_{ij} + \frac{1}{\lambda_n}(g_n h_n^T + J h_n g_n^T J)_{ij} \quad \text{for } 1 \leq i, j \leq n. \end{aligned}$$

Then, by persymmetry,

$$(B_{n+1})_{ij} = (B_{n+1})_{n+2-j,n+2-i}$$

This requires an additional  $n^2 + O(n)$  multiplications. Thus, we can compute  $T_{n+1}^{-1}$  with  $3n^2 + O(n)$  multiplications; and  $O(n^2)$  storage is needed. The algorithm is easily generalized to block Toeplitz matrices [54], [67].

Now, the above algorithm is not true unless we assume that the leading principal submatrices of  $T_{n+1}$  are nonsingular and that  $(T_{n+1})_{11} = 1$ . If the leading principal submatrices of  $T_{n+1}$  are nonsingular, then  $(T_{n+1})_{11} = t_0 \neq 0$ , and we may normalize to satisfy the second condition:  $S_{n+1} \equiv (1/t_0)T_{n+1}$ . (Note that the leading principal submatrices and the trailing principal submatrices are the same.)

Thus, Trench’s algorithm may fail if one of the leading principal submatrices is singular. Worse, the algorithm will not fail but may give incorrect results if one of these principal submatrices is nearly singular, i.e., ill-conditioned. As we have seen in § 3, it is quite possible for a well-conditioned Toeplitz matrix to have an ill-conditioned leading principal submatrix when the Toeplitz matrix is nonsymmetric or symmetric indefinite.

However, if  $T_{n+1}$  is symmetric positive definite then all its leading principal submatrices are also symmetric positive definite. As we have seen in § 3, if  $T_{n+1}$  is also well-conditioned, then its leading principal submatrices must also be well-conditioned. Hence, we expect Trench’s algorithm to be stable on well-conditioned symmetric positive definite matrices since it is based on partitioning and persymmetry; and we expect that if Trench’s algorithm gave poor results for some symmetric positive definite Toeplitz matrix  $T_{n+1}$  then  $T_{n+1}$  must have been ill-conditioned. Indeed, Cybenko [11], [12], [13] has shown that Trench’s algorithm is stable for symmetric positive definite Toeplitz matrices.

There are many other variants of Trench’s method [4], [51]–[54] which share its stability properties: stable for positive definite matrices, and unstable otherwise. One such is Durbin’s algorithm [15] for the Yule-Walker equations [60], [65]:

$$Tx = -r,$$

where

$$T = \begin{bmatrix} t_0 & & & & & \\ & t_1 & & & & \\ & & t & & & \\ & & & \cdots & & \\ & & & & & t_{n-1} \\ & & & & & \vdots \\ & & & & & \vdots \\ & & & & & \vdots \\ & & & & & t_1 \\ t_{n-1} & & \cdots & & t_1 & t_0 \end{bmatrix} \quad \text{and} \quad r = \begin{bmatrix} t_1 \\ t_2 \\ \vdots \\ t_{n-1} \\ t_n \end{bmatrix},$$

i.e., a symmetric Toeplitz system with a special right-hand side. Cybenko [12] showed that Durbin’s algorithm is stable when  $T$  is symmetric positive definite. But, as with Trench’s it is unstable when  $T$  is symmetric indefinite.

Another variant of Trench's algorithm for Toeplitz systems is Bareiss' method [2]. However, Douglas Sweet [57] showed that Bareiss' method is unstable. Jain [29] modified Trench's algorithm so that instead of computing  $T^{-1}$  and then  $x$  from  $x = T^{-1}b$ ,  $Tx = b$  is solved by the Gokhberg-Sementsul formula [21], [22], [23] in  $2n^2 + 8n \log_2 n + O(n)$  multiplications (see § 6). We shall show later in § 6 that Jain's algorithm is also unstable. Rissanen's algorithm [51], [52], [53] for the triangular factorization of Hankel matrices, e.g., for the partial minimal realization problem, was shown by de Jong [14] to be unstable.

In order to stabilize Trench's method or its variants some kind of pivoting is needed. However, pivoting can destroy the Toeplitz structures. Thus, maintaining the Toeplitz structure while having stability is a very difficult problem. A first step in this direction has been made by Douglas Sweet [57] in attempting to stabilize Bareiss' variant [4] of Trench's algorithm.

**5. New "fast" methods: the Bitmead-Anderson algorithm.** The first "fast" algorithm that we shall consider is by Bitmead and Anderson [5]; it uses block inversion and the Fast Fourier Transform (FFT), and follows from work of Kailath, Morf, et al. [17], [18], [31]-[34], [37], [46], [47].

Let  $T$  be an  $n \times n$  nonsingular Toeplitz matrix, where  $n = 2^r$ . We seek upper and lower triangular Toeplitz matrices  $U_j$  and  $L_j$  such that

$$T^{-1} = \sum_{j=1}^k U_j L_j,$$

where  $k$  is independent of  $n$ .

Then, to solve  $Tx = b$ , we compute  $z = \sum_{j=1}^k U_j(L_j b)$  by  $2k$  FFT's.

Let us partition  $T$  and  $T^{-1} = S$  as

$$T = \begin{bmatrix} T_{11} & T_{12} \\ T_{21} & T_{22} \end{bmatrix}, \quad S = \begin{bmatrix} S_{11} & S_{12} \\ S_{21} & S_{22} \end{bmatrix},$$

where  $T_{ij}$ ,  $S_{ij}$  are  $n/2 \times n/2$ ,  $1 \leq i, j \leq 2$ .

Let  $\Delta \equiv T_{22} - T_{21} T_{11}^{-1} T_{12}$ . Then

$$\begin{aligned} S_{11} &= T_{11}^{-1} + T_{11}^{-1} T_{12} \Delta^{-1} T_{21} T_{11}^{-1}, & S_{12} &= -T_{11}^{-1} T_{12} \Delta^{-1}, \\ S_{21} &= -\Delta^{-1} T_{21} T_{11}^{-1}, & S_{22} &= \Delta^{-1}. \end{aligned}$$

The various matrix products (and convolutions) can be computed by FFT's. This algorithm is then applied recursively. If  $T(n)$  is the number of multiplications to calculate the decomposition

$$T^{-1} = \sum_{j=1}^k U_j L_j,$$

where  $T$  is  $n \times n$ , then the recursion equation for the algorithm is

$$T(n) = 2T(n/2) + O(n \log n),$$

since the decomposition of the  $n \times n$  matrix  $T^{-1}$  requires the decomposition of two  $n/2 \times n/2$  matrices  $T_{11}^{-1}$  and  $\Delta^{-1}$ , and some subsidiary calculations (done by FFT's requiring  $O(n \log n)$  multiplications). The solution of the recursion equation is  $T(n) = O(n \log^2 n)$  multiplications. The storage required is  $O(n)$ .

A more precise operation count [5] gives  $T(n) = 48n \log_2^2 n$ . This would be faster than the  $3n^2$  for Trench's algorithm for  $n \gtrsim 2000$ , not an unreasonably large order for

Toeplitz matrices. The Bitmead-Anderson algorithm is easily generalized to block Toeplitz matrices.

Makarov [40] proved that if  $T$  is an  $n \times n$  nonsingular Toeplitz matrix then computing  $T^{-1}$  requires at least  $\frac{1}{2}n^2 - \frac{3}{2}n + 1$  multiplications. Does the Bitmead-Anderson algorithm contradict Makarov's theorem? No! We have not computed  $T^{-1}$  itself here, but only the  $U_j$  and  $L_j$  such that  $T^{-1} = \sum_{j=1}^k U_j L_j$ . If we computed  $T^{-1}$  itself from the  $U_j$  and  $L_j$  then we would need  $O(n^2)$  multiplications. However, we can compute the  $U_j$  and  $L_j$  with  $O(n \log^2 n)$  multiplications. Then we can use  $T^{-1} = \sum_{j=1}^k U_j L_j$  to solve  $Tx = b$  by

$$x = \sum_{j=1}^k U_j(L_j b)$$

using  $2k$  FFT's, requiring  $O(n \log n)$  multiplications.

What is the difficulty with the Bitmead-Anderson algorithm? We had to assume  $T_{11}$  and  $\Delta = T_{22} - T_{21} T_{11}^{-1} T_{12}$  were nonsingular, and this would have to be assumed at each step of the recursion. As we have seen in § 3,  $T$  may be nonsingular while  $T_{11}$  may be singular. Even more dangerous is the situation when  $T$  is nonsingular and well-conditioned but one of the needed submatrices is nonsingular but ill-conditioned, i.e., nearly singular. (This is more dangerous since inaccurate answers may be produced and we would not realize that they were inaccurate.) As we have seen in § 3, this is indeed possible for nonsymmetric and symmetric indefinite Toeplitz matrices. Hence, the Bitmead-Anderson algorithm is unstable for nonsymmetric and symmetric indefinite Toeplitz systems.

However, if  $T$  is symmetric positive definite, then  $T_{11}$  and  $\Delta$  are symmetric positive definite, as are the necessary matrices at each step of the recursion. Furthermore, if  $T$  is well-conditioned then so is  $T_{11}$ . The Bitmead-Anderson algorithm is probably stable for symmetric positive definite Toeplitz systems (although it has not been proved in detail yet).

**6. New "fast" methods: the Brent-Gustavson-Yun algorithm.** The second "fast" algorithm we will consider is by Brent, Gustavson, and Yun [7]. In § 2 we showed that the computation of the entries in the Padé Table requires the solution of a Toeplitz system of equations. B-G-Y (Brent, Gustavson, and Yun) reverse the process: they show that the  $(n, n)$  entry in a particular Padé Table gives the solution to  $Tx = e_0$  and they compute this entry in  $O(n \log^2 n)$  operations by the Fast Extended Euclidean Algorithm (for polynomials).

That the entries of the Padé Table can be computed by the Extended Euclidean Algorithm was known to Kronecker [36]; although, it would require  $O(n^2)$  operations if the Extended Euclidean Algorithm was used in the usual manner. However, the Extended Euclidean Algorithm can be modified by Divide and Conquer and the FFT to give a Fast Extended Euclidean Algorithm (FEEA), with its recursion equation as:

$$T(n) = 2T(n/2) + O(n \log n).$$

Here, a problem of degree  $n$  is divided into two problems of degree  $n/2$  with some subsidiary calculations which can be done by FFT's. The solution of the recursion equation is  $O(n \log^2 n)$ . For a detailed discussion of the FEEA, see [2], [7], [43], [44].

The Extended Euclidean Algorithm for polynomials computes the greatest common divisor  $u$  of two polynomials  $p$  and  $q$  as well as polynomials  $w$  and  $v$  such that

$$wp + vq = u,$$

where  $\deg(w) < \deg(q)$  and  $\deg(v) < \deg(p)$ .

Let

$$T = \begin{bmatrix} a_n & a_{n-1} & \cdots & a_0 \\ & \ddots & \ddots & \ddots \\ a_{n+1} & & & \\ \vdots & & & \\ a_{2n} & & & a_n \end{bmatrix}$$

be Toeplitz. Let  $p(x) = x^{2n+1}$  and  $q(x) = a_0 + a_1x + \cdots + a_{2n}x^{2n}$ . Then B-G-Y show that their FEEA produces  $u, v, w$  such that

$$\begin{bmatrix} a_0 & \cdots & 0 \\ \hline a_n & \cdots & a_0 \\ \hline a_{2n} & \cdots & a_n \\ \hline 0 & \cdots & a_{2n} \end{bmatrix} \begin{bmatrix} v_0 \\ \vdots \\ v_n \end{bmatrix} = \begin{bmatrix} u_0 \\ \vdots \\ u_n \\ 0 \\ \vdots \\ 0 \\ -w_0 \\ \vdots \\ -w_n \end{bmatrix}$$

Thus, the  $n$ th through  $2n$ th equations give

$$Tv = u_n e_0.$$

Hence, if  $u_n \neq 0$ , then  $x = (1/u_n)v$ . Further, B-G-Y show that  $\det T = 0$  iff  $u_n = 0$ . So, if  $u_n = 0$ , we have determined that  $\det T = 0$ , while if  $u_n \neq 0$  then we have computed the first column of  $T^{-1}$ .

Similarly, if  $\det T \neq 0$ , we can compute the first row of  $T^{-1}$  by solving

$$Ty^R = T(Jy) = Je_0 = e_n.$$

Given the first row and column of  $T^{-1}$  we can compute  $T^{-1}$  from the first Gokhberg-Sementsul [23] formula if  $T_{11}^{-1} (= x_0 = y_0) \neq 0$ :

$$T^{-1} = \frac{1}{x_0} \left\{ \begin{bmatrix} x_0 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ x_n & \cdots & x_1 & x_0 \end{bmatrix} \begin{bmatrix} y_0 & y_1 & \cdots & y_n \\ \vdots & \vdots & \ddots & \vdots \\ y_1 & & & \\ y_0 & & & \end{bmatrix} - \begin{bmatrix} 0 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ y_n & \cdots & y_1 & 0 \end{bmatrix} \begin{bmatrix} 0 & x_n & \cdots & x_1 \\ \vdots & \vdots & \ddots & \vdots \\ x_n & & & \\ 0 & & & 0 \end{bmatrix} \right\}$$

We can solve  $Tz = b$  from  $z = T^{-1}b$  above by using four FFT's. Thus, if  $T_{11}^{-1} \neq 0$ , we can solve  $Tz = b$  with  $O(n \log^2 n)$  multiplications and  $O(n)$  storage.

However, if  $T_{11}^{-1} = 0$ , then  $T^{-1}$  need not depend solely on the first row and column of  $T^{-1}$ ; for example,

$$T = \begin{bmatrix} 0 & 0 & a \\ c & 0 & 0 \\ d & c & 0 \end{bmatrix}, \quad T^{-1} = \begin{bmatrix} 0 & 1/c & 0 \\ 0 & -d^2/c & 1/c \\ 1/a & 0 & 0 \end{bmatrix}.$$



But, if  $T_{11}^{-1} = 0$ , we can use the second Gokhberg-Sementsul formula. Let

$$\tilde{T} = \left[ \begin{array}{c|c} & \begin{matrix} \gamma \\ a_0 \\ \vdots \\ a_{n-1} \end{matrix} \\ \hline T & a_n \end{array} \right]$$

If  $\tilde{T}\tilde{x} = e_0$  and  $\tilde{T}\tilde{y}^R = e_{n+1}$  exist, and if  $\tilde{T}_{11}^{-1} (= \tilde{x}_0 = \tilde{y}_0) \neq 0$ , then

$$T^{-1} = \frac{1}{\tilde{x}_0} \left\{ \begin{array}{l} \left[ \begin{array}{ccc|ccc} \tilde{x}_0 & & & & & \\ & \mathbf{0} & & & & \\ \tilde{x}_1 & \ddots & & & & \\ \vdots & & \ddots & & & \\ \tilde{x}_n & \cdots & & \tilde{x}_1 & \tilde{x}_0 & \\ \hline & & & & & \mathbf{0} \end{array} \right] \left[ \begin{array}{cccc} \tilde{y}_0 & \tilde{y}_1 & \cdots & \tilde{y}_n \\ & \ddots & & \vdots \\ & & & \tilde{y}_1 \\ & & & \tilde{y}_0 \end{array} \right] \\ - \left[ \begin{array}{ccc|ccc} \tilde{y}_{n+1} & & & & & \\ & \mathbf{0} & & & & \\ \tilde{y}_n & \ddots & & & & \\ \vdots & & \ddots & & & \\ \tilde{y}_1 & \cdots & & \tilde{y}_n & \tilde{y}_{n+1} \\ \hline & & & & & \mathbf{0} \end{array} \right] \left[ \begin{array}{cccc} \tilde{x}_{n+1} & \tilde{x}_n & \cdots & \tilde{x}_1 \\ & \ddots & & \vdots \\ & & & \tilde{x}_n \\ & & & \tilde{x}_{n+1} \end{array} \right] \end{array} \right\}$$

B-G-Y show that if  $x_0 = y_0 = 0$ ,  $\det T \neq 0$ , and  $\det \tilde{T} \neq 0$ , then  $\tilde{x}_0 \neq 0$ . Hence, we only need to choose  $\beta$  and  $\gamma$  to guarantee that  $\det \tilde{T} \neq 0$ . Let  $\gamma = 1$ . Then  $\det \tilde{T} = 0$  iff  $\beta = 1$ . Take  $\beta = -1$ .

Let

$$p(x) = x^{2n+1}, \quad q(x) = -x^{2n+2} + a_{2n}x^{2n+1} + \cdots + a_0x + 1.$$

Then  $\tilde{x}$  and  $\tilde{y}$  exist, and  $\tilde{x}_0 = \tilde{y}_0 \neq 0$ . So we may apply the second Gokhberg-Sementsul formula and the FFT to solve  $Tz = b$  with  $O(n \log^2 n)$  multiplications and  $O(n)$  storage.

This algorithm will never fail in exact arithmetic. In finite precision arithmetic the FFT and the FEEA are stable. But what about the Gokhberg-Sementsul formulas? Whenever we have an algorithm which says to do one thing when  $x_0 \neq 0$  and to do something else when  $x_0 = 0$  must be unstable when  $x_0$  is near zero. For an algorithm to be stable we need a continuous change in the algorithm as  $x_0 \rightarrow 0$ . Here we have a discontinuous jump in algorithms between when  $x_0$  is near zero and when  $x_0$  is zero.

Let us construct an example to show that the first Gokhberg-Sementsul formula, and hence the B-G-Y algorithm, is unstable when  $T_{11}^{-1} = x_0$  is near zero.

Let  $T_{11}^{-1} = x_0 = y_0 = \varepsilon \neq 0$  and  $x_1 = 1 = y_1$ . Then

$$T = \begin{bmatrix} a_0 & a_1 \\ a_1 & a_0 \end{bmatrix},$$

where  $a_0 = -\varepsilon/(1 - \varepsilon^2)$  and  $a_1 = 1/(1 - \varepsilon^2)$ . Let  $b = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$ . Solving  $Tz = b$  by the first Gokhberg-Sementsul formula gives

$$z = \begin{bmatrix} 1 + \varepsilon \\ 1 + \varepsilon \end{bmatrix} \cong \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

in exact arithmetic, but gives

$$z_c = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

in floating point arithmetic when  $\varepsilon$  is small enough so that  $\text{fl}(1 + \varepsilon) = 1$ . Hence, the

Gokhberg-Sementsul formula, the B-G-Y algorithm, and Jain's algorithm are all unstable. Note that  $T$  is not positive definite in this example, so it is possible that the algorithms are stable for positive definite Toeplitz systems.

**7. Conclusions.** The classical  $O(n^2)$  algorithms related to Trench's method and the Bitmead-Anderson  $O(n \log^2 n)$  algorithm are based on partitioning. But, in the nonsymmetric case and in the symmetric indefinite case the principal submatrices may be singular even though the entire matrix is nonsingular, and may be ill-conditioned even though the entire matrix is well-conditioned. Thus, these algorithms may fail and, worse, are unstable for nonsymmetric and symmetric indefinite Toeplitz matrices.

However, in the symmetric positive definite case the principal submatrices are symmetric positive definite, and must be well-conditioned whenever the entire matrix is well-conditioned. Hence, these algorithms will not fail for symmetric positive definite Toeplitz matrices, and will be stable for well-conditioned symmetric positive definite Toeplitz matrices unless orthogonal transformations are used.

Furthermore, any algorithm for solving Toeplitz systems which is based on partitioning (and does not do some form of pivoting) will share this property: stable for well-conditioned symmetric positive definite Toeplitz matrices and unstable for nonsymmetric and symmetric indefinite matrices.

In order to stabilize these algorithms in the nonsymmetric and the symmetric indefinite cases, some kind of pivoting techniques are needed in order to maintain stability. However, pivoting can ruin the Toeplitz structure and can increase the operation count. Stable  $O(n^2)$  algorithms still elude us for nonsymmetric and symmetric indefinite Toeplitz matrices unless orthogonal transformations are used.

In contrast to the above algorithms the B-G-Y  $O(n \log^2 n)$  algorithm will not fail for any nonsingular Toeplitz matrix, i.e., it will always produce the solution (in exact arithmetic). However, the algorithm is unstable on a computer for nonsymmetric and symmetric indefinite matrices.

**Addendum.** Frank de Hoog (*On the solution of Toeplitz systems of equations*, to appear in *Linear Algebra and Appl.*) has presented an algorithm which calculates the first and last row of the inverse of a Toeplitz matrix  $T$  in  $O(n \log^2 n)$  operations but in a different manner from [5] or [7]. He then uses the Gokhberg-Sementsul formula to solve  $Tx = b$ . Once again this method is unstable if  $T$  is general or symmetric indefinite, but it may be stable if  $T$  is symmetric positive definite. Douglas Sweet (*Fast Toeplitz orthogonalization*, *Numer. Math.*, 43 (1984), pp. 1-21) has presented a stable algorithm for computing the QR decomposition of a Toeplitz matrix in  $25n^2 + O(n)$  operations.

#### REFERENCES

- [1] B. P. AGRAWAL AND V. K. JAIN, *Digital restoration of images in the presence of correlated noise*, *Comput. and Elec. Eng.*, 3 (1976), pp. 65-74.
- [2] A. V. AHO, J. E. HOPCROFT AND J. D. ULLMAN, *The Design and Analysis of Computer Algorithms*, Addison-Wesley, Reading, MA, 1974.
- [3] M. AOKI, *Optimization of Stochastic Systems*, Academic Press, New York, 1967.
- [4] E. H. BAREISS, *Numerical solution of linear equations with Toeplitz and vector Toeplitz matrices*, *Numer. Math.*, 13 (1969), pp. 404-424.
- [5] R. R. BITMEAD AND B. D. O. ANDERSON, *Asymptotically fast solution of Toeplitz and related systems of equations*, *Linear Algebra and Appl.*, 34 (1980), pp. 103-116.
- [6] N. K. BOSE AND S. BASU, *Theory and recursive computation of one-dimensional matrix Padé approximants*, *IEEE Trans. Circ. Sys.*, CS-27 (1980), pp. 323-325.

- [7] R. P. BRENT, F. G. GUSTAVSON AND D. Y. Y. YUN, *Fast solution of Toeplitz systems of equations and computation of Padé approximants*, J. Algorithms, 1 (1980), pp. 259-295.
- [8] A. BULTHEEL, *Recursive algorithms for the Padé table: two approaches*, in [64], pp. 211-230.
- [9] ———, *Recursive algorithms for the matrix Padé problem*, Math. Comp., 35 (1980), pp. 875-892.
- [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.
- [11] G. CYBENKO, *Error analysis of some signal processing algorithms*, Ph.D. thesis, Princeton Univ., Princeton, NJ, 1978.
- [12] ———, *The numerical stability of the Levinson-Durbin algorithm for Teopltz systems of equations*, this Journal, 1 (1980), pp. 303-309.
- [13] ———, *Round-off error propagation in Durbin's, Levinson's and Trench's algorithms*, Proc. Inter. Conf. on Acoustics, Speech and Signal Processing, Washington, DC, 1979.
- [14] L. S. DE JONG, *Numerical aspects of recursive realization algorithms*, SIAM J. Control Optim., 16 (1978), pp. 646-659.
- [15] J. DURBIN, *The fitting of time-series models*, Rev. Int. Stat. Inst., 28 (1959), pp. 229-249.
- [16] L. ELDÉN, *Algorithms for the regularization of ill-conditioned least squares problems*, BIT, 17 (1977), pp. 134-145.
- [17] B. FREIDLANDER, T. KAILATH, M. MORF AND L. LJUNG, *Extended and Chandrasekhar equations for general discrete-time estimation problems*, IEEE Trans. Automat. Control, AC-23 (1978), pp. 653-659.
- [18] ———, *New inversion formulas for matrices classified in terms of their distance from Toeplitz matrices*, Linear Algebra Appl., 27 (1979), pp. 31-60.
- [19] G. FROBENIUS, *Über Relation zwischen den Näherungsbrüchen von Potenzreihen*, J. für Math., 90 (1881), pp. 1-17.
- [20] P. E. GILL, G. H. GOLUB, W. MURRAY AND M. A. SAUNDERS, *Methods for modifying matrix factorizations*, Math. Comp., 28 (1974), pp. 505-535.
- [21] I. GOKHBERG AND I. FELDMAN, *Convolution equations and projection methods for their solutions*, Trans. Math. Monographs 41, American Mathematical Society, Providence, RI, 1974.
- [22] I. GOKHBERG AND G. HAJNIK, *Inversion of finite Toeplitz with entries being elements from a noncommutative algebra*, Rev. Roumaine Math. Pures Appl., 19 (1974), pp. 623-663. (In Russian.)
- [23] I. GOKHBERG AND A. SEMENTSUL, *On the inversion of finite Toeplitz matrices and their continuous analogs*, Mat. Issled., 2 (1972), pp. 201-233. (In Russian.)
- [24] W. GRAGG, *The Padé table and its relation to certain algorithms of numerical analysis*, SIAM Rev., 14 (1972), pp. 1-62.
- [25] W. GRAGG AND A. LINDQUIST, *On the partial realization problem*, Linear Algebra and Appl., 50 (1983), pp. 277-319.
- [26] P. R. GRAVES-MORRIS, *The numerical calculation of Padé approximants*, in [64], pp. 231-245.
- [27] U. GRENANDER AND G. SZEGÖ, *Toeplitz Forms and Their Applications*, Univ. California Press, Berkeley, 1958.
- [28] R. E. HARTWIG AND M. E. FISHER, *Asymptotic behaviour of Toeplitz matrices and determinants*, Arch. Rat. Mech. Anal., 32 (1969), pp. 190ff.
- [29] J. R. JAIN, *An efficient algorithm for a large Toeplitz set of linear equations*, IEEE Trans. Acoust., Speech, Sig. Processing, ASSP-27 (1979), pp. 612-615.
- [30] J. H. JUSTICE, *A Levinson-type algorithm for two-dimensional Wiener filtering using bivariate Szegő polynomials*, IEEE Proc., 65 (1977), pp. 882-886.
- [31] T. KAILATH, *A view of three decades of linear filtering theory*, IEEE Trans. Inf. Theory, IT-20 (1974), pp. 146-181.
- [32] ———, *Some alternatives in recursive estimation*, Int. J. Control, 32 (1980), pp. 311-328.
- [33] T. KAILATH, B. LEVY, L. LJUNG AND M. MORF, *The factorization and representation of operators in the algebra generated by Toeplitz operators*, SIAM J. Appl. Math., 37 (1979), pp. 467-484.
- [34] T. KAILATH, A. VIEIRA AND M. MORF, *Inverses of Toeplitz operators, innovations and orthogonal polynomials*, SIAM Rev., 20 (1978), pp. 106-119.
- [35] A. N. KOLMOGOROV, *Interpolation and extrapolation of stationary random sequences*, Izv. Akad. Nauk, 5 (1941), pp. 3-11 (Russian); German summary, pp. 11-14.
- [36] L. KRONECKER, *Zur Theorie der Elimination einer Variablen aus zwei algebraischen Gleichungen*, Monatsb. Konigl. Preuss Akad. Wiss. Berlin (1881), pp. 535-600.
- [37] S. KUNG AND T. KAILATH, *Fast projection methods for minimal design problems in linear systems theory*, Automatica, 16 (1980), pp. 399-403.
- [38] S. KUNG AND D. W. LIN, *Optimal Hankel-norm reductions: multivariate systems*, IEEE Trans. Automat. Control, AC-26 (1981), pp. 832-852.

- [39] N. LEVINSON, *The Wiener rms (root-mean-square) error criterion in filter design and prediction*, J. Math. Phys., 25 (1947), pp. 261-278.
- [40] D. MAKAROV, *The lower bound on the number of multiplications for algorithms for calculating the product of Hankel matrices*, USSR Comput. Math. and Math. Phys., 17 (1978), pp. 195-199.
- [41] J. MAKHOUL, *Linear prediction: a tutorial review*, IEEE Proc., 63 (1977), pp. 561-580.
- [42] M. A. MALCOLM AND J. PALMER, *A fast method for solving a class of tridiagonal linear systems*, Comm. ACM, 17 (1974), pp. 14-17.
- [43] R. MCELIECE AND J. SHEARER, *A property of Euclid's algorithm and an application to Padé approximations*, SIAM J. Appl. Math., 134 (1978), pp. 611-617.
- [44] R. MOENCK, *Fast Computation of GCD's*, Proc. Fifth Annual Symposium on Theory of Computing, 1973, pp. 142-171.
- [45] M. MORF, *Fast algorithms for multivariable systems*, Ph.D. thesis, Stanford Univ., Stanford, CA, 1974.
- [46] M. MORF, B. DICKINSON, T. KAILATH AND A. VIEIRA, *Efficient solution of covariance equations for linear prediction*, IEEE Trans. Acoust., Speech, Sig. Processing, ASSP-25 (1977), pp. 429-433.
- [47] M. MORF AND T. KAILATH, *Square-root algorithms for linear least-squares estimation*, IEEE Trans. Automat. Control, AC-20 (1975), pp. 487-497.
- [48] B. S. PARIISKI, *An economical method for the numerical solution of convolution equations*, USSR Comput. Math. and Math. Phys., 18 (1978), pp. 208-211.
- [49] B. N. PARLETT, *The Symmetric Eigenvalue Problem*, Prentice-Hall, Englewood Cliffs, NJ, 1980.
- [50] W. D. RAY, *The inverse of a finite Toeplitz matrix*, Technometrics, 12 (1970), pp. 153-156.
- [51] J. RISSANEN, *Recursive identification of linear systems*, SIAM J. Control Optim., 9 (1971), pp. 420-430.
- [52] ———, *Algorithms for the triangular decomposition of block Hankel and Toeplitz matrices with application to factoring positive matrix polynomials*, Math. Comp., 27 (1973), pp. 147-154.
- [53] ———, *Solution of linear equations with Hankel and Toeplitz matrices*, Numer. Math., 22 (1974), pp. 361-366.
- [54] P. ROEBUCK AND S. BARNETT, *A survey of Toeplitz and related matrices*, Int. J. Systems Sci., 9 (1978), pp. 921-934.
- [55] D. H. SINNOTT, *Matrix analysis of linear antenna arrays of equally-spaced elements*, IEEE Trans. Antenna and Prop., AP-21 (1973), pp. 385-386.
- [56] G. W. STEWART, *Introduction to Matrix Computations*, Academic Press, New York, 1973.
- [57] D. SWEET, *Numerical methods for Toeplitz matrices*, Ph.D. thesis, University of Adelaide, Australia, May 1982.
- [58] W. TRENCH, *An algorithm for the inversion of finite Toeplitz matrices*, SIAM J. Appl. Math., 12 (1964), pp. 515-522.
- [59] T. J. ULRYCH, D. E. SMYLIC, O. G. JENSEN AND G. K. C. CLARKE, *Predictive filtering and smoothing of short records by using maximum entropy*, J. Geophys. Res., 78 (1973), pp. 4959-4064.
- [60] G. WALKER, *On periodicity in series of related terms*, Proc. Royal Soc. London A, 131 (1931), p. 518.
- [61] N. WIENER, *Extrapolation, Interpolation and Smoothing of Stationary Time Series, with Engineering Applications*, Technology Press and Wiley, New York, 1949 (originally published in 1941 in a technical report).
- [62] R. A. WIGGINS AND E. A. ROBINSON, *Recursive solution to the multichannel filtering problem*, J. Geophys. Res., 70 (1965), pp. 1885-1891.
- [63] J. WILKINSON, *The Algebraic Eigenvalue Problem*, Oxford Univ. Press, London, 1965.
- [64] L. WUYTACK, *Padé Approximation and Its Applications*, Springer-Verlag, New York, 1979.
- [65] G. U. YULE, *On a method of investigating periodicities in disturbed series, with special reference to Wolfer's sunspot numbers*, Phil. Trans. Royal Soc. London A, 226 (1977), pp. 267-298.
- [66] S. ZOHAR, *Toeplitz matrix inversion: the algorithm of W. F. Trench*, J. Assoc. Comput. Mach., 16 (1969), pp. 592-601.
- [67] ———, *The solution of a Toeplitz set of linear equations*, J. Assoc. Comput. Mach., 21 (1974), pp. 272-276.

## ON BARTKY'S METHOD FOR EVALUATION OF INTEGRALS OF ELLIPTIC TYPE WITH APPLICATION TO ROUND-NOSED WEDGES\*

VERNON J. ROSSOW†

**Abstract.** Bartky's method for the numerical evaluation of integrals of the complete elliptic type is modified to improve its accuracy. The resulting modified method is then applied to integrals involving complex arithmetic which map the upper half of one plane onto either the interior or exterior of round-nosed wedges. These solutions have engineering applications to such problems as the design of magnets and the flow of fluids in and around corners.

**Key words.** Bartky's method, numerical evaluation of integrals, mapping of round-nosed wedges

**1. Introduction.** The analysis of many scientific problems leads to integrals that must be evaluated numerically to obtain results for engineering purposes. Bartky [2] introduced an effective transformation method for evaluating a large class of integrals which does not rely on finding clever algebraic relationships; e.g., King [9]. Unfortunately, the procedure as presented by Bartky is subject to severe loss of accuracy. Two modifications of Bartky's original method are presented here which are effective in maintaining accuracy. The first modification simplifies the procedure, and the second avoids the propagation and growth of truncation and round-off error. The revised technique is then applied to integrals which map the upper half of one plane onto either the inside or outside of round-nosed wedges. Such a mapping has applications in various physical problems that occur in electromagnetics and fluid dynamics (see [3], [4], [6]-[8], [12]-[14]).

**2. Alleviation of numerical inaccuracy in Bartky's method.** Bartky [2] considers integrals of the type

$$(2.1) \quad I(m_0, n_0) = \int_0^{\pi/2} \frac{F_0(R_0) d\varphi_0}{R_0}.$$

The integrand function  $F_0(R_0)$  is assumed to be a continuous function of a parameter  $R_0$  defined by

$$R_0^2 = m_0^2 \cos^2 \varphi_0 + n_0^2 \sin^2 \varphi_0,$$

where  $m_0$  and  $n_0$  are positive real numbers. Bartky rewrote the integral as

$$(2.2) \quad I(m_0, n_0) = \int_{n_0}^{m_0} \frac{F_0(R_0) dR_0}{\sqrt{(R_0^2 - n_0^2)(m_0^2 - R_0^2)}}$$

and then repeatedly applied Landen's transformation

$$(2.3) \quad R_{i+1} = \frac{1}{2} \left( R_i + \frac{m_i n_i}{R_i} \right)$$

beginning with  $i = 0$ . Landen's transformation has the property that  $R_{i+1}$  equals  $m_{i+1}$  at both limits of integration where  $R_i = m_i$  and  $n_i$ . Since the integration parameter  $R_{i+1}$  has its maximum excursion from  $m_{i+1}$  at  $R_{i+1} = \sqrt{m_i n_i} = n_{i+1}$ , which occurs when  $R_i = \sqrt{m_i n_i}$ , the substitution (2.3) folds the integration intervals at  $n_i$ . The integral (2.2)

\* Received by the editors July 14, 1982, and in revised form July 18, 1983.

† Low Speed Aircraft Research Branch, NASA Ames Research Center, Moffett Field, California 94035.

may then be expressed as

$$(2.4) \quad I(m_0, n_0) = I(m_i, n_i) = \int_{n_i}^{m_i} \frac{F_i(R_i) dR_i}{\sqrt{(R_i^2 - n_i^2)(m_i^2 - R_i^2)}} = \int_0^{\pi/2} \frac{F_i(R_i) d\varphi_i}{R_i},$$

where

$$(2.5) \quad m_{i+1} = \frac{1}{2}(m_i + n_i) \quad \text{and} \quad n_{i+1} = \sqrt{m_i n_i}$$

are the successive arithmetic and geometric means. The integrand functions,  $F_i$ , are based on the inverse of Landen's transformation which is obtained from (2.3) as (in Bartky's notation)

$$(2.6) \quad R_i = R_{i+1} \pm \sqrt{R_{i+1}^2 - n_{i+1}^2},$$

so that  $F_{i+1}$  is related to  $F_i$  by

$$(2.7) \quad F_{i+1}(R_{i+1}) = \frac{1}{2}[F_i(R_i + \sqrt{R_i^2 - n_i^2}) + F_i(R_i - \sqrt{R_i^2 - n_i^2})].$$

In effect, the process of going from the  $i$ th level to the  $(i + 1)$ th level folds the integral (2.2) or (2.4) once so that the integrand becomes the arithmetic mean of two functions which are related to  $R_{i+1}$  by the inverse of Landen's transformation.

The integral is more difficult to manage analytically in this form but repetition of the folding process leads to a situation where  $m_i$  and  $n_i$  are equal to each other to any desired number of significant figures. The integration parameter  $R_i$  and, consequently, the integrand function  $F_i(R_i)$  are then approximately constant over the integration interval so that the integral may be written as

$$(2.8) \quad I(m_0, n_0) = \int_0^{\pi/2} \frac{F_i(R_i) d\varphi_i}{R_i} \Big|_L = \frac{F_L(R_L)}{R_L} \frac{\pi}{2} + \varepsilon_L.$$

The quantity  $\varepsilon_L$  can be made as small as desired by increasing the level  $L$  of the approximation, to render  $R_i$  and consequently  $F_i(R_i)$  more nearly constant over the integration interval.

The procedure used in the numerical evaluation of (2.8) begins with  $m_0$  and  $n_0$  and then determines the arithmetic and geometric means,  $m_i$  and  $n_i$ , up to a chosen level of agreement,  $i = L$ . Since the integrand function,  $F_L(R_L)$ , is known only in terms of  $i = 0$ , it must be expanded from  $i = L$  down to  $i = 0$  by repeated application of (2.7). This process requires a corresponding expansion in  $R_i$  as indicated in (2.6) and (2.7). While a single subscript suffices for  $F_i$ , because it has the same form at each level, the integration parameter,  $R$ , needs a double subscript to properly index the multiple values it assumes at each level. The multiple values are brought about by the two roots indicated in (2.6) by the  $\pm$  sign. Therefore, the number of values of  $R$  doubles at each level below  $L$ . The indexing used here is then

$$(2.9) \quad R_{i-1, k_{i-1}} = R_{i, k_i} \pm \sqrt{R_{i, k_i}^2 - n_i^2},$$

where the subscript  $i$  is again associated with the level being considered at that point in the analysis, and  $k_i$  identifies the various values of the integration parameter at that level. The computation of the  $R_{i, k_i}$  begins with  $i = L$  and

$$(2.10) \quad R_{L,1} = m_L \quad \text{and} \quad R_{L,2} = n_L,$$

and progresses down to  $i = 0$  by repeated application of (2.9). After the integration parameters have been evaluated, the integral is determined by

$$(2.11) \quad I(m_0, n_0) \cong \frac{\pi}{R_L 2^{L+1}} \sum_{k_0=1}^{2^L} F_0(R_{0, k_0}).$$

Equation (2.11) incorporates the expansion of the integration functions from  $i = L$  to  $i = 0$  using (2.7).

It first appears that the  $L$ th approximation is independent of all the lower ones. A relationship does however exist between the various levels which becomes apparent when the integrand functions based on the arithmetic mean are written sequentially. By use of (2.7) and the relationships,

$$m_i = m_{i+1} + \sqrt{m_{i+1}^2 - n_{i+1}^2} \quad \text{and} \quad n_i = m_{i+1} - \sqrt{m_{i+1}^2 - n_{i+1}^2},$$

successive approximations to the integrand function are given by

$$\begin{aligned}
 F_1(m_1) &= \frac{1}{2}[F_0(m_0) + F_0(n_0)], \\
 F_2(m_2) &= \frac{1}{2}[F_1(m_1) + F_1(n_1)], \\
 &\vdots \\
 F_{i+1}(m_{i+1}) &= \frac{1}{2}[F_i(m_i) + F_i(n_i)].
 \end{aligned}
 \tag{2.12}$$

Hence, increasingly higher approximations to the integrand function may be obtained by use of the recursion formula (2.12). Each higher approximation is determined by evaluating only  $F_i(n_i)$  and averaging it with the previous approximation based on  $m_i$  so that the integration parameters  $R_{i,k_i}$  based on  $m_i$  need not be evaluated.

It is now noted that when  $i = L$ , the result

$$F_{L+1}(m_{L+1}) = \frac{1}{2}[F_L(m_L) + F_L(n_L)]$$

could be used in (2.8) to evaluate the integral. Since  $m_L$  and  $n_L$  approach the same value as  $L$  increases, only a small error (which can be made negligibly small) would occur if  $F_L$  in (2.8) were based on  $m_L$ ,  $n_L$  or both. The three possibilities are then

$$F_L(R_{L,k_L}) = F_L(m_L) + \epsilon_m = F_L(n_L) + \epsilon_n = \frac{1}{2}[F_L(m_L) + F_L(n_L)] + \epsilon.$$

These three choices for evaluating  $F_L$  are equivalent in the limit as  $L$  gets large, since then  $\epsilon_m$ ,  $\epsilon_n$  and  $\epsilon$  become negligibly small.

The numerical effort is smallest when the solution is based on only  $F_L(n_L)$  because the lower approximations of  $F_i(m_i)$  do not then need to be determined for the evaluation of  $F_L(m_L)$ . Furthermore, since (2.12) shows that  $F_L(m_L)$  is based on  $F_{L-1}$ , its convergence rate lags behind that of  $F_L(n_L)$ . Hence, if the computations are to be based on one level of approximation only, it is more efficient to exclude  $F_L(m_L)$  and to use the expression

$$F_L(R_{L,k_L}) = F_L(n_L). \tag{2.13}$$

Because of the foregoing reasons, the examples presented in the sequel were calculated using (2.13).

The integration parameters,  $R_{i,k_i}$ , which originate with  $n_L$ , must now be calculated in order to evaluate the integrand function,  $F_L(n_L)$ . When these computations were undertaken for the mappings presented in the sequel, it was found that the method suggested by Bartky [2], (2.9) needed to be modified because an inaccuracy arises from the square root term. This inaccuracy develops at the beginning of the computations where

$$R_{L,1} = n_L \quad \text{and} \quad R_{L-1,k_{L-1}} = n_L \pm \sqrt{n_L^2 - n_{L-1}^2}. \tag{2.14}$$

In cases where the arithmetic/geometric mean has been determined to a high degree of agreement (e.g.,  $L \geq 4$ ), the values of  $n_L$  and  $n_{L-1}$  also agree to a comparable number

TABLE 1  
Effect of initial error on computed value of integration parameter.

i	m <sub>i</sub>	n <sub>i</sub>	R <sub>i,1</sub>	
			No initial error	+0.000001 error in R <sub>4,1</sub>
4	1.340933	1.340933	1.340933	1.340934
3	1.340933	1.340933	1.340933	1.342664
2	1.341050	1.340817	1.340933	1.410818
1	1.366025	1.316074	1.358568	1.849699
0	1.000000	1.732051	1.695698	3.148443

of significant figures. Therefore, when the subtraction under the square root sign in (2.14) is made, most if not all of the significant figures in the computer are lost to yield a nearly meaningless remainder much less than one. The square root of this number introduces uncertainty into the right half of the figures defining  $R_{L-1,k_{L-1}}$ . Since  $R_{L-1,k_{L-1}}$  is used to compute  $R_{L-2,k_{L-2}}$ , etc., the uncertainty or inaccuracy is passed on to all of the  $R_{i,k_i}$  from  $i = L - 1$  down to  $i = 0$ . This growth of the error due to the square root and its transfer to subsequent values of  $R_{i,1}$  is illustrated in Table 1. The two results for  $R_{i,1}$  presented there differ because the second is assumed to have an error of 0.000001 (e.g., due to round off) in  $R_{4,1}$ . The divergence between the two numerical results shows how an error at the beginning can grow and propagate through the computations of  $R_{i,1}$ . Naturally, the same phenomena can occur with the other  $R_{i,k_i}$  so that a small error incurred near the beginning of the computations is capable of destroying all of the significant figures in the result for the integral. Errors which occur in later parts of the calculations are not as devastating but the loss in accuracy is often not tolerable.

This rapid loss of accuracy in the computation of  $R_{i,k_i}$  can be alleviated by reformulating the square root quantities through the use of the arithmetic and geometric means and their difference

$$(2.15) \quad c_i = \frac{1}{2}(m_i - n_i).$$

Equation (2.15) can be combined with (2.5) to yield

$$(2.16) \quad c_i = \frac{c_{i-1}^2}{4m_i}.$$

The square root in (2.14) can then be rewritten as

$$(2.17) \quad \sqrt{n_L^2 - n_{L-1}^2} = c_{L-1} \sqrt{\frac{n_{L-1}}{2m_L}},$$

which retains the same accuracy as  $n_L$  because the difference under the square root has been replaced by a stable calculation. Evaluation of  $R$  can now be made accurately and systematically by use of recursion relationships which follow. Expansion of  $R$  by repeated use of (2.14) and (2.17) produces the expression  $(n_L^2 - n_{L-j}^2)$  which is common to all levels under the square root. Extension of the procedure used to obtain (2.17) yields

$$n_L^2 - n_{L-j}^2 = (n_L^2 - n_{L-1}^2) + (n_{L-1}^2 - n_{L-2}^2) + \dots + (n_{L-j+1}^2 - n_{L-j}^2)$$



or,

$$(2.18a) \quad n_L^2 - n_{L-j}^2 = c_{L-j}^2 P_{L-j},$$

where

$$(2.18b) \quad P_{L-j} = \left[ \frac{(1/2)c_{L-j}P_{L-j+1} + n_{L-j-1}}{2m_{L-j}} \right]$$

and

$$(2.18c) \quad P_L = \frac{n_{L-1}}{2m_L}.$$

Introduction of the  $P_{L-j}$  parameters into (2.14) for  $R_{L-1,k_{L-1}}$  illustrates the proliferation of  $\pm$  signs,

$$R_{L-1,k_{L-1}} = n_L \pm c_{L-2} \left[ \frac{c_{L-2}}{4m_{L-1}} + \sqrt{P_{L-1} \pm \frac{n_L}{2m_{L-1}} \sqrt{P_L} + \frac{c_{L-1}}{4m_{L-1}} P_L} \right],$$

where the subscript  $k_{L-1}$  serves as the index for the four values of  $R$  determined by the four combinations of  $\pm$  signs. Repeated application of the equations for  $R$  leads to the identification of a second recursion relationship which helps to keep the computations orderly.

$$(2.19) \quad Q_{L-j,k_{L-j}} = \frac{c_{L-j-1}}{4m_{L-j}} \pm \sqrt{P_{L-j} + \frac{n_L \sqrt{P_{L-j+1}}}{2m_{L-j}} + \frac{c_{L-j} P_{L-j+1}}{4m_{L-j}}},$$

where

$$(2.20) \quad Q_{L,1} = +\sqrt{P_L} \quad \text{and} \quad Q_{L,2} = -\sqrt{P_L}.$$

After the  $Q$  parameters have been evaluated from  $Q_{L,k_L}$  down to  $L-j=1$ , or  $Q_{1,k_1}$ , the integration parameters used in (2.12) for the evaluation of the integrand functions are found by

$$(2.21) \quad R_{0,k_0} = n_L + c_1 Q_{1,k_1}.$$

The arithmetic mean of the integrand functions  $F$  evaluated at the  $k_0 = 2^{L-1}$  locations  $R_{0,k_0}$  on the integration interval over  $R$  yields a numerical value for the integral  $I(m_0, n_0)$ . The foregoing form of the recursion equations for  $R$  avoids the dramatic loss of significant figures which arise from the use of the simpler square root version of  $R$ . Such a technique permits Bartky's method to be applied to any level of approximation with no more than the usual loss of significant figures.

**3. Application to mapping of round-nosed wedges.** Bartky's method with the foregoing two modifications will now be used to obtain numerical results for an integral which is often of practical interest in engineering and physics. The integral is part of a function that maps the upper half of one plane (i.e., the original plane) onto another plane as the interior or exterior of a round-nosed wedge (see Fig. 1). The included angle,  $2\theta$ , between the sides of the wedge can be of any value between  $0^\circ$  and  $180^\circ$ . The circular arc which comprises the rounded nose of the wedge has therefore a corresponding angular content between  $180^\circ$  and  $0^\circ$ . Such a shape approximates the boundaries of a number of devices where the region on either side of the boundary can be represented by differential equations such as Laplace's equation (see e.g., [3], [6], [7], [8], [13] and [14]). For example, the limiting case where the angular opening of the wedge is zero has been analyzed using complete elliptic integrals [6], [7] to

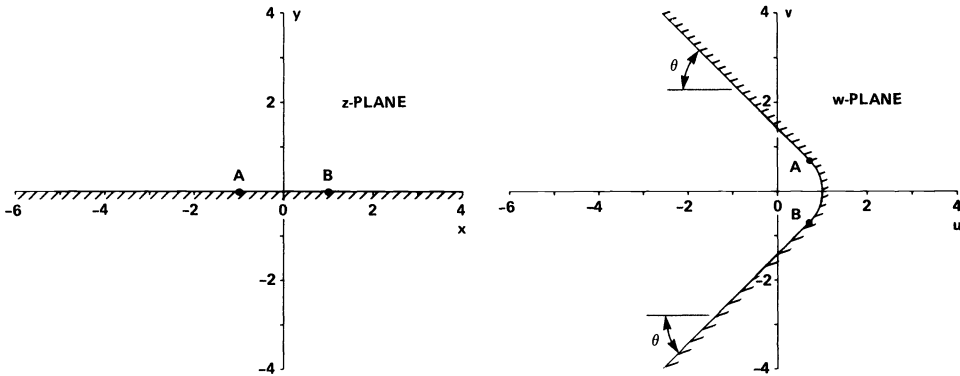


FIG. 1. Nomenclature used to map upper half of  $z$ -plane onto interior of round-nosed wedge in  $w$ -plane.

determine the electromagnetic characteristics of such a shape. In another example, elliptic integrals have been used to analyze the flow around airfoils whose shapes were composed of three circular arc segments [8], [14]. Although very limited numerical data are presented for these examples, they illustrate the background and the need for a general method which is capable of generating the numerical data accurately and easily.

**3.1. Derivation of a mapping function.** Various books and research papers ([3], [12], [13], for example) treat the theory of conformal mappings which uses complex variables to develop geometrical relationships between two planes. The technique to be used here is a special case of the conformal mapping that transforms the upper half of one plane onto the interior area of a polygon composed of circular arcs (i.e., a curvilinear polygon) in another plane ([3, pp. 251ff] or [12, pp. 198ff]). Such a mapping is an extension of the Schwartz–Christoffel theorem [3], [12], [13] which maps the upper half plane onto the interior of straight-sided polygons and which is based on the differential relationship

$$\frac{f''(z)}{f'(z)} = g(z) = \sum_{\nu=1}^n \frac{\mu_{\nu}}{a_{\nu} - z},$$

where a prime indicates a derivative with respect to  $z$ . The quantity  $f(z)$  is the mapping function for straight-sided polygons and  $g(z)$  determines through  $a_{\nu}$ , the lengths of the sides of the polygon and through  $\mu_{\nu}$ , the angles at which the sides intersect. When the Schwartz–Christoffel theorem for polygons with straight sides [ $f(z)$ ] is extended to a function  $w(z)$  which maps a straight line into polygons composed of segments of circles, the mapping is developed through the Schwarzian or the Schwarzian differential parameter [3], [12] which is written as

$$(3.1) \quad \left( \frac{w''(z)}{w'(z)} \right)' - \frac{1}{2} \left( \frac{w''(z)}{w'(z)} \right)^2 = \frac{1}{2} \sum_{\nu=1}^n \frac{1 - \alpha_{\nu}}{(a_{\nu} - z)^2} - \sum_{\nu=1}^n \frac{\beta_{\nu}}{a_{\nu} + z} + C,$$

where  $\pi\alpha_{\nu}$  is an interior angle at the intersection of two circular arcs of the polygon and the  $a_{\nu}$ 's again determine the lengths of the sides. The constants  $\beta_{\nu}$  are related to both  $\alpha_{\nu}$  and  $a_{\nu}$ . The general case is usually regarded as very difficult, but a curvilinear triangle composed of three circular arcs is tractable. For such a figure, the mapping

function is defined by

$$(3.2) \quad \left(\frac{w''}{w'}\right)' - \frac{1}{2}\left(\frac{w''}{w'}\right)^2 = \frac{1}{2(z-a)(z-b)(z-c)} \cdot \left[ \frac{(1-\alpha^2)(a-b)(a-c)}{z-a} + \frac{(1-\beta^2)(b-a)(b-c)}{z-b} + \frac{(1-\gamma^2)(c-a)(c-b)}{z-c} \right],$$

where  $w = u + iv$ , and a prime indicates a derivative with respect to  $z$ . In the mapping to be studied here,  $\alpha = 1, \beta = 1, \gamma = 2\theta/\pi, a = -1, b = +1$  and  $c = \pm\infty$  as illustrated in Fig. 1. The differential equation to be solved is then,

$$(3.3) \quad \left(\frac{w''}{w'}\right)' - \frac{1}{2}\left(\frac{w''}{w'}\right)^2 = \frac{1-(2\theta/\pi)^2}{2(z^2-1)}.$$

The assumption is now made that the mapping function can be represented as the ratio of two functions

$$(3.4) \quad w = \frac{W_1}{W_2} = \frac{ah_1 + bh_2}{ch_1 + bh_2},$$

where  $a, b, c$  and  $d$  are constants to be determined by the boundary conditions. Substitution of (3.4) into (3.3) shows that both functions  $W_1$  and  $W_2$  must satisfy the differential equation (with  $\gamma = 2\theta/\pi$ )

$$(3.5) \quad (z^2-1)W'' + (1-\gamma^2)\frac{W'}{4} = 0.$$

The two solutions,  $h_1$  and  $h_2$ , for the function  $W$  in the differential equation (3.5) are found by relating (3.5) to the hypergeometric differential equation [11, p. 7]. When the integral form of the solutions is integrated by parts once, it may be written as

$$h = C \int_0^1 \tau^\gamma (1-\tau^2)^{(1-\gamma)/2} (1-\zeta\tau^2)^{-(1-\gamma)/2} d\tau$$

or, if  $\tau = \sin \varphi$ ,

$$(3.6) \quad h = C\zeta \int_0^{\pi/2} \frac{(\tan \varphi)^\gamma \cos^2 \varphi d\varphi}{(1-\zeta \sin^2 \varphi)^{(1-\gamma)/2}} = C \int_0^{\pi/2} \frac{F(R)}{R} d\varphi,$$

where  $C$  is a constant and  $\zeta = (1-z)/2$  for  $h_1$ , and  $\zeta = (1+z)/2$  for  $h_2$ . The parameter  $R$  is given by

$$R^2 = 1 - \zeta \sin^2 \varphi = \cos^2 \varphi + (1-\zeta) \sin^2 \varphi$$

so that

$$\cos^2 \varphi = \frac{R-1+\zeta}{\zeta}, \quad \tan^2 \varphi = \frac{1-R^2}{R^2-1+\zeta}$$

which yield

$$F(R) = R^\gamma (R^2-1+\zeta) \left[ \frac{(1-R^2)}{(R^2-1+\zeta)} \right]^{\gamma/2}$$

as the integrand function. These substitutions convert (3.6) into the Bartky form shown in (2.1).

In the limiting case,  $\gamma = 0$ , the integral reduces to

$$(3.7) \quad h = C \int_0^1 \left[ \frac{(1-\tau^2)}{(1-\zeta^2)} \right]^{1/2} d\tau, \quad h = C[E(\sqrt{\zeta}) - (1-\zeta)K(\sqrt{\zeta})],$$

where  $K$  and  $E$  are the complete elliptic integrals of the first and second kinds of modulus  $\sqrt{\zeta}$ . The mapping function derived from (3.7) is equivalent to the one found in [7].

The general expression for the mapping function  $w$  is now found as a combination of the two integrals  $h_1$  and  $h_2$  in (3.6), i.e.,

$$(3.8) \quad w = \frac{h_1 + h_2 e^{i\pi(1-\gamma)/2}}{h_1 e^{i\pi(1-\gamma)/2} + h_2}.$$

The exponential function  $e^{i\pi(1-\gamma)/2}$  fixes the junctures of the curvilinear triangle as indicated in Fig. 1.

**3.2. Application of mapping integrals to round-nosed wedges.** The mapping function equation (3.8) can be numerically calculated by evaluation of the integrals (3.6) using Bartky's method as modified in § 2. Bartky derived his method for real positive values of  $m$  and  $n$ , but nothing appears to prevent the application of his method to complex values. Nonetheless, in the early part of this investigation the technique was checked against known values for elliptic integrals by use of various expansions and transformations (e.g., [1], [2], [9], [10]) to be certain that Bartky's method as modified here would provide valid results when complex arithmetic is used on the computer.

The upper half of the original plane is mapped onto the interior or exterior of round-nosed wedges by means of (3.4) and (3.6). In order to insure that the point to be mapped is in the upper half of the original plane ( $\text{Im}(z) > 0$ ), the boundary line was chosen as  $y = +10^{-5}$ . In the computations, it was first necessary to find out what level of approximation  $L$  is needed to provide plotting accuracy (i.e., three or more significant figures). Since the method suggested in this paper for the calculation of the integration parameters,  $R_{i,k}$ , uses values for  $c_{i-j}$ ,  $m_{i-j}$  and  $n_{i-j}$  in the computations, the convergence rate is slower than the convergence of the arithmetic and geometric

TABLE 2  
Numerical values of the wedge angle as a function  
of  $L$ ,  $x = -5$ ,  $y = +10^{-5}$ .

$L$	Wedge angle, $\theta$	
	$\gamma = 2.0$	$\gamma = 1.5$
3	86.009°	19.165°
4	4.731°	57.481°
5	-0.001°	48.286°
6	0.000°	46.118°
7	0.000°	45.392°
8	0.000°	45.139°
9	0.000°	45.049°
10	0.000°	45.017°
11	0.000°	45.006°
exact	0°	45°

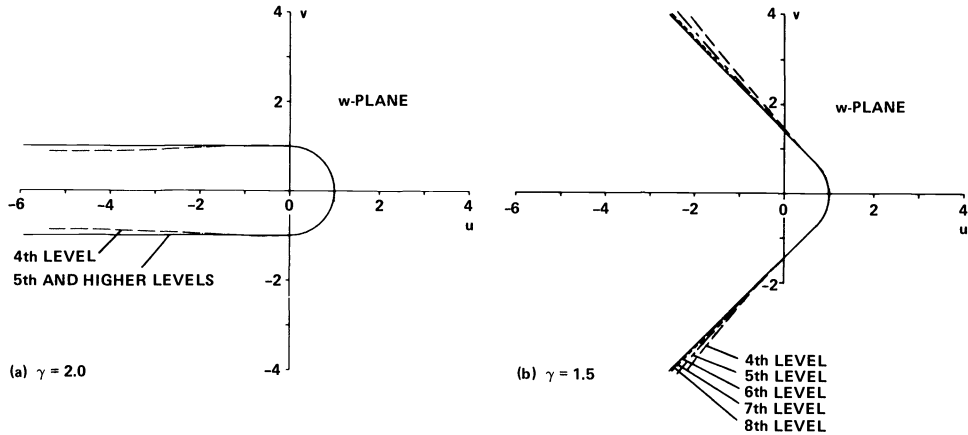


FIG. 2. Examples of the convergence rates of mapping integrals on the boundary for various approximations.

means to a common limit. It was also found that the convergence rate was dependent on the integral being considered. These characteristics are illustrated in Fig. 2 and in Table 2 where mapping data is presented for the boundary line of two wedge openings;  $\theta = 0^\circ$  and  $45^\circ$  (i.e.,  $\gamma = 2.0$  and  $1.5$ ). Figure 2 illustrates graphically how the boundary line converges to the limiting shape as the number of approximations  $L$  is increased. Table 2 presents numerical values for the angle of the wedge when a point at  $x = -5$ ,  $y = +10^{-5}$  in the original plane is mapped onto the plane of the round-nosed wedge for  $\theta = 0^\circ$  and  $45^\circ$ . Both Fig. 2a and the tabulated values in Table 2 indicate that when  $\gamma = 2.0$  the level of approximation,  $L = 5$ , provides plotting accuracy. However, when  $\gamma = 1.5$ ,  $L$  should be eight or greater.

The mapping function (3.8) was used on a grid of one size when  $\gamma$  is less than one, Fig. 3, and on a smaller grid when  $\gamma$  is greater than one, Fig. 4. Both the interior or exterior of the wedges are mapped from the upper half of the original or  $z$ -plane by choosing values of  $\gamma$  larger or smaller than one. Some of the curves stop at the edge of the plotting field rather than at the end of the grid presented in Figs. 3a and 4a.

**4. Concluding remarks.** The foregoing text first recommends that the evaluation of integrals by Bartky's method be based on only the geometric mean rather than on both the arithmetic and geometric means. Such a change simplifies the procedure used in the computations and makes them more efficient. A new method for the computation of the integration parameters was then introduced to circumvent the severe loss of accuracy caused by a square root term in Bartky's procedure. These changes broaden the applicability of the method and increase the accuracy of the method for a wide range of applications. Not mentioned in the previous text is that integrals which are not necessarily of the elliptic type can also be treated by Bartky's method. Even if  $R$  does not occur naturally in the denominator of the integrand, the basic integrand could be put in the Bartky form of  $F(R)/R$  by dividing and multiplying by  $R$ . The parameter  $R$  in the numerator is then simply combined with the integrand function  $F(R)$  and the computations conducted as described in the foregoing text. This possibility in addition to those formulations that can be derived from the manipulation of the integration variable and/or the limits of integration make it possible to evaluate a wide variety of integrals by Bartky's numerical method. Although the various integrand functions may have different rates of convergence, the modifications introduced here permit approximations to be carried out to any level with no more than the usual loss of significant figures in the calculations.

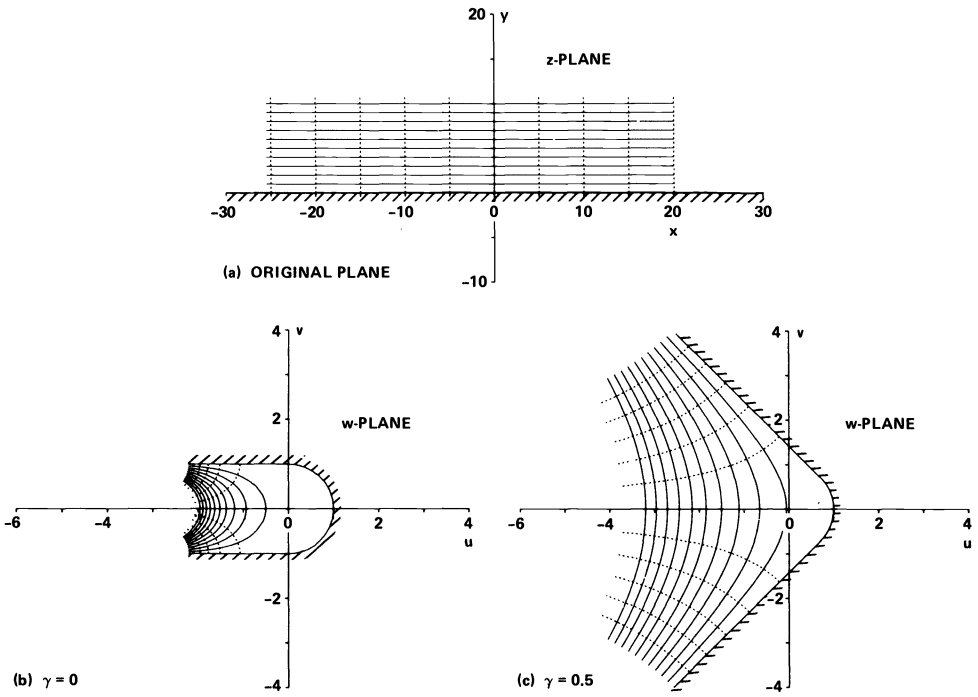


FIG. 3. Map of a grid in  $z$ -plane onto  $w$ -plane for two values of  $\gamma = 2\theta/\pi$ , less than one;  $L = 8$ .

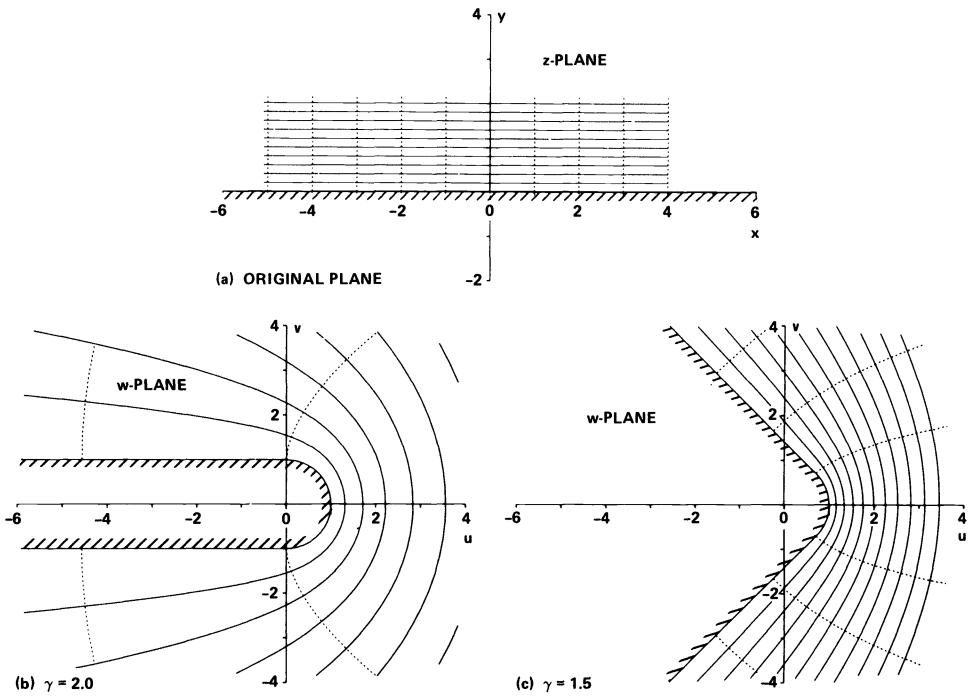


FIG. 4. Map of a grid in  $z$ -plane onto  $w$ -plane for two values of  $\gamma = 2\theta/\pi$ , greater than one;  $L = 8$ .

**Acknowledgment.** The author thanks David C. Galant for calling his attention to Bartky's method and for helpful discussions during the course of the investigation.

## REFERENCES

- [1] M. ABRAMOWITZ AND I. A. STEGUN, *Handbook of Mathematical Functions*, National Bureau of Standards, Appl. Math. Series, 55, Washington, DC, 1964.
- [2] W. BARTKY, *Numerical calculation of a generalized complete elliptic integral*, Rev. Modern Phys., 10 (1938), pp. 264-269.
- [3] A. BETZ, *Konforme Abbildung*, Springer-Verlag, Berlin, 1948.
- [4] R. BULIRSCH, *An extension of the Bartky transformation to incomplete elliptic integrals of the third kind*, Numer. Math., 13 (1969), pp. 266-284.
- [5] P. F. BYRD AND M. D. FRIEDMAN, *Handbook of Elliptic Integrals for Engineers and Physicists*, Springer-Verlag, Berlin, 1954.
- [6] N. DAVY AND N. H. LANGTON, *The two-dimensional magnetic or electric field inside a semi-infinite slot terminated by a semi-circular cylinder*, Brit. J. Appl. Phys., 3 (1952), pp. 156-158.
- [7] ———, *The external magnetic field of a single thick semi-infinite parallel plate terminated by a convex semi-circular cylinder*, Quart. J. Mech. Appl. Math., VI (1953), pp. 115-121.
- [8] S. D. DAYMOND AND J. HODGKINSON, *A type of aerofoil*, Quart. J. Math. (Oxford), 10 (1939), pp. 136-144.
- [9] L. V. KING, *On the Direct Numerical Calculation of Elliptic Functions and Integrals*, Cambridge Univ. Press, London, 1924.
- [10] W. MAGNUS, F. OBERHETTINGER AND R. P. SONI, *Formulas and Theorems for the Special Functions of Mathematical Physics*, Springer-Verlag, Berlin, 1966.
- [11] W. MAGNUS AND F. OBERHETTINGER, *Formulas and Theorems for the Functions of Mathematical Physics*, Chelsea, New York, 1949.
- [12] Z. NEHARI, *Conformal Mapping*, Dover, New York, 1952.
- [13] V. L. STREETER, *Fluid Dynamics*, McGraw-Hill, New York, 1948.
- [14] W. WOLFF, *Einfluss der Abrundung scharfer Eintrittskanten auf den Widerstand von Fluegeln*, Ingenieur-Archiv, IV (1933), pp. 521-544.

## THE NUMERICAL SOLUTION OF A NONLINEAR SYSTEM ARISING IN TIME SERIES ANALYSIS\*

DAVID H. ANDERSON† AND EUGENE C. GARTLAND, JR.‡

**Abstract.** This paper is concerned with the analysis and methods of solution of a particular system of nonlinear equations that arises in the identification of parameters for certain time series models. Necessary and sufficient conditions for the existence of real solutions to the system are given. Numerical solution methods include a generating function factorization method, Newton's method and the simplified Newton's method. Initial estimates for starting the algorithms are also investigated.

**Key words.** time series, parameter identification, generating functions, Newton's method

**AMS (MOS) subject classifications.** primary 65H10, secondary 65U05

**Introduction.** In time series modeling, one encounters the problem of numerically solving the nonlinear algebraic system  $F(\theta) = 0$ , where  $\theta \equiv (\theta_0, \dots, \theta_n)^T$  and the component functions of  $F$  are

$$(0.1) \quad F_i(\theta) \equiv \sum_{j=0}^{n-i} \theta_j \theta_{j+i} - c_i, \quad i = 0, \dots, n.$$

Here it is assumed that  $c_0, \dots, c_n$  are given real numbers where, in particular,  $c_0 > 0$  and  $c_n \neq 0$ . If  $c_n = 0$ , then the system  $F(\theta) = 0$  can be reduced in order.

In applications to time series analysis, the values  $c_0, \dots, c_n$  are estimates of the autocovariances of a moving average or (in the case of an assumed mixed autoregressive-moving average model) a derived moving average process. The system (0.1) must be solved for the real solution that is *invertible*, provided one exists—it is possible (depending on  $c_0, \dots, c_n$ ) that (0.1) will have no real solutions, and this must be determined.

Once obtained, the invertible solution is used to provide an initial guess for a nonlinear least squares routine that computes efficient estimates of the true moving average parameters. This entire model identification and estimation procedure is documented in Box and Jenkins (1970, §§ 6.3.2 and A6.2).

Hereafter we will simply refer to  $c_0, \dots, c_n$  as covariances, and the reader may think of them as the true autocovariances of a pure moving average process. In any event, it is the solution of (0.1), and not the entire model identification process, with which we are concerned.

The system (0.1) is a subclass of a more general system of the type

$$(0.2) \quad F_i(\theta) \equiv \sum_{j=0}^{p_i} \prod_{k=1}^{k_i} \theta_{q(i,j,k)} - c_i, \quad i = 0, \dots, n,$$

where each  $q(i, j, k)$  is an integer in  $\{0, 1, \dots, n\}$ ,  $p_i$  is a nonnegative integer, and  $c_i$  is a prescribed real number. The integer  $k_i$ ,  $1 \leq k_i \leq n$ , deserves special comment. A distinguishing feature of (0.2) is that for a given equation number  $i$ , the number  $k_i$  of factors in the product is the same for all  $j$  and thus is independent of the term number  $j$ . We see that  $k_i = 2$  for all  $j$  (and  $i$ ) in (0.1). Systems of nonlinear equations such as

\* Received by the editors August 11, 1982, and in revised form December 2, 1983.

† Department of Mathematics, Southern Methodist University, Dallas, Texas 75275; and Department of Medical Computer Science, University of Texas Health Science Center, Dallas, Texas 75235.

‡ Department of Mathematics, Southern Methodist University, Dallas, Texas 75275.



(0.2) arise in structural identification problems of compartmental analysis (see Anderson (1982), Delforge (1977) and Cobelli et al. (1979)).

The purpose of this article is to analyze real solutions of (0.1). The paper is expository in that it reviews work done in the past on these equations. Yet new results are also presented in a variety of areas: (1) new proofs of necessary and sufficient conditions for there to exist real solutions; (2) a new polynomial solution method; (3) a global convergence result for Newton's method which can be applied even where only a borderline invertible solution exists and the Jacobian  $F'(\theta)$  is known to be singular at this solution; (4) local existence and uniqueness of solutions obtained by the simplified Newton method (for which we prove convergence results subject to conditions on the  $c_0, \dots, c_n$  that are easy to check and only slightly more restrictive than a similar condition sufficient to guarantee existence of invertible solutions); (5) improved starting vectors for which algorithm convergence is ensured. Moreover, numerical results are reported, and the relative merits of the proposed schemes are discussed.

The solution of interest to time series analysts is the (real) invertible solution. The importance of invertible solutions is that the requirement of invertibility is needed if we are to associate present events with past happenings in a sensible manner in the time series model; that is, with invertibility we wish to exclude physical models that do not make sense (see Box and Jenkins (1970, pp. 50-51)). A simple procedure, which is based upon isolating roots of polynomials via Sturm sequences, to determine the existence of invertible solutions is given.

We hope that this paper will be useful to specialists in time series as a summary of relevant mathematical results and techniques, and to applied mathematicians by providing them with an important application of numerical and computational techniques to time series analysis.

**1. Necessary conditions for solutions.** To begin our discussion of (0.1), necessary conditions for the existence of real solutions of the problem are given. The quadratic nature of the equations suggests the possibility of using to our advantage forms like  $\langle \theta, N\theta \rangle$ , where  $\langle \cdot, \cdot \rangle$  is the usual inner product on  $R^{n+1}$ . In fact, if we let  $N$  be the  $(n+1) \times (n+1)$  matrix with ones on the superdiagonal and zeros elsewhere, then (0.1) can be rewritten as

$$(1.1) \quad \langle \theta, N^i \theta \rangle - c_i = 0, \quad i = 0, 1, \dots, n,$$

where  $\theta$  is any real solution of (0.1). With this form of (0.1), investigation of necessary conditions can proceed in at least two ways.

The first comes from the addition of the equations of (1.1) to arrive at the single equality

$$(1.2) \quad \langle \theta, A\theta \rangle = 2 \sum_{i=0}^n c_i,$$

where  $A$  is the  $(n+1)$ -square matrix with twos on its diagonal and ones elsewhere. Since a real solution of (1.1) cannot be the zero-vector, and because  $A$  is positive definite, then (1.2) tells us that a preliminary necessary condition for the existence of a real solution  $\theta^*$  of (1.1) is that  $\sum_{i=0}^n c_i > 0$ . This condition is now refined by appealing to the Rayleigh principle. The eigenvalues of  $A$  are  $n+2, 1, 1, \dots, 1$ , and by utilizing (1.1) and (1.2) the Rayleigh quotient is

$$(1.3) \quad R(\theta^*) \equiv \langle \theta^*, A\theta^* \rangle / \langle \theta^*, \theta^* \rangle = \left( 2 \sum_{i=0}^n c_i \right) / c_0.$$

Hence the inequalities of Rayleigh's principle,

$$(1.4) \quad 1 = \min_{\theta \neq 0} R(\theta) \leq R(\theta^*) \leq \max_{\theta \neq 0} R(\theta) = n + 2,$$

direct us towards the first solvability condition.

**THEOREM 1.1.** *If there exists a real solution of (0.1), then the inequalities*

$$(1.5) \quad 1 \leq \left( 2 \sum_{i=0}^n c_i \right) / c_0 \leq n + 2$$

*must be satisfied by the scalars  $c_i$ .*

We should also note that, according to the Rayleigh principle, the left-hand inequality of (1.5) becomes an equality for just those  $\theta^*$  whose components satisfy  $\sum_{j=0}^n \theta_j^* = 0$ . Likewise, the right-hand side of (1.5) is an equality provided  $\theta_j^* = 1$  for all  $j = 0, 1, \dots, n$ .

Anderson (1975) investigates bounds for the autocorrelations defined by

$$(1.6) \quad \rho_i \equiv \sum_{j=0}^{n-i} \theta_j \theta_{j+i} / \sum_{j=0}^i \theta_j^2, \quad i = 1, 2, \dots, n,$$

and  $\rho_0 \equiv 1$ . For these autocorrelations taken as a whole, he states (without proof) that

$$(1.7) \quad -\frac{1}{2} \leq \sum_{i=1}^n \rho_i \leq \frac{n}{2}.$$

At a real solution vector  $\theta^*$  of (0.1), his  $\rho_i$  is our ratio  $c_i/c_0$ , and it is then an easy calculation to show that (1.7) is equivalent to our condition (1.5).

A second way to develop a necessary condition is to observe that (1.1) implies

$$(1.8) \quad \langle \theta, [N^k + (N^k)^T] \theta \rangle = 2c_k, \quad k = 0, 1, \dots, n.$$

Consider the maximum eigenvalue  $r(M_k)$  of the real symmetric nonnegative matrix

$$M_k \equiv N^k + (N^k)^T, \quad k = 0, 1, \dots, n.$$

When  $k = 0$ ,  $M_0 = 2I$  and  $r(M_0) = 2$ . When  $k = 1$ ,  $M_1$  is tridiagonal and irreducible, and the Perron root  $r(M_1) = 2 \cos(\pi/(n+2))$ . When  $2 \leq k \leq n$ ,  $M_k$  is reducible and its normal form can be determined using directed graphs (see Lancaster (1969)). In this case,  $M_k$  is similar to the irreducible block diagonal matrix  $\text{diag}\{B, B, \dots, B, C\}$ , where each block matrix  $B$  or  $C$  is a square matrix with ones on the superdiagonal and subdiagonal, and zeros elsewhere. Let  $[n/k]$  denote the greatest integer in  $n/k$ . Then  $B$  is of dimensions  $([n/k]+1) \times ([n/k]+1)$ , and the last block  $C$  is  $s \times s$ , where  $s \leq [n/k]+1$ . Thus the maximal eigenvalue  $r(M_k)$ ,  $2 \leq k \leq n$ , is

$$r(M_k) = 2 \cos \frac{\pi}{[n/k]+2}.$$

The above arguments combined with another application of the Rayleigh principle produce the following second necessary condition on real solutions of  $F(\theta) = 0$ .

**THEOREM 1.2.** *If there exist real solutions of (0.1), then the inequalities*

$$(1.9) \quad |c_k| \leq c_0 \cos \frac{\pi}{[n/k]+2}, \quad k = 1, 2, \dots, n,$$

*must be satisfied by the scalars  $c_k$ .*

Anderson (1974) gives the equivalent bound

$$|\rho_k| \leq \cos \frac{\pi}{[n/k] + 2}, \quad k = 1, 2, \dots, n.$$

These maximum possible autocorrelations also seem to have been discovered independently by Davies et al. (1974). Both derivations are different than that presented above. We might also note that inequality (1.9) immediately implies the more easily checked conditions

$$(1.10) \quad |c_k| \leq \begin{cases} c_0, & 1 \leq k \leq \frac{n}{2}, \\ c_0/2, & \frac{n}{2} < k \leq n. \end{cases}$$

**2. Nature of solutions.** Following Wilson (1969), we study the nature of solutions of (0.1) by using generating functions. With the vectors  $\theta = (\theta_0, \dots, \theta_n)^T$  and  $c = (c_0, \dots, c_n)^T$  associate the functions

$$\Theta(z) = \theta_0 + \theta_1 z + \dots + \theta_n z^n \quad \text{and} \quad C(z) = c_0 + c_1 \left(z + \frac{1}{z}\right) + \dots + c_n \left(z^n + \frac{1}{z^n}\right).$$

Then the nonlinear system  $F(\theta) = 0$  is equivalent to

$$(2.1) \quad C(z) = \Theta(z)\Theta\left(\frac{1}{z}\right),$$

that is, the vector  $\theta$  satisfies (0.1) if and only if the corresponding polynomial  $\Theta(z)$  satisfies (2.1). This is easily verified by expanding the right-hand side of the factorization above and equating the coefficients of like powers of  $z$ .

The function  $\overline{C(z)}$  has the properties that  $C(z) = \overline{C(1/\bar{z})}$  and, since the coefficients  $c_0, \dots, c_n$  are real,  $\overline{C(z)} = C(\bar{z})$ . The following properties of the roots of  $C(z) = 0$  are easily established.

**PROPOSITION 2.1.** *Let  $\alpha$  be a root of  $C(z) = 0$ . Then  $\bar{\alpha}$  and  $1/\alpha$  are also roots and all have the same multiplicity. If 1 or  $-1$  is a root of  $C(z) = 0$ , then it must have even multiplicity.*

These observations can be used to characterize the existence of real solutions to (0.1).

**THEOREM 2.2.** *There exist real solutions to  $F(\theta) = 0$  if and only if those roots of  $C(z) = 0$  of unit modulus have even multiplicity.*

*Proof.* Given that there exists a real solution  $\theta = (\theta_0, \dots, \theta_n)^T$ . Let  $\alpha$  be a root of  $C(z) = 0$  with modulus one. Then  $\alpha$  and  $\bar{\alpha}$  are roots of  $\Theta(z) = 0$ , where  $\Theta(z)$  is the polynomial associated with the vector  $\theta$ ; let their common multiplicity be  $p$ . Thus

$$C(z) = \Theta(z)\Theta\left(\frac{1}{z}\right) = (z - \alpha)^p (z - \bar{\alpha})^p \left(\frac{1}{z} - \alpha\right)^p \left(\frac{1}{z} - \bar{\alpha}\right)^p D_1(z),$$

and  $D_1(z)$  does not vanish at  $\alpha$  or  $\bar{\alpha}$ . But  $\bar{\alpha} = 1/\alpha$ , since  $|\alpha| = 1$ , so

$$C(z) = (z - \alpha)^{2p} (z - \bar{\alpha})^{2p} D_2(z).$$

Again  $D_2(z)$  is nonzero at  $\alpha$  and  $\bar{\alpha}$ , and it follows that the multiplicity of  $\alpha$  (and  $\bar{\alpha}$ ) is even.

Conversely, given that those roots of  $C(z) = 0$  of unit modulus have even multiplicity, then all of the roots can be listed  $\alpha_1, \dots, \alpha_n, 1/\alpha_1, \dots, 1/\alpha_n$ , where complex

roots in  $\{\alpha_1, \dots, \alpha_n\}$  occur in conjugate pairs of the same multiplicity. It follows that

$$(2.2) \quad C(z) = K(z - \alpha_1) \cdots (z - \alpha_n) \left(\frac{1}{z} - \alpha_1\right) \cdots \left(\frac{1}{z} - \alpha_n\right),$$

where

$$(2.3) \quad K = \frac{(-1)^n c_n}{\alpha_1 \cdots \alpha_n}.$$

The positivity of the constant term,  $c_0$ , of  $C(z)$  implies that of the constant  $K$  above. This can be seen as follows. First,

$$(z - \alpha_1) \cdots (z - \alpha_n) = z^n + \gamma_{n-1}z^{n-1} + \cdots + \gamma_0,$$

where  $\gamma_0, \dots, \gamma_{n-1}$  are real. Similarly,

$$\left(\frac{1}{z} - \alpha_1\right) \cdots \left(\frac{1}{z} - \alpha_n\right) = \left(\frac{1}{z}\right)^n + \gamma_{n-1}\left(\frac{1}{z}\right)^{n-1} + \cdots + \gamma_0.$$

So the constant term of  $(z - \alpha_1) \cdots (z - \alpha_n)((1/z) - \alpha_1) \cdots ((1/z) - \alpha_n)$  is

$$1 + \gamma_{n-1}^2 + \cdots + \gamma_0^2,$$

and this is positive. But the constant term of  $C(z)$  is  $K(1 + \gamma_{n-1}^2 + \cdots + \gamma_0^2)$ , and for this to be positive  $K$  must be positive.

Now we can take

$$(2.4) \quad \Theta(z) = K^{1/2}(z - \alpha_1) \cdots (z - \alpha_n).$$

Then  $C(z) = \Theta(z)\Theta(1/z)$ , and  $\Theta(z) = \theta_0 + \theta_1z + \cdots + \theta_nz^n$ , where  $\theta = (\theta_0, \dots, \theta_n)^T$  is a real solution of  $F(\theta) = 0$ .  $\square$

**3. Existence of an invertible solution.** A (necessarily real) solution  $\theta$  of (0.1) is called *invertible* if all of the roots of its associated polynomial  $\Theta(z)$  have modulus greater than one. In this case, the moving average process is equivalent to an (infinite order) autoregressive process that is stationary. The same construction as in Theorem 2.2 above can be used to characterize the existence of invertible solutions. This characterization appears to be well-known among time series analysts.

**THEOREM 3.1.** *There exists an invertible solution to  $F(\theta) = 0$  if and only if  $C(z) = 0$  has no roots of unit modulus. This solution is unique subject to the normalization  $\theta_0 > 0$ .*

The question of whether or not  $C(z) = 0$  has a root of unit modulus can be treated as follows. If  $|z| = 1$ , then  $z = \exp(i\phi)$ , and  $C(z) = 0$  is equivalent to

$$\begin{aligned} 0 = C(e^{i\phi}) &= c_0 + c_1(e^{i\phi} + e^{-i\phi}) + \cdots + c_n(e^{in\phi} + e^{-in\phi}) \\ &= c_0 + 2c_1 \cos \phi + \cdots + 2c_n \cos n\phi. \end{aligned}$$

If  $\alpha = \exp(i\phi)$  is a root, then so is  $1/\alpha = \exp(-i\phi)$ ; so it is sufficient to consider  $0 \leq \phi \leq \pi$  (we should note here that necessary and sufficient conditions for real solutions of (0.1) are that the  $c_j$  satisfy  $0 < c_0 + 2c_1 \cos \phi + \cdots + 2c_n \cos n\phi$  for all  $|\phi| \leq \pi$  (see Anderson (1975a, p. 137))).

Recall the definition of the Chebyshev polynomials  $T_k(x)$  in terms of the transformation  $x = \cos \phi$ :

$$T_k(\cos \phi) = \cos k\phi.$$

Define the  $n$ th degree polynomial  $\tilde{C}(x)$  by

$$\tilde{C}(x) = c_0 + 2c_1 T_1(x) + \dots + 2c_n T_n(x).$$

Then  $C(z) = 0$  has a root of unit modulus if and only if  $\tilde{C}(x) = 0$  has a root in  $[-1, 1]$ . The question of whether or not a real polynomial vanishes in an interval can be answered by the classical technique of Sturm sequences (see, for example, Henrici (1974)). One constructs a Sturm sequence from  $\tilde{C}(x)$  and  $\tilde{C}'(x)$  by successive polynomial long divisions. The number of distinct zeros of  $\tilde{C}(x)$  in  $[-1, 1]$  is equal to the difference between the number of strict sign variations in the sequence at  $x = -1$  and at  $x = 1$ .

A related technique, the application of which is easier but which does not give as much information, is provided by the Fourier-Budan theorem (see Henrici (1974)). To apply this theorem, construct the sequence of derivatives  $\{\tilde{C}(x), \tilde{C}'(x), \dots, \tilde{C}^{(n)}(x)\}$ ; the theorem states that the number of zeros (counted with multiplicity) of  $\tilde{C}(x)$  in  $[-1, 1]$  is equal to the difference between the number of strict sign variations in this sequence at  $x = -1$  and at  $x = 1$  minus an even integer. Thus if the difference is 0 or 1,  $\tilde{C}(x)$  has no root in the interval. See Henrici (1974) for examples.

There are other algorithms for determining when polynomials have roots in  $|z| \leq 1$ , or in  $|z| < 1$ , or the like. For instance, see Miller (1971) and references contained therein.

The discussion above leads to a sufficient condition for the existence of an invertible solution. While this condition is rather stringent, it is very easily checked.

**PROPOSITION 3.2.** *If the constants  $c_0, c_1, \dots, c_n$  satisfy*

$$|c_1| + \dots + |c_n| < \frac{1}{2}c_0,$$

*then  $F(\theta) = 0$  has a unique invertible solution.*

*Proof.* If  $x \in [-1, 1]$ , then  $|T_k(x)| \leq 1, k = 0, 1, \dots$ . So

$$|\tilde{C}(x)| \geq c_0 - 2(|c_1| + \dots + |c_n|), \quad -1 \leq x \leq 1,$$

and by assumption, this is positive. It follows that  $\tilde{C}(x)$  cannot vanish in  $[-1, 1]$  and  $C(z) = 0$  has no roots of modulus one.  $\square$

**4. Polynomial factorization method.** Turning now to the actual solution of (0.1), the first method to be considered is suggested by the above theory and consists of factoring the function  $C(z)$  as in (2.2) and then constructing the solution  $(\theta_0, \theta_1, \dots, \theta_n)^T$  via (2.3) and (2.4) using the elementary symmetric functions relating the roots  $\alpha_i$  to the polynomial coefficients  $\theta_k$ . The following steps can be used:

(a) The  $2n$  roots of  $C(z)$  are found (for instance, through the calculation of the zeros of the polynomial  $z^n C(z)$ ).

(b) In keeping with the results of § 2, these zeros are listed as  $\alpha_1, \dots, \alpha_n, 1/\alpha_1, \dots, 1/\alpha_n$ , where  $\alpha_1, \dots, \alpha_n$  are in decreasing order of magnitude and the complex roots in  $\alpha_1, \dots, \alpha_n$  occur in conjugate pairs.

(c) Compute  $K$  from (2.3).

(d) Compute the coefficients  $\theta_0, \theta_1, \dots, \theta_n$  of  $\Theta$  from knowledge of  $\alpha_1, \dots, \alpha_n, K$ , by utilizing the symmetric functions or the Newton identities.

This factorization method is feasible in practice provided  $n$  is not too large (as in the case with many of these problems); one need only have access to a good polynomial root finder. There can sometimes be a loss of accuracy in reconstructing the  $\theta_k$  from the roots  $\alpha_i$  through the symmetric functions, but this most often is only a problem for larger  $n$ -values. Another difficulty is that with this method there is no way to improve accuracy in the final  $\theta_k$ -estimates. However, in time series analysis,

this may be no problem since often the estimates for  $\theta_k$  are used as initial guesses for further numerical calculations anyway and are therefore not needed to high accuracy.

**5. Newton's method.** A standard method for the numerical solution of nonlinear systems of equations is Newton's method. Its use for the present problem is advocated by Wilson (1969). Let  $\theta^{(k)}$  and  $F'(\cdot)$  denote the  $k$ th Newton iterate and the Jacobian matrix. Newton's algorithm proceeds as follows:

$$(5.1) \quad \begin{aligned} &\text{given } \theta^{(0)} \\ &\theta^{(k+1)} = \theta^{(k)} - F'(\theta^{(k)})^{-1}F(\theta^{(k)}), \quad k=0, 1, \dots \end{aligned}$$

In Wilson (1969) it is shown (except for a couple of oversights) that if an invertible solution  $\theta^*$  exists and if the initial vector  $\theta^{(0)}$  satisfies certain mild conditions, then  $\theta^{(k)} \rightarrow \theta^*$  as  $k \rightarrow \infty$ . Here those results are completed and expanded.

The iterate  $\theta^{(k+1)}$  is uniquely determined if and only if  $F'(\theta^{(k)})$  is nonsingular. It is therefore of interest to know when  $F'(\theta)$  can be singular. If we associate with the vector  $\theta = (\theta_0, \theta_1, \dots, \theta_n)^T$  the polynomial  $\Theta(z) = \theta_0 + \theta_1 z + \dots + \theta_n z^n$ , as before, then we can answer this question in terms of the zeros of  $\Theta(z)$ .

**THEOREM 5.1.** *The Jacobian  $F'(\theta)$  is singular if and only if there exists a nonzero, complex number  $\alpha$  that satisfies  $\Theta(\alpha) = \Theta(1/\alpha) = 0$ .*

*Proof.* The Jacobian  $F'(\theta)$  is singular if and only if there exists a nontrivial, real vector  $x = (x_0, \dots, x_n)^T$  that satisfies  $F'(\theta)x = 0$ . Now  $F'(\theta) = G(\theta) + H(\theta)$ , where

$$G(\theta) = \begin{bmatrix} \theta_0 & \theta_1 & \dots & \theta_n \\ & \theta_0 & \dots & \theta_{n-1} \\ & & \ddots & \vdots \\ & & & \theta_0 \end{bmatrix} \quad \text{and} \quad H(\theta) = \begin{bmatrix} \theta_0 & \theta_1 & \dots & \theta_n \\ \theta_1 & & & \\ \vdots & & & \\ \theta_n & & & \end{bmatrix}.$$

With the vector  $x$ , associate the polynomial  $X(z)$  defined by

$$X(z) = x_0 + x_1 z + \dots + x_n z^n.$$

Then  $F'(\theta)x = 0$  is equivalent to

$$(5.2) \quad \Theta(z)X\left(\frac{1}{z}\right) + X(z)\Theta\left(\frac{1}{z}\right) = 0.$$

This is easily verified by expanding the left-hand side and equating like powers of  $z$ —the coefficient of  $(z^k + 1/z^k)$  is precisely the  $k$ th component of  $F'(\theta)x$ .

Let  $\alpha_1, \dots, \alpha_n$  and  $\beta_1, \dots, \beta_n$  be the zeros of  $\Theta(z)$  and  $X(z)$ . Then (5.2) is equivalent to

$$(5.3) \quad \frac{-1}{\alpha_1 \dots \alpha_n} \frac{(z - \alpha_1) \dots (z - \alpha_n)}{(z - 1/\alpha_1) \dots (z - 1/\alpha_n)} = \frac{1}{\beta_1 \dots \beta_n} \frac{(z - \beta_1) \dots (z - \beta_n)}{(z - 1/\beta_1) \dots (z - 1/\beta_n)}.$$

Thus there exists a nontrivial right null vector for  $F'(\theta)$  if and only if there exist  $\beta_1, \dots, \beta_n$  producing an identity above. This can happen if and only if there is some cancellation between the numerator and denominator of the left-hand side of (5.3).

To see this, suppose that  $\alpha$  and  $1/\alpha$  are roots of  $\Theta(z) = 0$ . We can assume, without loss of generality, that  $\alpha_1 = \alpha$  and  $\alpha_2 = 1/\alpha$ . Let  $\beta_j$  be equal to  $\alpha_j$ ,  $j = 3, \dots, n$ . Then (5.3) becomes

$$\frac{-1}{\alpha_1 \alpha_2} \frac{(z - \alpha_1)(z - \alpha_2)}{(z - 1/\alpha_1)(z - 1/\alpha_2)} = \frac{1}{\beta_1 \beta_2} \frac{(z - \beta_1)(z - \beta_2)}{(z - 1/\beta_1)(z - 1/\beta_2)}.$$

But  $\alpha_1 = 1/\alpha_2$ ; so the left-hand side above is equal to  $-1$ , and this equation is satisfied by  $\beta_1 = 1$  and  $\beta_2 = -1$ . If no such reciprocal roots ( $\alpha$  and  $1/\alpha$ ) exist, then there cannot be any  $\beta_1, \dots, \beta_n$  to satisfy (5.3) (the  $\beta_1, \dots, \beta_n$  would have to be a reordering of  $\alpha_1, \dots, \alpha_n$ , and this would lead to the contradiction  $-1 = 1$ ).  $\square$

There are some important consequences of this theorem. If  $\theta$  is an invertible solution, then all of the roots of  $\Theta(z) = 0$  are strictly greater than one in modulus. Consequently no two roots can be reciprocals, and  $F'(\theta)$  is guaranteed to be nonsingular. If, on the other hand,  $\theta$  is a real solution such that  $\Theta(z)$  has a root of unit modulus, then the reciprocal must also be a root (because  $\bar{z} = 1/z$  on  $|z| = 1$ ), and it follows that  $F'(\theta)$  is necessarily singular.

If real solutions of (0.1) exist, then there exists just one,  $\theta^*$ , with the property that all roots of the corresponding polynomial  $\Theta^*(z)$  are greater than or equal to one in magnitude. The next theorem states that Newton's method, if properly started, is guaranteed to converge to this solution (even though  $F'(\theta^*)$  may be singular).

**THEOREM 5.2.** *If real solutions exist and if the initial vector  $\theta^{(0)}$  is such that the corresponding polynomial  $\Theta^{(0)}(z)$  is greater than zero in  $|z| \leq 1$ , then the sequence  $\{\theta^{(k)}\}$  produced by Newton's algorithm is well defined and converges to the real solution  $\theta^*$  indicated above.*

*Proof.* First we show that the sequence  $\{\theta^{(k)}\}$  is well defined, that is,  $F'(\theta^{(k)})$  is nonsingular,  $k = 0, 1, \dots$ . It is sufficient to show that if  $\Theta^{(k)}(z)$  has no roots in  $|z| \leq 1$ , then neither does  $\Theta^{(k+1)}(z)$ . If  $\Theta^{(k)}(z)$  has no roots in  $|z| \leq 1$ , then  $F'(\theta^{(k)})$  is nonsingular, and  $\theta^{(k+1)}$  is uniquely determined. It can be shown that the Newton equation

$$F'(\theta^{(k)})\theta^{(k+1)} = F'(\theta^{(k)})\theta^{(k)} - F(\theta^{(k)})$$

is equivalent to

$$(5.4) \quad \Theta^{(k)}(z)\Theta^{(k+1)}\left(\frac{1}{z}\right) + \Theta^{(k+1)}(z)\Theta^{(k)}\left(\frac{1}{z}\right) = \Theta^{(k)}(z)\Theta^{(k)}\left(\frac{1}{z}\right) + \Theta^*(z)\Theta^*\left(\frac{1}{z}\right).$$

Here we have used the facts that  $F'(\theta)\theta = 2(F(\theta) + c)$  and  $C(z) = \Theta^*(z)\Theta^*(1/z)$ .

Rewrite this equation in the form

$$\Theta^{(k+1)}(z)\Theta^{(k+1)}\left(\frac{1}{z}\right) = [\Theta^{(k+1)}(z) - \Theta^{(k)}(z)]\left[\Theta^{(k+1)}\left(\frac{1}{z}\right) - \Theta^{(k)}\left(\frac{1}{z}\right)\right] + \Theta^*(z)\Theta^*\left(\frac{1}{z}\right).$$

Now on  $|z| = 1$ , we have  $\bar{z} = 1/z$ ; so this implies that

$$(5.5) \quad |\Theta^{(k+1)}(z)|^2 = |\Theta^{(k+1)}(z) - \Theta^{(k)}(z)|^2 + |\Theta^*(z)|^2 \quad \text{on } |z| = 1.$$

It follows that  $\Theta^{(k+1)}(z)$  does not vanish on  $|z| = 1$  (that would imply the vanishing of  $\Theta^{(k)}(z)$  there) and

$$|\Theta^{(k+1)}(z) - \Theta^{(k)}(z)| \leq |\Theta^{(k+1)}(z)| \quad \text{on } |z| = 1.$$

Thus the meromorphic function  $f(z)$  defined by  $f(z) = \Theta^{(k)}(z)/\Theta^{(k+1)}(z)$  has neither zeros nor poles on  $|z| = 1$  and satisfies

$$|f(z) - 1| \leq 1 \quad \text{on } |z| = 1.$$

Now the argument principle implies that the number of zeros of  $f$  in  $|z| \leq 1$  minus the number of poles there is equal to zero (the number of times that the trace  $w = f(z)$ ,  $|z| = 1$ , encircles the origin). So the number of zeros of  $\Theta^{(k+1)}(z)$  in  $|z| \leq 1$  is equal to the number of zeros of  $\Theta^{(k)}(z)$  there, namely zero. Thus  $F'(\theta^{(k+1)})$  is nonsingular, and it follows by induction that the sequence  $\{\theta^{(k)}\}$  is well defined.

The remainder of the proof can now proceed exactly as in Wilson (1969). It can be shown that  $\Theta^{(k+1)}(z)$  and  $\Theta^{(k)}(z)$  are of the same constant sign on  $[-1, 1]$ , which will be positive since  $\Theta^{(0)}(z)$  is chosen to be positive there, and

$$0 < \frac{1}{2}\Theta^{(k)}(z) < \Theta^{(k+1)}(z) \leq \Theta^{(k)}(z) \quad \text{for } -1 \leq z \leq 1.$$

It follows that  $\{\Theta^{(k)}(z)\}$  is nonincreasing and bounded below on  $-1 \leq z \leq 1$ , and therefore it converges to a limit polynomial that must be identical to  $\Theta^*(z)$  (this follows from taking limits in (5.4)). Thus  $\theta^{(k)} \rightarrow \theta^*$  as  $k \rightarrow \infty$ .  $\square$

The proof above follows closely that of Wilson (1969). It differs from Wilson’s proof on two points. First, it addresses the question of the nonsingularity of the Jacobian—Wilson tacitly assumes that  $\theta^{(k+1)}$  is uniquely determined from  $\theta^{(k)}$ , i.e., that the Jacobians are all nonsingular. And second, it extends Wilson’s basic argument to include the case of a Jacobian that is singular at the solution.

If the real solution  $\theta^*$  is invertible, then the convergence is (asymptotically) quadratic. Otherwise the Jacobian  $F'(\theta^*)$  is singular, and thus the convergence is only linear. This has some nice implications in terms of the development of a code for these problems. It says that if you use Newton’s method, properly started, then you will observe one of three things: (1) quadratic convergence, in which case  $\theta^*$  is invertible, (2) linear convergence, in which case  $\theta^*$  is noninvertible, or (3) no convergence, in which case no real solutions exist.

The implications of this theorem in terms of the time series analysis are also interesting. It says that while the underlying moving average process may be nearly noninvertible, in which case the process with the estimated covariances can actually be noninvertible, the Newton algorithm will still converge (albeit linearly) to the borderline invertible solution  $\theta^*$ , and this can still be used as a starting value in the nonlinear least squares estimation of the parameters.

A starting vector that satisfies the hypothesis of the last theorem is given by

$$(5.6) \quad \theta^{(0)} = \left[ \frac{c_0}{c_0^2 + \dots + c_n^2} \right]^{1/2} \cdot (c_0, \dots, c_n)^T.$$

Numerical experiments have shown this to be superior to the vector proposed by Wilson (1969); it is obtained in the following way. Time series analysts suggest that a “reasonable” starting vector is given by

$$(5.7) \quad \theta^{(-1)} = (c_0^{1/2}, 0, \dots, 0)^T,$$

which is the solution of the modified system (0.1) with right-hand side  $(c_0, 0, \dots, 0)^T$ —the more  $c_0$  dominates the remaining coefficients, the better this approximation becomes. Now  $F'(\theta^{(-1)})$  is diagonal matrix, and consequently one iteration of Newton’s method can be calculated explicitly to produce

$$\tilde{\theta}^{(0)} = (c_0, \dots, c_n)^T / c_0^{1/2}.$$

The initial vector (5.6) is then obtained by normalizing  $\tilde{\theta}^{(0)}$  so as to satisfy the first equation  $\langle \theta^{(0)}, \theta^{(0)} \rangle = c_0$  of (1.1). Since  $\theta^{(0)}$  is obtained from  $\theta^{(-1)}$  (which satisfies the initial vector condition of Theorem 5.2) by a Newton step followed by a normalization, we are assured that  $\theta^{(0)}$  too will be such that  $\Theta^{(0)}(z)$  has no roots in  $|z| \leq 1$ .

**6. Local existence and uniqueness of solutions.** In time series models, it is common for  $c_0$  to be quite a bit larger than  $|c_i|$  for any  $i \geq 1$ . This type of condition is exactly what is needed to show local existence and uniqueness of solutions to  $F(\theta) = 0$  by the



simplified Newton's method (Ortega and Rheinboldt (1970, p. 421)). Using this method, the vector sequence  $\{\theta^{(p)}\}$  is generated by

$$(6.1) \quad \theta^{(p+1)} = \theta^{(p)} - F'(\theta^{(0)})^{-1}F(\theta^{(p)}), \quad p = 0, 1, 2, \dots$$

This method is especially attractive for at least two reasons. First, there is an initial vector  $\theta^{(0)}$ , which is the vector in (5.7), that is usually close to a desirable solution and which makes  $F'(\theta^{(0)})$  a diagonal matrix and thus easy to invert. Second, conditions for the existence and uniqueness of a real solution in a neighborhood about  $\theta^{(0)}$  can be given just in terms of the scalars  $c_i$  of the equation  $F(\theta) = 0$  (see (6.3) and (6.5) below). In particular, condition (6.5) is only slightly stiffer than the sufficient condition for the existence of an invertible solution of  $F(\theta) = 0$  that was presented earlier in Proposition 3.2. We also note that the conditions given below are sufficient for the convergence of the Newton iterates in (5.1) as well as for the convergence of (6.1). Therefore the point of attraction,  $\theta^*$ , of the sequence generated from (6.1) is the invertible solution of  $F(\theta) = 0$ .

**THEOREM 6.1.** *For the system  $F(\theta) = 0$  as given in (0.1), let  $\theta^{(0)}$  be the vector in (5.7), and define the neighborhood*

$$(6.2) \quad D_0 \equiv \{\theta \in R^{n+1}: \|\theta - \theta^{(0)}\|_\infty \leq c_0^{1/2}/(2n+2)\}.$$

*If the scalars  $c_i$  satisfy the single condition*

$$(6.3) \quad \max_{i \geq 1} |c_i| \leq \frac{c_0}{4n+4},$$

*then there exists a unique real solution  $\theta^*$  of (0.1) in  $D_0$  to which the simplified Newton iterates (6.1) converge.*

*Proof.* The choice of the starting vector (5.7) is important because it allows the Newton-Kantorovich conditions to be easily applied. For, observe that since no off-diagonal entry in  $F'(\theta)$  contains  $\theta_0$ , then

$$F'(\theta^{(0)}) = G(\theta^{(0)}) + H(\theta^{(0)}) = c_0^{1/2} \cdot \text{diag}\{2, 1, 1, \dots, 1\}$$

is a diagonal matrix from which we can easily derive the matrix norm estimate  $\|F'(\theta^{(0)})^{-1}\|_\infty = 1/c_0^{1/2} \equiv \beta$ . Moreover, the  $j$ th entry of the first row of the matrix  $F'(\theta) - F'(\phi)$ , for any  $\theta, \phi$  in  $R^{n+1}$ , is  $2(\theta_j - \phi_j)$ . This row has the largest (in magnitude) row sum, and thus

$$\|F'(\theta) - F'(\phi)\|_\infty = 2 \sum_{j=1}^{n+1} |\theta_j - \phi_j| \leq 2(n+1)\|\theta - \phi\|_\infty.$$

In this inequality, define  $\gamma \equiv 2(n+1)$ . Now let us consider

$$F'(\theta^{(0)})^{-1}F(\theta^{(0)}) = -(0, c_1, c_2, \dots, c_n)^T/c_0^{1/2}.$$

Then

$$\|F'(\theta^{(0)})^{-1}F(\theta^{(0)})\|_\infty = \max_{i \geq 1} |c_i|/c_0^{1/2} \equiv \eta.$$

The demand that  $\alpha \equiv \beta\gamma\eta \leq 0.5$  is met by condition (6.3):

$$\beta\gamma\eta = 2(n+1) \max_{i \geq 1} |c_i|/c_0 \leq 2(n+1)c_0/(4n+4)c_0 = 0.5.$$

This requirement that  $\alpha \leq 0.5$  is the basic Newton-Kantorovich condition for the

convergence of the iterates of (6.1). Furthermore, setting

$$t^* \equiv (\beta\gamma)^{-1}[1 - (1 - 2\alpha)^{1/2}],$$

we get  $t^* \equiv (\beta\gamma)^{-1} = c_0^{1/2}/2(n+1)$ . From (6.2), we then see that the closed sphere  $\bar{S}(\theta^{(0)}, t^*)$  with center  $\theta^{(0)}$  and radius  $t^*$  in  $R^{n+1}$  is a subset of  $D_0$ . Also if

$$t^{**} \equiv (\beta\gamma)^{-1}[1 + (1 - 2\alpha)^{1/2}],$$

then  $t^{**} \equiv (\beta\gamma)^{-1} = c_0^{1/2}/2(n+1)$  and so the closed sphere  $\bar{S}(\theta^{(0)}, t^{**})$  with center  $\theta^{(0)}$  and radius  $t^{**}$  in  $R^{n+1}$  has intersection  $\bar{S}(\theta^{(0)}, t^{**}) \cap D_0 = D_0$ . Hence an appeal to Ortega and Rheinboldt (1970, Thm. 12.6.1) tells us that the iterates of (6.1) converge to a solution  $\theta^*$  of  $F(\theta) = 0$  which is unique in  $D_0$ .  $\square$

By changing the norm on  $R^{n+1}$ , we arrive at another version of the last theorem through arguments similar to those presented above.

**THEOREM 6.2.** *The conclusion of Theorem 6.1 remains valid if statements (6.2) and (6.3) are replaced by the statements*

$$(6.4) \quad D_0 \equiv \left\{ \theta \in R^{n+1} : \|\theta - \theta^{(0)}\|_1 \leq \frac{c_0^{1/2}}{2} \right\},$$

$$(6.5) \quad \sum_{i=1}^n |c_i| \leq \frac{c_0}{4},$$

respectively.

**7. Numerical results.** In this section, we report some simple examples of numerical experiments that were conducted in order to appraise the performance and relative merits of the Newton and direct factorization methods. The four examples are given below. Here  $c = (c_0, \dots, c_n)$ , the covariances;  $\theta = (\theta_0, \dots, \theta_n)$ , the moving average parameters; and  $\alpha = (\alpha_1, \dots, \alpha_n)$ , the roots of  $\Theta(z)$ . For simplicity, these examples are limited to the case  $n = 3$ ; they are, however, representative of larger scale problems that were tested.

*Example 7.1.*

$$c = (8004, 2491, 622, 85), \quad \theta = (85, 27, 7, 1), \quad \alpha = (-1 - 4i, -1 + 4i, 5).$$

This example is taken from Wilson (1969). It satisfies the sufficient condition of Proposition 3.2 for the existence of an invertible solution, and in addition, all of the roots are simple.

*Example 7.2.*

$$c = (450, -311, 100, -12), \quad \theta = (12, -16, 7, -1), \quad \alpha = (2, 2, 3).$$

In this example again, an invertible solution exists, but the covariance generating function now has multiple roots.

*Example 7.3.*

$$c = (195.9462, -139.3835, 47.4706, -6.06), \quad \theta = (6.06, -11.05, 6.01, -1.0), \\ \alpha = (1.01, 2.0, 3.0).$$

This example is nearly borderline invertible:  $\Theta(z)$  has the root  $\alpha_1 = 1.01$ , which is close to unit modulus.

*Example 7.4.*

$$c = (10, -6, 5, -2), \quad \theta = (2, -1, 2, -1), \quad \alpha = (i, -i, 2).$$

This example is borderline invertible: it has the roots  $\alpha_1 = i$  and  $\alpha_2 = -i$  on the unit circle.

These examples were run on a CDC 6600 in single precision, the equivalent of about 14 decimal digits. The Newton method was implemented in a straightforward manner using a standard linear system routine (from LINPACK) to solve for the Newton correction at each step. The iteration was continued until full attainable accuracy was reached. The direct factorization method used a standard root finder (from IMSL) to compute all of the zeros of the  $2n$ th degree polynomial  $z^n C(z)$ . One is able to improve the efficiency of this by using the fact that each computed root,  $\alpha$ , actually determines two roots,  $\alpha$  and  $1/\alpha$ , and then  $\bar{\alpha}$  can be given as a (sometimes very accurate) initial guess for the next root. The moving average parameters,  $\theta_0, \dots, \theta_m$ , were constructed from the roots,  $\alpha_1, \dots, \alpha_m$ , in terms of elementary symmetric functions, which were computed recursively from sums of powers of the roots by using the Newton identities (see, for example, Isaacson and Keller (1966, Chapt. 7, § 2.1).

The results of these experiments are summarized in Tables 1 and 2. For problem 7.1, Newton's method gives very rapid convergence; it is helped by the fact that  $c_0$  dominates the covariances, which makes  $\theta^{(0)}$  a very good initial guess—it already has almost 3 significant digits. The factorization routine also performs well, giving essentially full machine precision.

TABLE 1  
*Newton's method for problems 7.1-7.4.*

Problem	$n$	Relative error	$n$	Relative error
7.1	0	5.7 (-3)	2	3.8 (-10)
	1	2.7 (-5)	3	0.0
7.2	0	3.2 (-1)	4	1.6 (-4)
	1	1.6 (-1)	5	8.8 (-8)
	2	4.7 (-2)	6	1.1 (-14)
7.3	3	6.7 (-3)		
	0	4.6 (-1)	7	3.6 (-3)
	1	2.9 (-1)	8	1.0 (-3)
	2	1.6 (-1)	9	1.4 (-4)
	3	8.6 (-2)	10	3.6 (-6)
	4	4.4 (-2)	11	2.4 (-9)
7.4	5	2.1 (-2)	12	6.8 (-12)
	6	9.3 (-3)		
	0	3.8 (-1)	12	8.9 (-5)
	1	1.9 (-1)	—	—
	2	9.1 (-2)	18	1.4 (-6)
	3	4.5 (-2)	19	6.9 (-7)
	4	2.3 (-2)	20	3.5 (-7)
	—	—	21	1.7 (-7)
8	1.4 (-3)	22	7.6 (-8)	
—	—	23	3.8 (-8)	

TABLE 2  
*Factorization method for problems 7.1-7.4.*

Problem	Relative error in roots	Relative error in solution
7.1	2.8 (-14)	2.7 (-14)
7.2	3.2 (-7)	3.9 (-8)
7.3	5.7 (-12)	9.8 (-12)
7.4	3.5 (-8)	6.8 (-8)

On Example 7.2, Newton does fine again—although the initial guess is not quite as good as in the previous example—delivering quadratic convergence and a solution that has full machine accuracy. The direct factorization method, however, delivers only about half machine precision. This is due to the fact that  $C(z)$  has multiple roots, which are ill-conditioned.

Example 7.3 is the nearly borderline invertible example. Both methods suffer a bit, losing about two decimal digits. Newton's method appears to converge linearly for a while (because of the near singularity of the Jacobian at the solution) then homes in quadratically once it gets close. The factorization method has trouble because  $\alpha_1 = 1.01$  and  $1/\alpha_1 \cong 0.9901$ . These are both roots of  $C(z)$ ; they are poorly separated, and this hurts their conditioning.

The last example is borderline invertible; so we know that the Jacobian is singular at the solution and that Newton's method will converge linearly. The solution is badly conditioned, and we can only get about half machine accuracy. Factorization of the covariance function suffers a similar fate, because although the roots of  $\Theta(z)$  are simple, the roots of  $C(z)$  that lie on the unit circle are multiple (as they must be for real solutions to exist) and therefore ill-conditioned.

We mention that numerical experiments were also conducted with the simplified Newton's method. This method can easily be implemented, without any root finders or linear system solvers, because the Jacobian  $F'(\theta^{(0)})$  is diagonal. In the cases where the hypotheses of Theorems 6.1 and 6.2 were satisfied, the method provided linear convergence, as predicted. It even was observed to converge in cases where the hypotheses were not satisfied as long as an invertible solution existed. However the rate of convergence became unbearably slow as a root of  $\Theta(z)$  approached the unit circle, and the iteration failed to converge in any borderline invertible case.

**8. Conclusions.** The Newton method and direct factorization method can be compared with respect to attainable accuracy, efficiency and information provided. Theory and experiment indicate that the attainable accuracy of the Newton method is always at least as good as that of the factorization method. In those places where Newton has low accuracy (where the Jacobian is singular or nearly singular at the solution), so does factorization; but the factorization method can lose accuracy when Newton does not (when the roots of  $\Theta(z)$  are all much bigger than 1 in modulus but poorly separated). Also, for large  $n$ , the factorization method can lose accuracy due to the accumulation of roundoff error in computing the moving average parameters from the roots.

Both programs have about the same complexity. Newton's method requires that a dense  $(n+1)$  by  $(n+1)$  linear system be solved at each iteration; this requires  $O(n^3)$  multiplications. Actually, the special form of the Jacobian matrix (Toeplitz plus Hankel) can be exploited to solve these systems more efficiently by using recursive techniques (see Marple (1982) or Merchant and Parks (1982)). The factorization method gets just as expensive in constructing  $\{\theta_0, \dots, \theta_n\}$  from  $\{\alpha_1, \dots, \alpha_n\}$  via symmetric functions. These require the computation of sums of powers of the roots,  $\alpha_1 + \dots + \alpha_n$ ,  $\alpha_1^2 + \dots + \alpha_n^2$ ,  $\dots$ ,  $\alpha_1^n + \dots + \alpha_n^n$ , and this again leads to  $O(n^3)$  arithmetic.

Both methods provide roughly the same information (in different ways). In Newton's method, by way of the rate of convergence, one can tell whether the solution is invertible, borderline invertible or nearly borderline invertible, and from this infer something about the accuracy of the computed solution. In the factorization method, one has all of the roots of the covariance function in hand and can easily make the same appraisals.

The methods are essentially equivalent. Both require little programming effort, and both are guaranteed to converge to the required solution. Newton's method would have to be given a slight edge based upon the fact that the solutions computed using the factorization method can lose accuracy in situations where Newton's method does not.

The simplified Newton method is easy to implement, but it only gives linear convergence and is restricted in application to the case where  $c_0$  dominates  $c_1, \dots, c_n$  sufficiently strongly. The analysis of this method is a clean and appealing application of the Newton-Kantorovich theorem and gives rise to conditions on the data  $c_0, \dots, c_n$  that are easy to check. It does not possess the "robustness" of the Newton and direct factorization methods, and it could only be considered a competitor in certain special circumstances (in the case where one does not have access to a general root finder or linear system solver, as in the case of desk top calculations, say).

## REFERENCES

- D. H. ANDERSON (1982), *Structural properties of compartmental models*, Math. Biosci., 58, pp. 61-81.
- O. D. ANDERSON (1974), *An inequality with a time series application*, J. Econometrics, 2, pp. 189-193.
- (1975), *Bounding sums for the autocorrelations of moving average processes*, Biometrika, 62, pp. 706-707.
- (1975a), *Time Series Analysis and Forecasting: A Box-Jenkins Approach*, Butterworths, London.
- G. E. P. BOX AND G. M. JENKINS (1970), *Time Series Analysis, Forecasting and Control*, Holden-Day, San Francisco.
- C. COBELLI, A. LEPSCHY AND G. ROMANIN-JACUR (1979), *Identifiability of compartmental systems and related structural properties*, Math. Biosci., 44, pp. 1-18.
- N. DAVIES, M. B. PATE AND M. G. FROST (1974), *Maximum autocorrelations for moving average processes*, Biometrika, 61, pp. 199-200.
- J. DELFORGE (1977), *The problem of structural identifiability of a linear compartmental system: solved or not?*, Math. Biosci., 36, pp. 119-125.
- P. HENRICI (1974), *Applied and Computational Complex Analysis*, Vol. 1, John Wiley, New York.
- E. ISAACSON AND H. B. KELLER (1966), *Analysis of Numerical Methods*, John Wiley, New York.
- P. LANCASTER (1969), *Theory of Matrices*, Academic, New York.
- S. L. MARPLE (1982), *Fast algorithms for linear prediction and system identification filters with linear phase*. IEEE Trans. Acoust. Speech Signal Process ASSP-30, pp. 942-953.
- G. A. MERCHANT AND T. W. PARKS (1982), *Efficient solution of a Toeplitz-plus-Hankel coefficient matrix system of equations*, Ibid., ASSP-30, pp. 40-44.
- J. J. H. MILLER (1971), *On the location of zeros of certain classes of polynomials with applications in numerical analysis*, J. Inst. Math. Appl., 8, pp. 397-406.
- J. M. ORTEGA AND W. C. RHEINBOLDT (1970), *Iterative Solution of Nonlinear Equations in Several Variables*, Academic, New York.
- G. WILSON (1969), *Factorization of the covariance generating function of a pure moving average process*, SIAM J. Numer. Anal., 6, pp. 1-7.

## AN IMPLEMENTATION OF GAUSSIAN ELIMINATION WITH PARTIAL PIVOTING FOR SPARSE SYSTEMS\*

ALAN GEORGE† AND ESMOND NG†

**Abstract.** In this paper, we consider the problem of solving a sparse nonsingular system of linear equations. We show that the structures of the triangular matrices obtained in the  $LU$ -decomposition of a sparse nonsingular matrix  $A$  using Gaussian elimination with partial pivoting are contained in those of the Cholesky factors of  $A^T A$ , provided that the diagonal elements of  $A$  are nonzero. Based on this result, a method for solving sparse linear systems is then described. The main advantage of this method is that the numerical computation can be carried out using a static data structure. Numerical experiments comparing this method with other implementations of Gaussian elimination for solving sparse linear systems are presented and the results indicate that the method proposed in this paper is quite competitive with other approaches.

**Key words.** Gaussian elimination, sparse matrices, partial pivoting

**1. Introduction.** In this paper, we consider the direct solution of the system of linear equations

$$Ax = b,$$

where  $A$  is a given  $n \times n$  matrix,  $b$  is a given  $n$ -vector, and  $x$  is the  $n$ -vector to be computed. We assume that  $A$  is sparse, nonsymmetric and nonsingular. One of the most popular techniques for solving such a linear system involves computing an  $LU$ -decomposition of  $A$  using Gaussian elimination with partial pivoting. That is,  $A$  is decomposed into

$$A = P_1 L_1 P_2 L_2 \cdots P_{n-1} L_{n-1} U,$$

where  $P_k$  is an  $n \times n$  permutation matrix corresponding to the row interchanges in step  $k$ ,  $L_k$  is an  $n \times n$  unit lower triangular matrix whose  $k$ th column contains the multipliers, and  $U$  is an  $n \times n$  upper triangular matrix. Then the solution to the original linear system is obtained by solving the systems

$$P_1 L_1 P_2 L_2 \cdots P_{n-1} L_{n-1} y = b$$

and

$$Ux = y.$$

Given a sparse matrix  $A$ , it is usually true that *fill-in* will occur during the decomposition process; that is, nonzeros may be created in positions where there are zeros in  $A$ . Thus, space has to be allocated not only for the nonzeros in  $A$ , but also for the fill-in. Note that the structures of the triangular matrices  $L_k$  and  $U$  depend on both the *structure* of  $A$  and the *row interchanges* ( $P_k$ ). Also, the row interchanges depend on the *numerical values* of  $A$  (and of the subsequent reduced matrices). Thus, it appears that one cannot predict where fill-in will occur before the numerical

---

\* Received by the editors June 6, 1983, and in revised form December 15, 1983. 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 W-7405-eng-26 with the Union Carbide Corporation and by the U.S. Air Force Office of Scientific Research under contract AFOSR-ISSA-83-00060.

† Department of Computer Science, University of Waterloo, Waterloo, Ontario, Canada N2L 3G1.

decomposition begins, since the row interchanges are not known beforehand. Consequently, it is common practice in the implementation of sparse  $LU$ -decomposition with row interchanges to allocate space for any fill-in *during* the numerical computation phase. (That is, one uses a *dynamic* data structure.) Most computer packages that compute an  $LU$ -decomposition of a sparse matrix with row interchanges use a dynamic data structure. This usually results in substantial overhead in both storage requirements and execution time.

In this paper we show that the structures of the triangular matrices  $L_k$  and  $U$  are contained in the structures of the Cholesky factors of the symmetric positive definite matrix  $A^T A$ , as long as the diagonal elements of  $A$  are nonzero. Assume  $A^T A$  and its Cholesky factor are sparse. Since it is possible to determine the structure of the Cholesky factor of  $A^T A$  efficiently from the structure of  $A^T A$ , this gives us a scheme for implementing the  $LU$ -decomposition of  $A$  using Gaussian elimination with partial pivoting. The attractive feature of this scheme is that a dynamic data structure is *not* needed. By analyzing the structure of  $A^T A$ , we determine the structure of the Cholesky factor of  $A^T A$  and set up a storage scheme. Then we simply use that *static* storage scheme in the numerical decomposition of  $A$  using Gaussian elimination with partial pivoting.

The proposed scheme assumes that  $A^T A$  and its Cholesky factor are sparse if  $A$  is sparse, but there are examples in which  $A^T A$  and its Cholesky factor are dense even though  $A$  is sparse. However experience shows that this usually occurs when a relatively small number of the rows of  $A$  are dense. We will consider the problem of dense rows briefly and propose a technique to handle these dense rows so that we only have to compute the  $LU$ -decomposition of a sparse submatrix of  $A$ .

An outline of the paper is as follows. In § 2 we derive the main results which show that the structures of the triangular matrices  $L_k$  and  $U$  are contained in those of the Cholesky factors of  $A^T A$ . The effect of permuting the columns of  $A$  is examined in § 3. In § 4 the proposed method is described and in § 5 it is compared with other implementations of Gaussian elimination for solving sparse linear systems. Numerical experiments are also provided to compare the performance of the various implementations in § 5. We consider the effect of dense rows and propose a technique to handle them in § 6. Finally, some concluding remarks are provided in § 7.

The structure of the Cholesky factor of  $A^T A$  is also used in the solution of sparse least squares problems  $\min_x \|Ax - b\|_2$  [1] and sparse underdetermined systems of linear equations  $A^T x = b$  [13], where  $A$  is  $m \times n$ , with  $m \geq n$ .

## 2. Preliminary results.

LEMMA 2.1 (Duff [1]) *Let  $A$  be a sparse nonsingular matrix. Then there exists a permutation matrix  $Q$  such that the diagonal elements of  $QA$  are all nonzero.*

The matrix  $QA$  is then said to have a *zero-free diagonal*. The problem of finding such a permutation matrix is a well-known one and it is sometimes called the *assignment problem*. See [1] for a description of this problem and [4] for an efficient algorithm for finding  $Q$ .

Since Lemma 2.1 is true for every nonsingular matrix  $A$ , we may therefore assume that the rows of the given matrix  $A$  have already been permuted. That is, in the remainder of this section, we assume that  $A$  has a zero-free diagonal.

The following notation will be used throughout the discussion in this section. Let  $A$  be a sparse  $n \times n$  matrix. The  $(i, j)$ -element of  $A$  is denoted by  $A_{ij}$ . The set of subscripts of the nonzeros of  $A$  is denoted by  $\text{Nonz}(A)$ . That is,

$$\text{Nonz}(A) = \{(i, j) | A_{ij} \neq 0\}.$$

Furthermore, we assume that exact *structural* cancellation does not occur. Hence, for any  $n$  by  $n$  matrices  $A$  and  $B$ ,  $\text{Nonz}(A \pm B) = \text{Nonz}(A) \cup \text{Nonz}(B)$ .

The following result, which we state without proof, is an immediate consequence of the fact that  $A$  has a zero-free diagonal. It says that  $\text{Nonz}(A)$  is contained in  $\text{Nonz}(A^T A)$ .

LEMMA 2.2. *Assume  $A$  has a zero-free diagonal and let  $B = A^T A$ . If  $A_{ij} \neq 0$ , then  $B_{ij} \neq 0$ . That is,  $\text{Nonz}(A) \subseteq \text{Nonz}(A^T A)$ .*

Now consider applying the first step of Gaussian elimination to  $A$  with partial pivoting,

$$P_1^T A = \begin{pmatrix} \alpha & u^T \\ v & E \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ v/\alpha & I \end{pmatrix} \begin{pmatrix} \alpha & u^T \\ 0 & F \end{pmatrix}.$$

Here  $P_1$  is an  $n \times n$  permutation matrix chosen so that  $|\alpha| \cong \|v\|_\infty$  (and  $\alpha \neq 0$ ). For simplicity, we assume the elements of  $u$  and  $v$  are given by

$$u^T = (u_2, u_3, \dots, u_n) \quad \text{and} \quad v^T = (v_2, v_3, \dots, v_n).$$

We also assume that

$$E = \begin{pmatrix} E_{22} & E_{23} & \dots & E_{2n} \\ E_{32} & E_{33} & \dots & E_{3n} \\ \vdots & \vdots & \ddots & \vdots \\ E_{n2} & E_{n3} & \dots & E_{nn} \end{pmatrix} \quad \text{and} \quad F = \begin{pmatrix} F_{22} & F_{23} & \dots & F_{2n} \\ F_{32} & F_{33} & \dots & F_{3n} \\ \vdots & \vdots & \ddots & \vdots \\ F_{n2} & F_{n3} & \dots & F_{nn} \end{pmatrix}.$$

It is easy to see that

$$F = E - \frac{1}{\alpha} v u^T.$$

Thus, if structural cancellation does not occur,

$$\text{Nonz}(F) = \text{Nonz}(E) \cup \text{Nonz}(v u^T).$$

We now consider the structure of  $F$  more carefully. Assume  $P_1$  interchanges rows 1 and  $s$  of  $A$  ( $1 \leq s \leq n$ ). If  $s = 1$ , then  $P_1 = I$ , and  $E_{ii} = A_{ii} \neq 0$  for  $2 \leq i \leq n$ , since  $A$  has a zero-free diagonal. Hence  $F_{ii} \neq 0$  for  $2 \leq i \leq n$ . If  $s \neq 1$ , then first note that  $u_s = A_{ss} \neq 0$  and  $v_s = A_{11} \neq 0$ . Now  $E_{ii} = A_{ii} \neq 0$  for  $2 \leq i \leq n$  and  $i \neq s$ , but  $E_{ss} = A_{1s}$  may be zero. For  $2 \leq i \leq n$  and  $i \neq s$ , clearly

$$F_{ii} = E_{ii} - \frac{1}{\alpha} v_i u_i \neq 0.$$

For  $i = s$ ,

$$F_{ss} = E_{ss} - \frac{1}{\alpha} v_s u_s \neq 0,$$

since both  $u_s$  and  $v_s$  are nonzero. Thus we have proved that *all* the diagonal elements of  $F$  are nonzero.

LEMMA 2.3. *The  $(n - 1) \times (n - 1)$  matrix  $F$  has a zero-free diagonal.*

COROLLARY 2.4.  $\text{Nonz}(F) \subseteq \text{Nonz}(F^T F)$ .

Consider the  $n \times n$  symmetric positive definite matrix  $B = A^T A$ ,

$$\begin{aligned} B &= A^T A = A^T P_1 P_1^T A = (P_1^T A)^T (P_1^T A) \\ &= \begin{pmatrix} \alpha & v^T \\ u & E^T \end{pmatrix} \begin{pmatrix} \alpha & u^T \\ v & E \end{pmatrix} = \begin{pmatrix} \alpha^2 + v^T v & \alpha u^T + v^T E \\ \alpha u + E^T v & u u^T + E^T E \end{pmatrix} = \begin{pmatrix} \beta & w^T \\ w & G \end{pmatrix}, \end{aligned}$$



where  $\beta = \alpha^2 + v^T v$ ,  $w = \alpha u + E^T v$ , and  $G = uu^T + E^T E$ . Applying the first step of Cholesky decomposition to  $B$ , we obtain

$$B = \begin{pmatrix} \beta & w^T \\ w & G \end{pmatrix} = \begin{pmatrix} \beta^{1/2} & 0 \\ \frac{w}{\beta^{1/2}} & I \end{pmatrix} \begin{pmatrix} 1 & 0 \\ 0 & H \end{pmatrix} \begin{pmatrix} \beta^{1/2} & \frac{w^T}{\beta^{1/2}} \\ 0 & I \end{pmatrix},$$

where

$$H = G - \frac{1}{\beta} ww^T.$$

The following results show the relationship between the structures of  $F$ ,  $H$ ,  $u$ ,  $v$  and  $w$ .

**THEOREM 2.5.**  $\text{Nonz}(F^T F) \subseteq \text{Nonz}(H)$ .

*Proof.* Note that

$$H = G - \frac{1}{\beta} ww^T = E^T E + uu^T - \frac{1}{\beta} (\alpha^2 uu^T + \alpha E^T v u^T + \alpha u v^T E + E^T v v^T E),$$

and

$$F^T F = \left( E - \frac{1}{\alpha} v u^T \right)^T \left( E - \frac{1}{\alpha} v u^T \right) = E^T E - \frac{1}{\alpha} u v^T E - \frac{1}{\alpha} E^T v u^T + \frac{v^T v}{\alpha^2} u u^T.$$

Thus, assuming exact structural cancellation does not occur,

$$\begin{aligned} \text{Nonz}(H) &= \text{Nonz}(E^T E) \cup \text{Nonz}(uu^T) \cup \text{Nonz}(E^T v u^T) \cup \text{Nonz}(u v^T E) \\ &\quad \cup \text{Nonz}(E^T v v^T E), \end{aligned}$$

and

$$\begin{aligned} \text{Nonz}(F^T F) &= \text{Nonz}(E^T E) \cup \text{Nonz}(u v^T E) \cup \text{Nonz}(E^T v u^T) \cup \text{Nonz}(u u^T) \\ &\subseteq \text{Nonz}(H). \end{aligned} \quad \square$$

Using Corollary 2.4 and Theorem 2.5, we obtain the next result.

**THEOREM 2.6.**  $\text{Nonz}(F) \subseteq \text{Nonz}(H)$ .

**THEOREM 2.7.**

(1)  $\text{Nonz}(u) \subseteq \text{Nonz}(w)$ .

(2)  $\text{Nonz}(v) \subseteq \text{Nonz}(w)$ .

*Proof.* It is obvious that  $\text{Nonz}(u) \subseteq \text{Nonz}(w)$ , since  $w = \alpha u + E^T v$ .

Now consider the structure of  $v$ . First assume that  $s = 1$ . Then  $E_{ii} = A_{ii} \neq 0$  for  $2 \leq i \leq n$ . Note that

$$w_i = \alpha u_i + \sum_{\substack{k=2 \\ k \neq i}}^n E_{ki} v_k = \alpha u_i + E_{ii} v_i + \sum_{\substack{k=2 \\ k \neq i}}^n E_{ki} v_k \quad \text{for } 2 \leq i \leq n.$$

Thus,  $w_i \neq 0$  if  $v_i \neq 0$ .

On the other hand, suppose  $s \neq 1$ . For  $i \neq s$ ,  $E_{ii} = A_{ii} \neq 0$  and

$$w_i = \alpha u_i + E_{ii} v_i + \sum_{\substack{k=2 \\ k \neq i}}^n E_{ki} v_k.$$

Thus, if  $v_i \neq 0$ , then  $w_i \neq 0$ . For  $i = s$ ,  $E_{ss}$  may be zero, but  $v_s \neq 0$  and  $u_s \neq 0$ . Hence

$$w_s = \alpha u_s + \sum_{k=2}^n E_{ks} v_k \neq 0.$$

This shows that  $\text{Nonz}(v) \subseteq \text{Nonz}(w)$ .  $\square$

Theorems 2.6 and 2.7 show that, at least for the first step of the  $LU$ -decomposition of  $A$ , the structures of  $u, v$  and  $F$  are contained in those of  $w$  and  $H$ . These two results can be extended to cover the complete  $LU$ -decomposition of  $A$  using Gaussian elimination with partial pivoting, as the following discussion shows. First recall that  $F$  is  $(n-1) \times (n-1)$  and has a zero-free diagonal, and  $F^T F$  is  $(n-1) \times (n-1)$ , symmetric and positive definite. If we apply one step of Gaussian elimination with row interchanges to  $F$  and one step of Cholesky factorization to  $F^T F$ , we obtain the following:

$$\begin{aligned} \bar{P}^T F &= \begin{pmatrix} 1 & 0 \\ \bar{v} & I \\ \bar{\alpha} & \end{pmatrix} \begin{pmatrix} \bar{\alpha} & \bar{u}^T \\ 0 & \bar{F} \end{pmatrix}, \\ F^T F &= \begin{pmatrix} \bar{\beta}^{1/2} & 0 \\ \bar{w} & I \\ \bar{\beta}^{1/2} & \end{pmatrix} \begin{pmatrix} 1 & 0 \\ 0 & \bar{H} \end{pmatrix} \begin{pmatrix} \bar{\beta}^{1/2} & \bar{w}^T \\ 0 & I \end{pmatrix}. \end{aligned}$$

Applying Theorems 2.6 and 2.7,  $\text{Nonz}(\bar{u}) \subseteq \text{Nonz}(\bar{w})$ ,  $\text{Nonz}(\bar{v}) \subseteq \text{Nonz}(\bar{w})$  and  $\text{Nonz}(\bar{F}) \subseteq \text{Nonz}(\bar{H})$ . Now consider the first step of the Cholesky decomposition of the  $(n-1) \times (n-1)$  symmetric positive definite matrix  $H$ ,

$$H = \begin{pmatrix} \hat{\beta}^{1/2} & 0 \\ \hat{w} & I \\ \hat{\beta}^{1/2} & \end{pmatrix} \begin{pmatrix} 1 & 0 \\ 0 & \hat{H} \end{pmatrix} \begin{pmatrix} \hat{\beta}^{1/2} & \hat{w}^T \\ 0 & I \end{pmatrix}.$$

By Theorem 2.5,  $\text{Nonz}(F^T F) \subseteq \text{Nonz}(H)$ . It is therefore obvious that  $\text{Nonz}(\bar{w})$  and  $\text{Nonz}(\bar{H})$  must be contained in  $\text{Nonz}(\hat{w})$  and  $\text{Nonz}(\hat{H})$  respectively. Combining these observations,  $\text{Nonz}(\bar{u}) \subseteq \text{Nonz}(\hat{w})$ ,  $\text{Nonz}(\bar{v}) \subseteq \text{Nonz}(\hat{w})$ , and  $\text{Nonz}(\bar{F}) \subseteq \text{Nonz}(\hat{H})$ . By applying these arguments repeatedly, we obtain Theorem 2.8 which is an extension of Theorems 2.6 and 2.7.

Before stating Theorem 2.8, we first introduce more notation. Consider the two sequences of matrices:

$$\{F^{(0)}, F^{(1)}, F^{(2)}, \dots, F^{(n-1)}\}$$

and

$$\{H^{(0)}, H^{(1)}, H^{(2)}, \dots, H^{(n-1)}\},$$

where  $F^{(0)} = A$  and  $H^{(0)} = A^T A$ . For  $k = 1, 2, \dots, n-1$ ,  $F^{(k)}$  is obtained by applying one step of Gaussian elimination to  $F^{(k-1)}$  with partial pivoting:

$$\bar{P}_k^T F^{(k-1)} = \begin{pmatrix} \alpha_k & (u^{(k)})^T \\ v^{(k)} & E^{(k)} \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ v^{(k)} & I_{n-k} \\ \alpha_k & \end{pmatrix} \begin{pmatrix} \alpha_k & (u^{(k)})^T \\ 0 & F^{(k)} \end{pmatrix} = \bar{L}_k \begin{pmatrix} \alpha_k & (u^{(k)})^T \\ 0 & F^{(k)} \end{pmatrix},$$

where  $\bar{P}_k$  is a permutation matrix of order  $(n-k+1)$ ,  $I_{n-k}$  is the identity matrix of

order  $(n - k)$  and  $F^{(k)} = E^{(k)} - (1/\alpha_k)v^{(k)}(u^{(k)})^T$ . Similarly, for  $k = 1, 2, \dots, n - 1$ ,  $H^{(k)}$  decomposes to  $H^{(k-1)}$ :

$$H^{(k-1)} = \begin{pmatrix} \beta_k & (w^{(k)})^T \\ w^{(k)} & G^{(k)} \end{pmatrix} = \begin{pmatrix} \beta_k^{1/2} & 0 \\ w^{(k)} & I_{n-k} \end{pmatrix} \begin{pmatrix} 1 & 0 \\ 0 & H^{(k)} \end{pmatrix} \begin{pmatrix} \beta_k^{1/2} & (w^{(k)})^T \\ 0 & I_{n-k} \end{pmatrix}$$

$$= \hat{R}_k^T \begin{pmatrix} 1 & 0 \\ 0 & H^{(k)} \end{pmatrix} \hat{R}_k,$$

where  $H^{(k)} = G^{(k)} - (1/\beta_k)w^{(k)}(w^{(k)})^T$ . Clearly

$$A = P_1 L_1 P_2 L_2 \cdots P_{n-1} L_{n-1} U,$$

where

$$P_k = \begin{pmatrix} I_{k-1} & 0 \\ 0 & \bar{P}_k \end{pmatrix}, \quad L_k = \begin{pmatrix} I_{k-1} & 0 \\ 0 & \bar{L}_k \end{pmatrix} = \begin{pmatrix} I_{k-1} & 0 & 0 \\ 0 & 1 & 0 \\ 0 & \frac{v^{(k)}}{\alpha_k} & I_{n-k} \end{pmatrix}, \quad 1 \leq k \leq n - 1,$$

and

$$U = \begin{pmatrix} \alpha_1 & (u^{(1)})^T & & & \\ 0 & \alpha_2 & & (u^{(2)})^T & & \\ & 0 & \alpha_3 & & (u^{(3)})^T & \\ & & 0 & \cdot & & \cdot \\ & & & & & \cdot \\ & & & & & \cdot \end{pmatrix}.$$

Also,

$$B = R_1^T R_2^T \cdots R_{n-1}^T R_n^T R_n R_{n-1} \cdots R_2 R_1,$$

where

$$R_k = \begin{pmatrix} I_{k-1} & 0 \\ 0 & \hat{R}_k \end{pmatrix} = \begin{pmatrix} I_{k-1} & 0 & 0 \\ 0 & \beta_k^{1/2} & (w^{(k)})^T/\beta_k^{1/2} \\ 0 & 0 & I_{n-k} \end{pmatrix} \quad \text{for } 1 \leq k \leq n - 1,$$

and

$$R_n = \begin{pmatrix} I_{n-1} & 0 \\ 0 & (H^{(n-1)})^{1/2} \end{pmatrix}.$$

The proof of Theorem 2.8 is the same as those of Theorems 2.6 and 2.7, and hence is omitted.

**THEOREM 2.8.** For  $k = 1, 2, \dots, n - 1$ ,

- (1)  $F^{(k)}$  has a zero-free diagonal.
- (2)  $\text{Nonz}(u^{(k)}) \subseteq \text{Nonz}(w^{(k)})$ .
- (3)  $\text{Nonz}(v^{(k)}) \subseteq \text{Nonz}(w^{(k)})$ .
- (4)  $\text{Nonz}(F^{(k)}) \subseteq \text{Nonz}(H^{(k)})$ .

Because of the way in which  $H^{(k)}$ 's are generated, it is not hard to see that

$$\bigcup_{k=0}^{n-1} \text{Nonz}(H^{(k)}) = \bigcup_{k=1}^n \text{Nonz}(R_k + R_k^T).$$

Thus, Theorem 2.8 essentially says that the structures of the triangular matrices obtained in the  $LU$ -decomposition of  $A$  are contained in those of the Cholesky factors of  $A^T A$ . Consequently, if we allocate space for the nonzero structure of the Cholesky factor of  $A^T A$ , then that data structure will *always* have space to accommodate *any* fill-in that occurs during the  $LU$ -decomposition of  $A$ . These results are important since it allows the  $LU$ -decomposition of  $A$  using Gaussian elimination *with partial pivoting* to be computed in a *predictable* amount of space. One may regard this result as saying that the structures of the Cholesky factors of  $A^T A$  predicts the *worst possible* structures of the triangular matrices  $L_k$  and  $U$ , for *any* pivotal sequence  $\{P_1, P_2, \dots, P_{n-1}\}$ .

Suppose  $B$  is a sparse symmetric positive definite matrix and denote its Cholesky factor by  $R_B$ . It is well known that the structure of  $R_B$  can be *predicted* from that of the matrix  $B$ . Thus one can allocate space for the nonzeros of  $R_B$  *before* the numerical computation begins. Furthermore, there are algorithms that accomplish these tasks efficiently. See [15] for details.

Hence, by setting  $B = A^T A$  and finding a data structure for the Cholesky factors,  $R_B$  and  $R_B^T$ , of  $B$ , we know from Theorems 2.6, 2.7 and 2.8 that there will always be space in that (static) data structure to accommodate any fill-in created during the  $LU$ -decomposition of  $A$  using Gaussian elimination with partial pivoting. Note that the symmetric positive definite matrix  $A^T A$  and its Cholesky factor are assumed to be sparse. This may not be true in some cases and we will address this problem in a later section.

**3. Effect of permuting the columns of  $A$ .** Let  $P_c$  be an  $n \times n$  permutation matrix and consider the matrix  $AP_c$ . Assume for the moment that  $AP_c$  has a zero-free diagonal. Denote the  $LU$ -decomposition of  $AP_c$  by

$$AP_c = P_1 L_1 P_2 L_2 \cdots P_{n-1} L_{n-1} U.$$

The results in the previous section show that the nonzero structures of the triangular matrices are contained in those of the Cholesky factors of  $(AP_c)^T (AP_c) = P_c^T A^T A P_c = P_c^T B P_c$ , where  $B = A^T A$ . We assume that  $B$  is sparse. It is well known that, for a sparse symmetric and positive definite matrix  $B$ , the choice of the permutation matrix  $P_c$  can drastically affect the sparsity of the Cholesky factor of  $P_c^T B P_c$  [15]. Hence it is desirable to find a  $P_c$  so that  $P_c^T B P_c$  has a sparse Cholesky decomposition.

The problem of finding a  $P_c$  that yields minimal fill-in is an NP-complete problem [18]. However, there are efficient heuristic algorithms that produce a  $P_c$  so that  $P_c^T B P_c$  has a reasonably sparse Cholesky decomposition. Examples include the nested dissection algorithm and the minimum degree algorithm [15].

It should be noted that even though  $A$  has a zero-free diagonal, the matrix  $AP_c$  may not necessarily have one. We illustrate this by a small example. Consider the following  $4 \times 4$  matrix:

$$A = \begin{pmatrix} \times & \times & & \\ & \times & & \\ & & \times & \\ & & & \times \end{pmatrix}.$$

Then

$$A^T A = \begin{pmatrix} \times & \times & & \\ \times & \times & & \\ & & \times & \\ & & & \times \end{pmatrix},$$

and  $\text{Nonz}(A) \subseteq \text{Nonz}(A^T A)$ . Let

$$P_c = \begin{pmatrix} & & & 1 \\ & & 1 & \\ & 1 & & \\ 1 & & & \end{pmatrix}.$$

Now

$$AP_c = \begin{pmatrix} & \times & \times & \\ & \times & & \\ \times & & & \\ & & & \times \end{pmatrix},$$

and

$$P_c^T A^T A P_c = \begin{pmatrix} \times & & & \\ & \times & \times & \\ & \times & \times & \\ & & & \times \end{pmatrix}.$$

Note that  $AP_c$  does not have a zero-free diagonal and the nonzero structure of  $P_c^T A^T A P_c$  does not even contain that of  $AP_c$ .

One way to preserve the property that  $A$  has a zero-free diagonal is as follows. Instead of permuting the columns of  $A$  by  $P_c$ , we permute *both* the columns and rows of  $A$  symmetrically by  $P_c$ . That is, we would consider the matrix  $P_c^T A P_c$ . Note that

$$(P_c^T A P_c)^T (P_c^T A P_c) = P_c^T A^T A P_c.$$

Thus premultiplying  $AP_c$  by  $P_c^T$  does not affect the structure of  $P_c^T A^T A P_c$  at all. However,  $P_c^T A P_c$  now has a zero-free diagonal (as long as  $A$  has one). To illustrate this, consider the previous  $4 \times 4$  example again,

$$P_c^T A P_c = \begin{pmatrix} \times & & & \\ & \times & & \\ & \times & \times & \\ & & & \times \end{pmatrix},$$

and the structure of  $P_c^T A^T A P_c$  indeed contains that of  $P_c^T A P_c$ .

Alternatively, one can reorder the columns of  $A$  *first* and obtain  $AP_c$ . Then a row permutation  $Q$  is determined so that  $QAP_c$  has a zero-free diagonal. Since

$$(QAP_c)^T (QAP_c) = P_c^T A^T A P_c,$$

permuting the rows of  $AP_c$  does not affect the structure of the Cholesky factor of  $P_c^T A^T A P_c$ . Finally an  $LU$ -decomposition of  $QAP_c$  is computed using the approach we have described in the previous section. Notice that it is not necessary for  $A$  to have a zero-free diagonal in this case.

In our implementation, we have chosen the first approach because it facilitates the use of existing sparse matrix software.

**4. Proposed method.** The results and discussions in §§ 2 and 3 provide us with a scheme for solving a sparse system of linear equations

$$Ax = b.$$

In general, the given coefficient matrix  $A$  may not have a zero-free diagonal. Thus it is necessary to find a row permutation  $Q$  so that the diagonal elements of  $QA$  are nonzero. This can be achieved by using the algorithm described in [4].

We now summarize the solution scheme below:

- (1) Find a permutation matrix  $Q$  so that  $QA$  has a zero-free diagonal.
- (2) Determine the structure of  $B = (QA)^T(QA) = A^T A$ .
- (3) Find a symmetric permutation  $P_c$  so that  $P_c^T B P_c$  has a sparse Cholesky factor. Denote the Cholesky factorization by  $P_c^T B P_c = \hat{R}^T \hat{R}$ .
- (4) Determine the structure of the Cholesky factor  $\hat{R}$  of  $P_c^T B P_c$ , and set up a storage scheme that exploits the sparsity of  $\hat{R}$  and  $\hat{R}^T$ .
- (5) Input the numerical values of  $A$ , storing it as  $P_c^T Q A P_c$ .
- (6) Compute the  $LU$ -decomposition of  $P_c^T Q A P_c$  using Gaussian elimination with partial pivoting. Store the triangular factors in the storage structure for  $\hat{R}^T$  and  $\hat{R}$ .
- (7) Solve  $(P_c^T Q A P_c) P_c^T x = P_c^T Q b$  using the  $LU$ -decomposition.

A few remarks on the implementation are in order. First, we only work with the structures of  $A$  and  $A^T A$  in Steps (1)–(4). Second, efficient algorithms are available for performing Steps (1)–(4). In the experiments which we will describe in the next section, we use the code given in [4] to find the permutation  $Q$  in Step 1. We use the minimum degree algorithm from SPARSPAK to find the symmetric permutation  $P_c$  in Step (3) and also the symbolic factorization routine from SPARSPAK to carry out Step (4) [14]. Third, the approach we have employed assumes that  $A^T A$  and  $\hat{R}$  are sparse if  $A$  is sparse. However there are some instances in which  $A^T A$  and  $\hat{R}$  may be dense even though  $A$  is sparse. We will deal with this situation in § 6.

**5. Comparison with other methods for solving sparse linear systems.** There are several codes available for solving sparse systems of linear equations  $Ax = b$  using an  $LU$ -decomposition of  $A$ . The ones we have considered include SPARSPAK [14], MA28 from Harwell [2], NSPIV [17], and an implementation of the method proposed in § 4. In this section, we first consider the basic methodology used by each package and examine its advantages and disadvantages.

**SPARSPAK.** This package computes an  $LU$ -decomposition of  $A$  without partial pivoting. A symmetric row and column ordering is first chosen so that the Cholesky factor of the permuted  $A + A^T$  is (hopefully) sparse. Then a data structure is set up for the Cholesky factors. Finally it uses that static data structure to compute an  $LU$ -decomposition of  $A$  using Gaussian elimination without partial pivoting. Since there are no row interchanges, numerical stability may be a problem. Also, the Cholesky factor of the permuted  $A + A^T$  may be unnecessarily full if  $A$  is far from being symmetric. Furthermore, it requires the diagonal elements of  $A$  to be nonzero. Of course, this can be circumvented by finding an assignment before choosing the symmetric ordering. However, there is still a chance that some of the diagonal elements may become zero during the decomposition process because of exact numerical cancellation. This is illustrated in our experiments. The symmetric ordering we have used in the numerical experiments was a minimum degree ordering.

**MA28 from Harwell.** In this code, column and row permutations are chosen to maintain numerical stability and preserve sparsity simultaneously. That is, the permutations will depend on the numerical values of the nonzero elements of  $A$  and the pivotal sequence. Thus one cannot predict how much space is needed before numerical computation begins, and storage has to be allocated during the numerical computation phase. Experience shows that the overhead, both in terms of execution time and storage,

can be quite significant. A threshold pivoting technique is used in search for a pivot; that is, at the  $k$ th step of the decomposition process, an element in the reduced matrix, say  $A_{ki}$ , will be chosen as the pivot if it satisfies

$$|A_{ki}| \geq u \max_j |A_{kj}|,$$

where  $u$  is a user specified parameter satisfying  $0 \leq u \leq 1$  (see [2], [7] for details). Increasing the threshold parameter may improve the accuracy, but this may also increase both the storage requirements and execution times. In our experiments, we have set  $u = 0.1$ .

**NSPIV.** This code computes an  $LU$ -decomposition of  $A$  using partial pivoting. The elimination is carried out row by row. Storage for fill-in is allocated during the numerical decomposition phase. The user is responsible for the choice of initial row and column orderings. In our experiments, the initial orderings were those suggested by Sherman [17]. The column ordering was the original ordering of the variables, and the rows were arranged in increasing number of nonzeros. Experience indicates that it can be very efficient. However, the lower triangular matrix  $L$  is not saved. Thus, a potential drawback is that if it is used to solve several systems which have the same coefficient matrix, the factorization must be repeated for each new right-hand side.

*The method proposed in § 4.* The method we propose computes an  $LU$ -decomposition of  $A$  using Gaussian elimination with partial pivoting. Data structures for storing  $L_k$ ,  $1 \leq k \leq n-1$ , and  $U$  can be set up before the numerical computation begins by finding the structures of the Cholesky factors of  $A^T A$ . The numerical computation is then performed using the static data structure. The triangular matrices  $L_k$ ,  $1 \leq k \leq n-1$ , and  $U$  are saved so that solution of several systems with the same coefficient matrix is very convenient and efficient. A potential weakness is that it makes use of the structure of the Cholesky factor of the symmetric matrix  $A^T A$  which could be dense or severely overestimate the storage for  $L_k$  and  $U$ . (More on this can be found in § 6.) A good column ordering (that would yield low fill-in in the Cholesky factor of  $A^T A$ ) is chosen prior to the numerical computation. In our experiments, we have used a minimum degree ordering as the column ordering.

It should be mentioned that both MA28 and NSPIV have been superseded by new codes. New pivoting strategies which are based on the work of Zlatev [19] have been incorporated in a new version of MA28. It is claimed that these new strategies are effective on matrices arising from finite element problems. A new code, which is called NSPFAC, now replaces NSPIV and it computes (and saves) both the lower and upper triangular matrices using Gaussian elimination with a threshold pivoting technique. Unfortunately comparisons with these two codes cannot be made since we do not have access to them.

There are other packages which we have not considered. Examples include MA32 [5] and MA37 [8] from Harwell and the Yale sparse matrix package [10]. The package MA37 should be of particular interest. It computes an  $LU$ -decomposition of  $A$  using the so-called *multi-frontal technique* and is based on the ideas used in MA27 [9] for solving sparse symmetric indefinite systems. Based on our experience with MA27, we expect MA37 to be effective in terms of storage and execution times. Unfortunately we do not have a copy of MA37 available.

We now provide some numerical experiments to compare the performance of the various implementations. The experiments were carried out on an IBM 4341. The programs were written in ANSI standard FORTRAN and compiled using an IBM VS FORTRAN Optimizing compiler. Single precision arithmetic was used. The test prob-

lems include thirteen finite element and nine nonfinite element problems. (The nonfinite element problems were obtained from Harwell.) Their characteristics are given in Table 5.1. For the finite element problems, the numerical values for the coefficient matrices were generated using a uniform random number generator. For each of the twenty-two test problems, the right-hand side vector was chosen so that the solution vector contained all ones.

TABLE 5.1  
*Characteristics of test problems.*

Problem number	Number of unknowns	Number of nonzeros	Remarks
1	265	1,753	Graded-L finite element mesh.
2	406	2,716	Graded-L finite element mesh.
3	577	3,889	Graded-L finite element mesh.
4	778	5,272	Graded-L finite element mesh.
5	936	6,266	Finite element mesh—a hollow square (small hole).
6	1,009	6,865	Finite element mesh—a graded-L problem.
7	1,089	7,361	Finite element mesh—a square problem.
8	1,440	9,504	Finite element mesh—a hollow square (large hole).
9	1,180	7,750	Finite element mesh—a +-shaped problem.
10	1,377	8,993	Finite element mesh—an H-shaped problem.
11	1,138	7,450	Finite element mesh—a 3-hole problem.
12	1,141	7,465	Finite element mesh—a 6-hole problem.
13	1,349	9,101	Finite element mesh—a pinched hole problem.
14	113	655	Matrix pattern supplied by Morven Gentleman.
15	54	291	Matrix pattern supplied by Curtis.
16	57	281	Matrix pattern supplied by Willoughby.
17	199	701	Matrix pattern supplied by Willoughby.
18	130	1,037	Matrix from laser problem (A. R. Curtis).
19	363	3,279	Matrix from linear programming problem.
20	541	4,282	Facsimile convergence matrix.
21	991	6,027	Matrix from Philips Ltd (J. P. Whelan).
22	192	2,992	Matrix from parabolic pde.

Tables 5.2 and 5.3 contain respectively the storage and execution times required by the various methods. Storage requirements are given in terms of number of storage locations required, including space for pointers, subscripts, etc., and execution times are in seconds. Table 5.4 shows the accuracy achieved. We have used  $\|x - \bar{x}\|_\infty$  as a measure of the accuracy, where  $\bar{x}$  denotes the computed solution. Other terms used in the tables are explained below.

#### SPARSPAK.

analysis storage and analysis time—amount of space and time required to find an assignment for  $A$ , to find a symmetric ordering for  $A + A^T$  and to allocate space for the  $LU$ -decomposition of  $A$ .

solution storage and solution time—amount of space required to store the  $LU$ -decomposition and the time required for the computation.

total time—sum of analysis and solution times.

#### MA28 and NSPIV.

total storage—amount of space required to compute the  $LU$ -decomposition

total time—amount of time to compute the  $LU$ -decomposition (including the times required to do any structure analysis for MA28).



TABLE 5.2  
Storage requirements (in number of storage locations).

Problem	SPARSPAK		MA28	NSPIV	New method	
	analysis	solution	total	total	analysis	solution
1	5,362	9,795	21,985	17,326	10,902	18,765
2	8,275	17,184	37,716	38,747	17,007	35,239
3	11,818	26,688	71,700	55,152	24,461	55,672
4	15,991	38,347	93,449	96,863	33,267	84,716
5	19,081	42,874	119,831	81,527	39,257	96,581
6	20,794	51,712	134,336	123,914	43,422	119,039
7	22,346	52,803	147,364	141,046	46,406	114,967
8	29,089	61,818	155,042	106,907	59,081	125,702
9	23,761	42,966	98,066	71,675	48,061	76,957
10	27,626	47,739	107,349	82,694	55,566	83,870
11	22,867	46,194	108,391	113,869	46,123	94,512
12	22,918	46,294	120,558	85,876	46,266	98,366
13	27,646	64,876	192,496	133,844	57,306	143,317
14	3,050	3,900	3,897	2,932	5,494	4,229
15	983	1,002	1,851	1,609	1,835	1,575
16	1,022	935	1,781	1,422	1,730	1,310
17	3,716	6,268	6,493	7,240	5,044	7,435
18	3,903	3,285	5,057	4,401	12,247	7,305
19	13,948	18,799	17,219	19,934	20,280	23,132
20	14,722	23,308	39,846	99,696	31,798	45,252
21	20,471	73,540	178,779	164,498	57,220	279,483
22	7,329	9,271	21,136	21,011	16,033	15,894

*New method.*

analysis storage and analysis time—amount of space and time required to find an assignment for  $A$ , to determine a symmetric ordering for  $A^T A$  and to allocate space for the Cholesky factors of  $A^T A$ .

solution storage and solution time—amount of space required to store the Cholesky factors of  $A^T A$  and the time required to compute the  $LU$ -decomposition of  $A$ .

total time—sum of analysis time and solution time.

Following are some remarks on the results:

(1) Even though SPARSPAK requires the least amount of space and execution times in most cases, its accuracy is usually poor. This is expected since there is no pivoting for stability performed.

(2) The results indicate that the method proposed in this paper is quite competitive with both MA28 and NSPIV in most cases. In particular, for finite element problems, our method is certainly better than MA28, in terms of storage requirements, execution time and accuracy. The method may occasionally require a little more storage and execution time than NSPIV. However, it should be pointed out that NSPIV only stores the upper triangular matrix  $U$ , whereas in our case, we store *both* the lower and upper triangular matrices in the  $LU$ -decomposition. Furthermore, we have ignored the problem of finding “good” initial column and row orderings (if they exist) for NSPIV. Thus, taking these comments into account, it is fair to say that our method performs at least as well as NSPIV for these finite element problems. (Of course, both the storage

TABLE 5.3  
Execution times (in seconds).

Problem	SPARSPAK			MA28	NSPIV	New method		
	analysis	solution	total	total	total	analysis	solution	total
1	0.327	0.450	0.777	3.113	3.040	0.783	3.530	4.313
2	0.617	1.070	1.687	6.663	9.083	1.410	9.123	10.533
3	0.810	1.967	2.776	27.038	15.822	1.800	17.779	19.579
4	1.137	3.290	4.426	36.261	33.225	2.703	31.168	33.871
5	1.237	3.170	4.406	60.196	21.845	2.996	34.401	37.398
6	1.377	4.566	5.943	81.238	53.970	3.840	51.167	55.006
7	1.467	4.156	5.623	111.190	56.263	3.496	42.627	46.124
8	1.950	3.836	5.786	77.828	28.263	4.596	36.628	41.224
9	1.557	1.890	3.446	30.765	9.716	3.560	13.672	17.232
10	1.803	1.973	3.776	24.312	11.426	4.120	13.176	17.296
11	1.590	2.606	4.196	50.317	32.701	3.480	24.309	27.788
12	1.680	2.613	4.293	78.435	20.159	3.780	27.978	31.758
13	1.733	5.046	6.780	154.237	47.307	3.923	50.820	54.743
14	0.380	0.160	0.540	0.233	0.070	0.930	0.290	1.220
15	0.053	0.013	0.067	0.087	0.050	0.110	0.077	0.187
16	0.053	0.013	0.067	0.077	0.023	0.110	0.053	0.163
17	0.350	0.210	0.560	0.410	0.547	0.480	0.637	1.117
18	1.837	0.070	1.907	0.250	0.080	2.710	0.863	3.573
19	3.370	fail		1.280	2.823	4.366	3.606	7.973
20	11.363	1.503	12.866	5.043	1721.269	14.062	10.196	24.258
21	5.633	17.819	23.452	147.264	184.002	19.692	302.144	321.836
22	0.527	0.620	1.147	3.116	3.883	1.620	3.313	4.933

requirements and solution times for NSPIV may be improved if good initial column and row orderings are available.)

(3) For nonfinite element problems, the performance of our method is somewhat disappointing in some cases. Both the storage requirements and solution times are in general larger than those for MA28, but are comparable to NSPIV. A possible explanation is that the structures of the Cholesky factors of the permuted  $A^T A$  may have overestimated the actual amount of fill-in in the triangular matrices  $L_k$  and  $U$ . (Also see the discussion in (4).)

(4) Note that we have only proved that the structures of the triangular matrices obtained in the  $LU$ -decomposition of  $A$  are contained in the structures of the Cholesky factors of  $A^T A$ . There is a possibility that the amount of space allocated for the Cholesky factors of  $A^T A$  may be much larger than that required to store the  $LU$ -decomposition. In order to explore this question, we have shown in Table 5.5 the percentage of storage that is actually utilized. The results indicate that, for the finite element problems, the utilization is approximately 50% for the lower triangular portion and 66% for the upper triangular portion. For the nonfinite element problems, the corresponding percentages are roughly 40% and 50% respectively. Moreover, there are examples in which the percentages of utilization are very low. For these examples, MA28 usually performs better than our method.

(5) One nice feature about the new method is that the amount of space allocated to store the  $LU$ -decomposition is not sensitive to the numerical values in  $A$  and the row interchanges. In other words, the same amount of space (or the same data structure)

TABLE 5.4  
Errors (in  $l_\infty$  norm).

Problem	SPARSPAK	MA28	NSPIV	New method
1	1.23-1	1.53-2	3.16-3	9.44-4
2	2.78-2	4.60-2	6.26-4	8.47-4
3	4.42-1	1.76-1	4.33-3	2.56-3
4	1.15-1	1.31-1	2.75-3	4.31-3
5	1.58+0	7.68-1	1.32-2	8.57-3
6	5.34-1	1.18+0	3.36-3	6.29-3
7	8.66-2	1.09-1	5.81-3	1.62-3
8	1.88+0	5.77+0	4.55-2	4.49-2
9	5.06-2	2.88-1	3.71-3	5.95-3
10	5.76-2	1.67-1	2.16-3	2.76-3
11	2.22+0	1.17+0	7.43-2	1.41-2
12	3.76-1	5.26+0	4.59-3	1.20-2
13	2.25-1	5.95+0	1.19-2	2.66-3
14	1.18+0	5.43-3	1.02-2	1.23-1
15	3.90-3	1.39-3	3.19-4	2.26-3
16	2.93-3	6.03-3	9.70-4	3.72-4
17	5.75+0	5.98-3	1.99-3	1.52-3
18	6.69-1	7.50-1	5.91-1	3.05-1
19	fail	1.30-3	1.90-3	1.27-2
20	4.29-5	7.63-6	6.68-6	1.34-5
21	2.22-4	6.42+0	3.17-4	5.02-4
22	1.30-5	2.43-3	1.56-5	1.16-4

can be used for different coefficient matrices as long as they have the same structure. In fact, after the first system has been solved, it is not necessary to determine the data structure again when subsequent systems having the same structure are to be solved. This is not true in MA28 and NSPIV, in which the amount of space depends on the numerical values and the row interchanges (unless one wants to decompose a new matrix having the same structure using the pivotal sequence obtained during the decomposition of the old matrix). To illustrate this, we have generated, for each of problems 1 and 2, three matrices that have different numerical values but the same nonzero structure. The results are given in Table 5.6. For MA28, the storage requirement is in general smaller if one decomposes a second matrix having an identical structure using a previously determined pivotal sequence. The same is true for our method. After the decomposition of the first matrix, we can postprocess the data structure to release any unused storage locations. This allows us to recover some storage from the data structure. Thus when we decompose a new matrix having the same structure using the same pivotal sequence, the storage requirement will also be smaller. Efficient implementation of this idea is under consideration.

(6) Note that the threshold pivoting technique in MA28 does not necessarily give satisfactory results for some of our test problems. In our experiments, we have assigned the threshold parameter  $u$  the value 0.1. Obviously, one can use a larger threshold parameter so as to obtain more accurate results. The tradeoffs are increases in storage requirements and execution times. Similar observation has been made elsewhere. For example, see [3].

(7) Finally, the amount of space reported for MA28 is the *minimum amount* required in order to solve a given problem. With the minimum amount of space, MA28

TABLE 5.5  
*Data structure utilization by L and U in the  
 new method (percentage).*

Problem	L	U
1	51.8	62.7
2	49.7	62.1
3	49.7	66.1
4	48.7	66.8
5	50.2	66.2
6	49.9	67.6
7	49.9	66.4
8	49.9	66.6
9	49.7	62.4
10	48.5	63.5
11	50.5	64.8
12	49.9	66.5
13	48.6	65.5
14	20.5	63.4
15	45.1	68.9
16	40.4	65.4
17	40.6	54.5
18	13.1	29.3
19	30.2	42.4
20	40.4	44.2
21	47.1	48.9
22	57.7	66.6

has to perform numerous data compressions and consequently requires substantial execution time. Thus, in order to reduce the execution time, it is common practice to provide more space than the minimum amount. The extra space is sometimes known as the *elbow room*. See [12] for some numerical experiments. In our experiments, we have provided a lot of elbow room so that data compressions do not occur.

TABLE 5.6.  
*Solution of systems with same nonzero structure.*

Method	n	265	265	265	406	406	406
MA28	total storage	21,985	22,702	20,503	37,716	40,743	37,870
	total time	3.113	3.650	2.750	6.663	11.100	8.750
	error	1.53-2	3.70-2	2.11-1	4.60-2	3.34-1	2.60-2
NSPIV	total storage	17,326	19,146	19,906	38,747	40,743	37,870
	total time	3.040	3.043	3.670	9.083	7.057	7.973
	error	3.16-3	1.51-4	3.01-3	6.26-4	5.31-3	1.85-3
New method	analysis storage	10,902	10,902	10,902	17,007	17,007	17,007
	solution storage	18,765	18,765	18,765	35,239	35,239	35,239
	analysis time	0.783	0.783	0.783	1.410	1.410	1.410
	solution time	3.530	3.537	3.557	9.123	9.270	9.287
	total time	4.313	4.320	4.340	10.533	10.680	10.697
	error	9.44-4	3.35-4	6.40-3	8.47-4	6.06-4	5.52-4

**6. Handling of dense rows.** Throughout our discussion in the previous sections, we have assumed that the symmetric matrix  $A^T A$  can be permuted symmetrically so that its Cholesky factor is sparse whenever  $A$  is sparse. However, there are some instances in which this assumption is invalid. An example is given below.

$$A = \begin{pmatrix} \times & \times & \times & \times \\ & \times & & \\ & & \times & \\ & & & \times \end{pmatrix}.$$

Clearly  $A^T A$  and its Cholesky factor are dense matrices, but the  $LU$ -decomposition of  $A$  is as sparse as the original matrix  $A$ .

This example illustrates the main disadvantage of the method we propose in this paper. The structures of the Cholesky factors of  $A^T A$  may overestimate the structures of the triangular matrices obtained in the  $LU$ -decomposition of  $A$ . Experience indicates that this is usually caused by a relatively small number of dense rows of  $A$ . (In the previous  $4 \times 4$  example, the first row is dense.) One way to handle this situation is described below and it is similar to a technique employed by Heath in the treatment of dense rows in the solution of sparse linear systems using orthogonal transformations [16]. For convenience, let  $A$  be partitioned into

$$A = \begin{pmatrix} B \\ C \end{pmatrix},$$

where  $B$  and  $C$  contain respectively the sparse and dense rows of  $A$ . Assume  $B$  is  $p \times n$  and  $C$  is  $(n-p) \times n$ . We also assume that  $B$  has a "zero-free diagonal"; that is,  $B_{ii} \neq 0$  for  $1 \leq i \leq p$ . Suppose a sparse  $LU$ -decomposition of  $B$  is given by

$$B = P_1 L_1 P_2 L_2 \cdots P_{p-1} L_{p-1} (R \ S),$$

where  $P_k$  is a  $p \times p$  permutation matrix,  $L_k$  is  $p \times p$  unit lower triangular,  $R$  is  $p \times p$  upper triangular, and  $S$  is  $p \times (n-p)$ . This can be achieved using the method we have proposed earlier. For simplicity, let

$$\bar{L} = P_1 L_1 P_2 L_2 \cdots P_{p-1} L_{p-1}.$$

That is,

$$B = \bar{L}(R \ S) = (\bar{L}R \ \bar{L}S).$$

Partition  $C$  into

$$C = (C_1 \ C_2),$$

where  $C_1$  and  $C_2$  are respectively  $(n-p) \times p$  and  $(n-p) \times (n-p)$ . Then we have

$$A = \begin{pmatrix} B \\ C \end{pmatrix} = \begin{pmatrix} \bar{L}R & \bar{L}S \\ C_1 & C_2 \end{pmatrix} = \begin{pmatrix} \bar{L} & 0 \\ 0 & I \end{pmatrix} \begin{pmatrix} R & S \\ C_1 & C_2 \end{pmatrix}.$$

Now we can eliminate  $C_1$  using block elimination. That is, we find an  $(n-p) \times p$  matrix  $V$  so that

$$\begin{pmatrix} R & S \\ C_1 & C_2 \end{pmatrix} = \begin{pmatrix} I & 0 \\ V & I \end{pmatrix} \begin{pmatrix} R & S \\ 0 & W \end{pmatrix}.$$

It is not hard to see that  $V = C_1 R^{-1}$  and  $W = C_2 - C_1 R^{-1} S$ . Then we can decompose

the  $(n-p) \times (n-p)$  matrix  $W$  using Gaussian elimination with partial pivoting:

$$W = \hat{P}_1 \hat{L}_1 \hat{P}_2 \hat{L}_2 \cdots \hat{P}_{n-p-1} \hat{L}_{n-p-1} T,$$

where  $\hat{P}_k$  is an  $(n-p) \times (n-p)$  permutation matrix,  $\hat{L}_k$  is  $(n-p) \times (n-p)$  unit lower triangular, and  $T$  is  $(n-p) \times (n-p)$  upper triangular. Let

$$\hat{L} = \hat{P}_1 \hat{L}_1 \hat{P}_2 \hat{L}_2 \cdots \hat{P}_{n-p-1} \hat{L}_{n-p-1}.$$

Then

$$W = \hat{L}T,$$

and

$$\begin{pmatrix} R & S \\ 0 & W \end{pmatrix} = \begin{pmatrix} I & 0 \\ 0 & \hat{L} \end{pmatrix} \begin{pmatrix} R & S \\ 0 & T \end{pmatrix}.$$

Combining all identities, we obtain the following decomposition.

$$A = \begin{pmatrix} B \\ C \end{pmatrix} = \begin{pmatrix} \bar{L} & 0 \\ 0 & I \end{pmatrix} \begin{pmatrix} I & 0 \\ V & I \end{pmatrix} \begin{pmatrix} I & 0 \\ 0 & \hat{L} \end{pmatrix} \begin{pmatrix} R & S \\ 0 & T \end{pmatrix} = \begin{pmatrix} \bar{L} & 0 \\ V & \hat{L} \end{pmatrix} \begin{pmatrix} R & S \\ 0 & T \end{pmatrix}.$$

Let the right-hand side vector  $b$  be partitioned into

$$b = \begin{pmatrix} c \\ d \end{pmatrix},$$

where  $c$  and  $d$  are respectively  $p$ - and  $(n-p)$ -vectors. Similarly, partition the solution vector  $x$  into

$$x = \begin{pmatrix} u \\ v \end{pmatrix},$$

where  $u$  and  $v$  are respectively  $p$ - and  $(n-p)$ -vectors. Then  $x$  is obtained by solving

$$\begin{pmatrix} \bar{L} & 0 \\ V & \hat{L} \end{pmatrix} \begin{pmatrix} e \\ f \end{pmatrix} = \begin{pmatrix} c \\ d \end{pmatrix},$$

and

$$\begin{pmatrix} R & S \\ 0 & T \end{pmatrix} \begin{pmatrix} u \\ v \end{pmatrix} = \begin{pmatrix} e \\ f \end{pmatrix}.$$

This approach will be effective if  $(n-p)$  is small. In that case, the matrices  $C$ ,  $V$  and  $W$  can be stored and processed as small dense matrices.

Suppose  $A$  has a zero-free diagonal. In practice, it may not be possible to partition  $A$  into

$$\begin{pmatrix} B \\ C \end{pmatrix}$$

such that  $B$  contains the sparse rows of  $A$  and at the same time has a zero-free diagonal. To illustrate this, consider the following example.

$$A = \begin{pmatrix} \times & 0 & 0 & 0 & 0 \\ 0 & \times & 0 & 0 & 0 \\ \times & \times & \times & \times & \times \\ 0 & 0 & 0 & \times & 0 \\ 0 & 0 & 0 & 0 & \times \end{pmatrix}.$$

Thus,

$$B = \begin{pmatrix} \times & 0 & 0 & 0 & 0 \\ 0 & \times & 0 & 0 & 0 \\ 0 & 0 & 0 & \times & 0 \\ 0 & 0 & 0 & 0 & \times \end{pmatrix},$$

and it does not have a zero-free diagonal. To solve this problem, we use the following approach in our implementation. Let  $B$  be the  $n \times n$  matrix obtained from  $A$  by replacing the dense rows by null rows. Thus, in the previous example,

$$B = \begin{pmatrix} \times & 0 & 0 & 0 & 0 \\ 0 & \times & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \times & 0 \\ 0 & 0 & 0 & 0 & \times \end{pmatrix}.$$

Then we perform the  $LU$ -decomposition on  $B$  using Gaussian elimination with partial pivoting, skipping any step where we encounter a zero pivot. The resulting  $n \times n$  upper triangular matrix, denoted by  $U$ , will be a (row) permuted form of

$$\begin{pmatrix} R & S \\ 0 & 0 \end{pmatrix}.$$

Finally, to eliminate  $C_1$ , all we have to do is to identify those rows in  $U$  whose diagonal elements are nonzero.

**7. Concluding remarks and open problems.** In this paper we have considered the solution of the sparse linear system  $Ax = b$  using Gaussian elimination with partial pivoting. We have proved that the structures of the triangular matrices obtained in the  $LU$ -decomposition of the  $n \times n$  matrix  $A$  are contained in the structures of the Cholesky factors of the symmetric positive definite matrix  $A^T A$ , regardless of the choice of the row interchanges. These results are important since they allow us to implement Gaussian elimination with partial pivoting using a static data structure which is obtained by analyzing the structure of  $A^T A$ . The latter can be achieved efficiently using techniques developed for solving sparse symmetric positive definite systems. As a result, the overhead (in terms of storage for pointers and execution times) involved in the numerical computation is smaller than that needed by most existing methods which usually employ dynamic data structures. Preliminary numerical experiments indicate that, in general, the method we have proposed can be quite competitive with existing methods for solving general sparse systems of linear equations. (The results derived in this paper can also be obtained using a bipartite graph model which will be described elsewhere.)

Clearly, our approach will perform poorly if the matrix  $A^T A$  and its Cholesky factor are dense. Experience has indicated that this usually occurs when  $A$  has a relatively small number of dense rows. We have derived an algorithm to cope with this situation. However the problem of identifying dense rows remains open. (This problem does not only occur in our scheme; it also occurs in the solution of sparse least squares problems and sparse underdetermined systems using orthogonal decomposition. See [11], [13] for details.) A common strategy is to use the number of nonzeros in a row; a row will be regarded as dense if the nonzero count is larger than certain threshold which is usually problem-dependent.





## REFERENCES

- [1] I. S. DUFF, *Analysis of sparse systems*, D. Phil. Thesis, Oxford Univ., Cambridge, 1972.
- [2] ———, MA28—*A set of FORTRAN subroutines for sparse unsymmetric linear equations*, Tech. Report AERE R-8730, Harwell, 1977.
- [3] ———, *Practical comparisons of codes for the solution of sparse linear systems*, in *Sparse Matrix Proceedings 1978*, I. S. Duff and G. W. Stewart, eds., Society for Industrial and Applied Mathematics, pp. 107–134.
- [4] ———, *Algorithm 575. Permutations for a zero-free diagonal*, *ACM Trans. Math. Software*, 7 (1981), pp. 387–390.
- [5] ———, *The design and use of a frontal scheme for solving sparse unsymmetric equations*, in *Proc. Third IIMAS Workshop on Numerical Analysis*, 1981, J. P. Hennart, ed., *Lecture Notes in Mathematics* 909, Springer-Verlag, New York, 1982, pp. 240–247.
- [6] I. S. DUFF AND J. K. REID, *An implementation of Tarjan's algorithm for the block triangularization of a matrix*, *ACM Trans. Math. Software*, 4 (1978), pp. 137–147.
- [7] ———, *Some design features of a sparse matrix code*, *ACM Trans. Math. Software*, 5 (1979), pp. 18–35.
- [8] ———, *The multifrontal solution of unsymmetric sets of linear equations*, Report CSS 133, Harwell, 1983; this *Journal*, 5 (1984), pp. 633–641.
- [9] ———, *The multifrontal solution of indefinite sparse symmetric linear equations*, *ACM Trans. Math. Software*, 9 (1983), pp. 302–325.
- [10] S. C. EISENSTAT, M. C. GURSKY, M. H. SCHULTZ AND A. H. SHERMAN, *Yale sparse matrix package, II, the nonsymmetric codes*, Research Report 114, Dept. Computer Science, Yale Univ., New Haven, CT, 1977.
- [11] J. A. GEORGE AND M. T. HEATH, *Solution of sparse linear least squares problems using Givens rotations*, *Linear Algebra Appl.*, 34 (1980), pp. 69–83.
- [12] J. A. GEORGE, M. T. HEATH AND E. G. Y. NG, *A comparison of some methods for solving sparse linear least squares problems*, this *Journal*, 4 (1983), pp. 177–187.
- [13] ———, *Solution of sparse underdetermined systems of linear equations*, this *Journal*, to appear.
- [14] J. A. GEORGE AND J. W. H. LIU, *The design of a user interface for a sparse matrix package*, *ACM Trans. Math. Software*, 5 (1979), pp. 134–162.
- [15] ———, *Computer Solution of Large Sparse Positive Definite Systems*, Prentice-Hall, Englewood Cliffs, NJ, 1981.
- [16] M. T. HEATH, *Some extensions of an algorithm for sparse linear least squares problems*, this *Journal*, 3 (1982), pp. 223–237.
- [17] A. H. SHERMAN, *Algorithm 533. NSPIV, a FORTRAN subroutine for sparse Gaussian elimination with partial pivoting*, *ACM Trans. Math. Software*, 4 (1978), pp. 391–398.
- [18] M. YANNAKAKIS, *Computing the minimum fill-in is NP-complete*, *SIAM J. Alg. Disc. Meth.*, 2 (1981), pp. 77–79.
- [19] Z. ZLATEV, *On some pivotal strategies in Gaussian elimination by sparse technique*, *SIAM J. Numer. Anal.*, 17 (1980), pp. 18–30.

## AN ANALYSIS OF THE TOTAL APPROXIMATION PROBLEM IN SEPARABLE NORMS, AND AN ALGORITHM FOR THE TOTAL $l_1$ PROBLEM\*

M. R. OSBORNE† AND G. A. WATSON‡

**Abstract.** In many data fitting problems there are errors in the  $m \times n$  data matrix  $M$  as well as in the observed vector  $\mathbf{f} \in R^m$ . It is possible to take account of this by formulating and solving the total approximation problem in which some norm of the  $m \times (n+1)$  error matrix is minimized. For a general class of matrix norms, which we call separable, it is shown that the solution is a rank one matrix, and that the problem may be solved when the solution is known to a vector norm minimization problem on  $R^m$  with a single equality constraint. Attention is focussed on the total  $l_1$  problem. A finite descent algorithm is developed, and numerical results illustrating it are given.

**Key words.** total approximation problem, separable norms, resistant estimation procedure,  $l_1$  norm, finite descent algorithm

**1. Introduction.** The standard linear regression problem requires that  $\mathbf{x} \in R^n$  be found by solving the problem

$$(1.1) \quad \min_{\mathbf{x}} \|\mathbf{r}\|: \mathbf{r} = M\mathbf{x} - \mathbf{f},$$

where  $M: R^n \rightarrow R^m$  and  $\mathbf{f} \in R^m$  are given, and where  $M$  has full column rank  $n < m$ . Usually it is assumed that the expected values of the  $r_i$  are zero (for example, as experimental errors in observing  $\mathbf{f}$ ). If  $\|\cdot\|$  is the Euclidean vector norm then (1.1) can be interpreted as giving the maximum likelihood estimate for  $\mathbf{x}$  under the assumption that the  $r_i$  are independent and identically normally distributed random variables. However, there is interest in estimates for  $\mathbf{x}$  which have a certain resistance to isolated gross errors. A suitable procedure corresponds to choosing the norm in (1.1) to be the  $l_1$  norm

$$(1.2) \quad \|\mathbf{r}\|_1 = \sum_{i=1}^m |r_i|.$$

This norm is not differentiable if any  $r_i = 0$ , and it is useful to introduce the subdifferential of a convex function to characterize the minimum in (1.1). Let  $\sigma$  be the index set

$$(1.3) \quad \sigma = \{i: r_i = 0\}$$

where the number of elements in  $\sigma$  is  $|\sigma| = k$ , where there is an assumed enumeration of the elements so that  $\sigma(j) \in \sigma, j = 1, 2, \dots, k$ , and where the dependence on  $\mathbf{x}$  is implicit. Then the subdifferential of  $\|\mathbf{r}\|_1$  with respect to  $\mathbf{x}$  is given by

$$(1.4) \quad \partial_{\mathbf{x}} \|\mathbf{r}(\mathbf{x})\|_1 = \sum_{i \in \sigma^c} \theta_i \mathbf{m}_i^T + \sum_{i \in \sigma} [-1, 1] \mathbf{m}_i^T$$

where  $\theta_i = \text{sgn}(r_i)$ ,  $\mathbf{m}_i^T = \boldsymbol{\rho}_i(M)$ , the  $i$ th row of  $M$  (the corresponding column operator

\* Received by the editors March 7, 1983.

† Department of Statistics, Institute of Advanced Studies, Australian National University, Canberra, Australia.

‡ Department of Mathematical Sciences, University of Dundee, Dundee DD14HN, Scotland. The work of this author was carried out while a Visiting Fellow in the Institute of Advanced Studies, Department of Statistics, and was supported by a grant from the Mathematics Research Centre, Australian National University. This visit was assisted by a grant from the Carnegie Trust for the Universities of Scotland.

is  $\kappa_i(M)$ , and  $\sigma^c$  denotes the complement of  $\sigma$ . The condition for a minimum at  $\hat{\mathbf{x}}$  is

$$(1.5) \quad 0 \in \partial_x \|\mathbf{r}(\hat{\mathbf{x}})\|_1$$

so that there exist scalars  $u_i$ ,  $-1 \leq u_i \leq 1$ ,  $i \in \sigma$ , such that

$$(1.6) \quad 0 = \sum_{i \in \sigma^c} \theta_i \mathbf{m}_i + \sum_{i \in \sigma} u_i \mathbf{m}_i.$$

There always exists a minimizer  $\hat{\mathbf{x}}$  for which  $\dim(\mathbf{m}_i, i \in \sigma) = n$  (for example, see Watson [8]), and in practice it is convenient to ignore the possibility  $|\sigma| > n$  which corresponds to a degenerate situation. Then  $\hat{\mathbf{x}}$  is determined by the system of linear equations

$$(1.7) \quad r_i(\mathbf{x}) = 0, \quad i \in \sigma.$$

In particular,  $\hat{\mathbf{x}}$  is independent of  $f_i$ ,  $i \in \sigma^c$ , and (1.6) is unchanged by (possibly very large) perturbations of the  $f_i$ ,  $i \in \sigma^c$  which do not change  $\text{sgn}(r_i)$ ,  $i \in \sigma^c$ , so that

$$(1.8) \quad \frac{\partial \mathbf{x}}{\partial f_i} = 0, \quad \frac{\partial \mathbf{u}}{\partial f_i} = 0, \quad i \in \sigma^c.$$

We say that an estimator with these properties is resistant.

The assumption that the errors are restricted to the dependent variable  $\mathbf{f}$  in (1.1) can be a significant over-simplification. However, the more general situation of the errors in variables model is complicated by the need to make further assumptions about the error structure in order to ensure identifiability (Moran [7]). If the model equations in the errors in variables problem are

$$(1.9) \quad (M + N)\mathbf{x} = \mathbf{f} + \mathbf{r},$$

and if the errors are assumed to be independent and identically normally distributed with zero mean, then the maximum likelihood method leads to the minimization problem

$$(1.10) \quad \min_{\mathbf{x}} \|[N|\mathbf{r}]\|_F$$

subject to (1.9) where the norm is the Frobenius matrix norm. A generalization of this "total least squares problem" is considered by Golub and Van Loan [5]. They solve

$$(1.11) \quad \min_{\mathbf{x}} \|T_1[N|\mathbf{r}]T_2\|_F$$

subject to (1.9) where  $T_1, T_2$  are positive diagonal matrices. Let  $Z = T_1[M|\mathbf{f}]T_2$ , and let  $\mathbf{v}, \mathbf{v}^T \mathbf{v} = 1$ , be a singular vector associated with the smallest singular value of  $Z$ . Then if  $v_{n+1} \neq 0$

$$(1.12) \quad T_1[N|\mathbf{r}]T_2 = -Z\mathbf{v}\mathbf{v}^T$$

is a solution to (1.11). The condition on  $v_{n+1}$  is necessary to ensure the existence of a minimizing  $\mathbf{x}$ . However, the problem of finding the perturbation matrix of smallest norm such that  $Z + T_1[N|\mathbf{r}]T_2$  does not have full rank is well determined.

*Example 1.1.* Let

$$M = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 1 \end{bmatrix}, \quad \mathbf{f} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}, \quad T_1 = I_3, \quad T_2 = I_2.$$

Then the smallest singular value of  $Z$  is 1 and the corresponding singular vector is  $\mathbf{v} = (1/\sqrt{2})(1, -1, 0)^T$ . Thus

$$[N|\mathbf{r}] = \frac{1}{2} \begin{bmatrix} -1 & 1 & 0 \\ 1 & -1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

and has norm 1. However

$$\mathbf{f} + \mathbf{r} \notin \text{range}(M + N).$$

The striking feature of (1.12) is that it is a rank one matrix. In this paper we generalise this result to *total approximation problems* for a class of norms which we call separable and at the same time extend results given in Watson [9]. This class of problems includes the total least squares problem and also the total least  $p$ th problem for  $p \geq 1$ . We show that

- (i) the solution of the total approximation problem

$$\min \|[N|\mathbf{r}]\|$$

subject to (1.9) has rank one whenever the norm is separable, and

- (ii) the solution can be computed when the solution to a *vector* minimization problem subject to a single normalizing constraint is known. We concentrate on the total  $l_1$  problem and show that in the absence of degeneracy the solution has the characteristic property that *only one column of  $[N|\mathbf{r}]$  is different from zero*. It is possible to determine the resistance of the total  $l_1$  problem by examining the sensitivity of the solution to perturbations in the problem data, and we obtain the somewhat disappointing result that this cannot be expected to have the strong resistance properties associated with the  $l_1$  regression problem. An effective descent algorithm for finding a local minimizing solution is presented and its finiteness established in the nondegenerate case. Finally numerical results are given which show that a number of local solutions may exist, highlighting the nonconvex nature of the problem.

**2. Separable norms and the total approximation problem.** Let the matrix  $Z: R^{n+1} \rightarrow R^m$  having full column rank  $n + 1 \leq m$  be given. We consider the total approximation problem in the form: determine a perturbation matrix  $E$  such that, for a given norm,

- (i)  $\|E\|$  is a minimum, and
- (ii)  $\text{rank}(Z + E) < n + 1$ .

If we consider a matrix as a particular organisation of an extended vector in  $R^{m \times (n+1)}$  then we can readily define a suitable class of matrix norms, as the required concepts from vector analysis generalise in a straightforward manner. For example, we can define the subdifferential of  $\|Z\|$  by

$$\partial\|Z\| = \{G: R^{n+1} \rightarrow R^m; \|S\| \geq \|Z\| + \text{trace}(SX - Z)^T G, \forall S: R^{n+1} \rightarrow R^m\}.$$

It is easily seen that  $G \in \partial\|Z\|$  is equivalent to the statements

- (i)  $\|Z\| = \text{trace}(G^T Z)$ , and
- (ii)  $\|G\|^* \leq 1$ ,

where

$$\|G\|^* = \max_{\|S\| \leq 1} \text{trace}(S^T G)$$

and  $\|\cdot\|^*$  is the polar or dual norm to  $\|\cdot\|$ . It should be noted that the role of the norm and its dual can be interchanged in this definition.

Here we restrict attention to norms on  $E$  which are *separable*.

DEFINITION. The norm  $\|\cdot\|$  on the set of  $m \times (n+1)$  matrices is separable if there exist vector norms  $\|\cdot\|_R$  and  $\|\cdot\|_D$  on  $R^m$  and  $R^{n+1}$ , the range and domain spaces respectively, such that for all  $\mathbf{u} \in R^m, \mathbf{v} \in R^{n+1}$

$$(2.1a) \quad (i) \quad \|\mathbf{u}\mathbf{v}^T\| = \|\mathbf{u}\|_R \|\mathbf{v}\|_D^*$$

and

$$(2.1b) \quad (ii) \quad \|\mathbf{u}\mathbf{v}^T\|^* = \|\mathbf{u}\|_R^* \|\mathbf{v}\|_D.$$

The second condition is needed to prove Lemma 2.1 which states a key property of the subdifferential of a separable norm.

LEMMA 2.1. Let  $\|\cdot\|$  be separable. Then

$$(2.2) \quad Z = \mathbf{u}\mathbf{v}^T \Rightarrow G = \mathbf{s}\mathbf{t}^T \in \partial\|Z\|$$

where  $\mathbf{s} \in \partial\|\mathbf{u}\|_R$  and  $\mathbf{t} \in \partial\|\mathbf{v}\|_D^*$ .

Proof. It follows from the definitions of  $\mathbf{s}$  and  $\mathbf{t}$  and (2.1a) that

$$\text{trace}(G^T \mathbf{u}\mathbf{v}^T) = \mathbf{v}^T G^T \mathbf{u} = (\mathbf{v}^T \mathbf{t})(\mathbf{s}^T \mathbf{u}) = \|\mathbf{u}\|_R \|\mathbf{v}\|_D^* = \|\mathbf{u}\mathbf{v}^T\|,$$

where we have used the result that if  $\mathbf{w} \in \partial\|z\|$  then  $\|z\| = \mathbf{w}^T z$ . Thus  $G \in \partial\|\mathbf{u}\mathbf{v}^T\|$  provided  $\|G\|^* \leq 1$ . But this is an immediate consequence of (2.1b) as  $\|\mathbf{s}\|_R^*, \|\mathbf{t}\|_D \leq 1$  follow from the assumptions.

We now prove our main result for separable norms.

THEOREM 2.1. Let  $\|\cdot\|$  be separable. Then

$$(2.3) \quad \|Z\| \cong \max_{\|\mathbf{v}\|_D \leq 1} \|Z\mathbf{v}\|_R.$$

Proof. Let  $\mathbf{v}$  solve the optimization problem in (2.3). Then the necessary conditions for an optimum are that there exist  $\mathbf{y} \in \partial\|Z\mathbf{v}\|_R, \mathbf{w} \in \partial\|\mathbf{v}\|_D, \gamma \cong 0$  such that

$$\mathbf{y}^T Z - \gamma \mathbf{w}^T = 0$$

(for example see [6]). Taking the scalar product with  $\mathbf{v}$  gives

$$\gamma = \|Z\mathbf{v}\|_R.$$

Let  $G = \mathbf{y}\mathbf{v}^T$ . As  $\|G\|^* \leq 1$  by (2.1b) it follows that

$$\|Z\| \cong \text{trace}(G^T Z) = \text{trace}(\mathbf{y}\mathbf{v}^T Z) = \mathbf{y}^T Z\mathbf{v} = \gamma.$$

Example 2.1. (i) If the norm is defined by requiring that equality hold in (2.3) then it is an operator or subordinate norm. In this case

$$\|Z\| = \max_{\substack{\|\mathbf{y}\|_R^* = 1 \\ \|\mathbf{v}\|_D = 1}} \mathbf{y}^T Z\mathbf{v}$$

so that  $G = \mathbf{y}\mathbf{v}^T \in \partial\|Z\|$ . That (2.1a) is satisfied is an immediate consequence. Equation (2.1b) requires that

$$\|\mathbf{s}\|_R^* \|\mathbf{t}\|_D = \max_{\|S\| \leq 1} \text{trace}(S^T \mathbf{s}\mathbf{t}^T).$$

This follows directly from the constraint on  $S$  which gives

$$1 \cong \max_{\substack{\|\mathbf{y}\|_R^* = 1 \\ \|\mathbf{w}\|_D = 1}} \text{trace}(S\mathbf{y}\mathbf{w}^T).$$

The total approximation problem in operator norms is discussed in Watson [9].

(ii) In the case considered by Golub and van Loan [5] (see (1.11)) we verify (2.1a) only as (2.1b) is similar. We have

$$\|\mathbf{u}\mathbf{v}^T\|^2 = \|T_1\mathbf{u}\mathbf{v}^T T_2\|_F^2 = \sum_{i=1}^{n+1} v_i^2 (T_2)_i^2 \|T_1\mathbf{u}\|_2^2 = \|T_1\mathbf{u}\|_2^2 \|T_2\mathbf{v}\|_2^2.$$

Thus  $\|\mathbf{u}\|_R = \|T_1\mathbf{u}\|_2$ ,  $\|\mathbf{v}\|_D = \|T_2^{-1}\mathbf{v}\|_2$ , and  $\|Z\|^* = \|T_1^{-1}ZT_2^{-1}\|_F$ . Theorem 2.1 gives

$$\|Z\| \cong \max_{\mathbf{v}^T T_2^{-2}\mathbf{v}=1} \|T_1 Z \mathbf{v}\|_2 = \max_{\mathbf{w}^T \mathbf{w}=1} \|T_1 Z T_2 \mathbf{w}\|_2.$$

This shows that  $\|Z\|$  is bounded below by the largest singular value of  $T_1 Z T_2$ .

(iii) In the  $l_p$  norm

$$\|Z\| = \left\{ \sum_{i=1}^m \sum_{j=1}^{n+1} |Z_{ij}|^p \right\}^{1/p}.$$

It follows that

$$\|\mathbf{u}\mathbf{v}^T\|^p = \sum_{j=1}^{n+1} |v_j|^p \sum_{i=1}^m |u_i|^p,$$

so that

$$\|\mathbf{u}\mathbf{v}^T\| = \|\mathbf{u}\|_p \|\mathbf{v}\|_p.$$

Thus  $\|\cdot\|_R$  is the  $p$  norm on  $R^m$  and  $\|\cdot\|_D$  is the  $q$  norm on  $R^{n+1}$  where  $1/p + 1/q = 1$ . This verifies (2.1a). Here (2.1b) is similar with  $p$  replaced by  $q$ .

(iv) It is not difficult to find meaningful norms which are not separable. Consider, for example, the case where the errors in different columns are independent, but where the errors in each particular column have non-trivial variance-covariance matrices. Under the assumption of normal errors the method of maximum likelihood leads to the minimization of  $\|E\|^2 = \sum_{i=1}^{n+1} \boldsymbol{\kappa}_i(E)^T W_i^{-1} \boldsymbol{\kappa}_i(E)$  where  $W_i$  is the variance-covariance matrix of the errors in the  $i$ th variable (assumed known). Here

$$\|\mathbf{u}\mathbf{v}^T\|^2 = \sum_{i=1}^{n+1} v_i^2 \mathbf{u}^T W_i^{-1} \mathbf{u}$$

and is separable only if  $\mathbf{u}^T W_i^{-1} \mathbf{u} = k_i \|\mathbf{u}\|_R^2$ .

**THEOREM 2.2.** *Let  $\|\cdot\|$  be any separable norm. Then the problem*

$$\min \|E\|$$

*subject to*

$$(2.4) \quad \text{rank}(Z + E) < n + 1$$

*has the solution*

$$(2.5) \quad E = -Z\hat{\mathbf{v}}\hat{\mathbf{w}}^T$$

*where  $\hat{\mathbf{v}}$  solves the problem*

$$(2.6) \quad \min_{\|\mathbf{v}\|_D=1} \|Z\mathbf{v}\|_R,$$

*and  $\hat{\mathbf{w}} \in \partial\|\hat{\mathbf{v}}\|_D$ .*

*Proof.* If  $E$  satisfies the rank condition in (2.4) then there exists  $\bar{\mathbf{v}}$ ,  $\|\bar{\mathbf{v}}\|_D = 1$ , such that

$$Z\bar{\mathbf{v}} = -E\bar{\mathbf{v}}.$$

Thus, using Theorem 2.1, for each feasible  $E$

$$(2.7) \quad \min_{\|\mathbf{v}\|_D=1} \|Z\mathbf{v}\|_R \leq \|Z\hat{\mathbf{v}}\|_R = \|E\hat{\mathbf{v}}\|_R \leq \|E\|.$$

However,  $E = -Z\hat{\mathbf{v}}\hat{\mathbf{w}}^T$  satisfies the rank condition (since  $(Z + E)\hat{\mathbf{v}} = 0$ ) and gives equality in (2.7) as a consequence of (2.1a).

*Remark 2.1.* The vector  $\mathbf{x}$  satisfying (1.9) can be deduced from (2.5), (2.6) by noting that  $Z = [M | \mathbf{f}]$  so that

$$(2.8) \quad \begin{bmatrix} \mathbf{x} \\ -1 \end{bmatrix} = -\frac{1}{\hat{\mathbf{v}}_{n+1}} \hat{\mathbf{v}}$$

provided  $\hat{\mathbf{v}}_{n+1} \neq 0$ .

The minimization problem (2.6) is not convex (the feasible region is the *outside* of the unit ball  $\|\mathbf{v}\|_D = 1$ ), so that uniqueness is not immediately guaranteed. In fact local minima are possible, and the interpretation of these is aided by considering the Lagrange multiplier formulation of (2.4). To be specific we say that a point is a *stationary point* if the first order necessary conditions are satisfied. If, in addition, there exists no feasible descent direction then the point is a *local minimum*, and it is an *isolated (or strong) local minimum* if the objective function increases in every feasible direction. The appropriate Lagrangian function is

$$(2.9) \quad \mathcal{L} = \|E\| + \boldsymbol{\lambda}^T (Z + E)\mathbf{v} + \gamma(\|\mathbf{v}\|_D - 1)$$

where the equality constraints

$$(2.10) \quad (Z + E)\mathbf{v} = 0$$

ensure that  $\text{rank}(Z + E) < n + 1$ , while the constraint  $\|\mathbf{v}\|_D = 1$  serves as a normalising condition. To develop multiplier relations for this equality constrain problem it is necessary that the matrix

$$\left[ \begin{array}{c|c|c|c|c} Z + E & v_1 I_m & v_2 I_m & \cdots & v_{n+1} I_m \\ \hline \mathbf{w}^T & 0 & 0 & \cdots & 0 \end{array} \right]$$

have full row rank where  $\mathbf{w} \in \partial\|\mathbf{v}\|_D$  (Hiriart-Urruty [6]). This condition is clearly satisfied as  $\mathbf{v}, \mathbf{w}$  necessarily  $\neq 0$  when  $\text{rank}(Z) = n + 1$ .

The multiplier conditions are developed in the following lemma.

**LEMMA 2.2.** *The necessary conditions for a solution of (2.4) are that there exist  $\boldsymbol{\lambda} \in R^m, G \in \partial\|E\|$  such that*

$$(2.11) \quad \boldsymbol{\lambda}^T (Z + E) = 0,$$

$$(2.12) \quad G + \boldsymbol{\lambda}\mathbf{v}^T = 0.$$

*Proof.* The necessary conditions that follow from considering (2.9) are

$$(2.13) \quad \boldsymbol{\lambda}^T (Z + E) + \gamma\mathbf{w}^T = 0$$

for some  $\mathbf{w} \in \partial\|\mathbf{v}\|_D$ , and

$$(2.14) \quad G_{ij} + \lambda_i v_j = 0, \quad i = 1, 2, \dots, m, \quad j = 1, 2, \dots, n + 1$$

for some  $G \in \partial\|E\|$ . Taking the scalar product of (2.13) with  $\mathbf{v}$  gives

$$\boldsymbol{\lambda}^T (Z + E)\mathbf{v} + \gamma\mathbf{w}^T \mathbf{v} = 0,$$

so that  $\gamma = 0$  by (2.10) and the result follows.

*Remark 2.2.* It is interesting that (2.14) shows that  $G$  always has rank one independent of the assumption of separability.

In order to establish the main result on local minima of (2.6) we require a preliminary lemma which is effectively a strengthened form of the first order necessary conditions for a solution of (2.6). A proof is given in Watson [9].

LEMMA 2.3. Let  $\hat{\mathbf{v}}$  be a local minimum of (2.6). Then for every  $\mathbf{w} \in \partial \|\hat{\mathbf{v}}\|_D$  there exists  $\mathbf{y} \in \partial \|Z\hat{\mathbf{v}}\|_R$  such that

$$(2.15) \quad \mathbf{y}^T Z = \|Z\hat{\mathbf{v}}\|_R \mathbf{w}^T.$$

THEOREM 2.3. Let  $\hat{\mathbf{v}}$  be a local minimum of (2.6) and  $\mathbf{w} \in \partial \|\hat{\mathbf{v}}\|_D$ . Then

$$E = -Z\hat{\mathbf{v}}\mathbf{w}^T$$

is a stationary point of the total approximation problem.

*Proof.* By Lemma 2.3 (2.15) is satisfied. Set

$$\hat{G} = -\mathbf{y}\hat{\mathbf{v}}^T.$$

By Lemma 2.1  $\hat{G} \in \partial \|E\|$ , and it remains to show that  $\lambda = \mathbf{y}$  satisfies (2.11). We have

$$\mathbf{y}^T (Z + E) = \mathbf{y}^T (Z - Z\hat{\mathbf{v}}\mathbf{w}^T) = \mathbf{y}^T Z - \|Z\hat{\mathbf{v}}\|_R \mathbf{w}^T = 0.$$

**3. The total  $l_1$  problem.** We now specialize to the norm

$$\|Z\| = \sum_{i=1}^m \sum_{j=1}^{n+1} |Z_{ij}|.$$

This is shown to be separable with

$$\|\mathbf{u}\|_R = \|\mathbf{u}\|_1 = \sum_{i=1}^n |u_i|$$

and

$$\|\mathbf{v}\|_D = \|\mathbf{v}\|_\infty = \max_{1 \leq i \leq n+1} |v_i|$$

in Example 2.1 (iii). It follows from Theorem 2.2 that the solution of the problem (2.4) is given by

$$\hat{E} = -Z\hat{\mathbf{v}}\mathbf{w}^T,$$

where  $\hat{\mathbf{v}}$  solves

$$(3.1) \quad \min_{\|\mathbf{v}\|_\infty=1} \|Z\mathbf{v}\|_1$$

and  $\mathbf{w} \in \partial \|\hat{\mathbf{v}}\|_\infty$ .

*Remark 3.1.* Here (2.15) holds with (compare (1.3), (1.4))

$$(3.2) \quad \mathbf{y}^T Z \in \partial_v \|Z\mathbf{v}\|_1 = \sum_{i \in \sigma^c} \theta_i \mathbf{z}_i^T + \sum_{i \in \sigma} [-1, 1] \mathbf{z}_i^T,$$

where

$$\sigma = \{i; \mathbf{z}_i^T \mathbf{v} = 0\},$$

$\theta_i = \text{sgn}(\mathbf{z}_i^T \mathbf{v})$ ,  $i \in \sigma^c$ , and  $\mathbf{z}_i = \boldsymbol{\rho}_i(Z)^T$ , and

$$\mathbf{w} \in \partial \|\mathbf{v}\|_\infty = \text{conv} \{\phi_j \mathbf{e}_j, j \in \nu\}$$



where  $\phi_j = \text{sgn}(v_j)$ , and

$$\nu = \{j; |v_j| = \|\mathbf{v}\|_\infty\}.$$

The special structure of the problem permits us to extend the results of the previous section.

**THEOREM 3.1.** *A necessary and sufficient condition for  $\hat{\mathbf{v}}$  to be a local solution of (3.1) is that for every  $\mathbf{w} \in \partial\|\hat{\mathbf{v}}\|_D$  there exist  $\mathbf{y} \in \partial\|Z\hat{\mathbf{v}}\|_R$  satisfying (2.15).*

*Proof.* Necessity follows from Lemma 2.3. Sufficiency is a consequence of Watson [9, Thm. 4] and uses the polyhedral nature of the constraint explicitly.

This result can be interpreted in terms of the number of zeros of  $Z\hat{\mathbf{v}}$  and extrema of  $\hat{\mathbf{v}}$ .

**THEOREM 3.2.** *Either there exists a solution  $\hat{\mathbf{v}}$  to (3.1) with  $|\sigma| + |\nu| \geq n + 1$ , or there is a nearby point which can be reached without increasing the objective function at which there is a downhill direction.*

*Proof.* Let  $\tilde{\mathbf{v}}$  be a solution to (3.1) with  $|\sigma| + |\nu| \leq n$ . Then there always exists a nontrivial  $\mathbf{s} \in R^{n+1}$  such that

$$\mathbf{z}_i^T \mathbf{s} = 0, \quad i \in \sigma, \quad s_j = 0, \quad j \in \nu,$$

and

$$(3.3) \quad \mathbf{g}^T \mathbf{s} \leq 0,$$

where

$$\mathbf{g} = \sum_{i \in \sigma^c} \theta_i \mathbf{z}_i.$$

It follows that for  $\tau > 0$  small enough

$$\|Z(\tilde{\mathbf{v}} + \tau\mathbf{s})\|_1 = \|Z\tilde{\mathbf{v}}\|_1 + \tau\mathbf{g}^T \mathbf{s} \leq \|Z\tilde{\mathbf{v}}\|_1$$

and

$$\|\tilde{\mathbf{v}} + \tau\mathbf{s}\|_\infty = \|\tilde{\mathbf{v}}\|_\infty = 1.$$

Thus  $\tilde{\mathbf{v}} + \tau\mathbf{s}$  must be a solution for all  $\tau$  small enough. However, there must be a first value of  $\tau$  for which the slope of one of the piecewise linear functions  $\|Z(\tilde{\mathbf{v}} + \tau\mathbf{s})\|_1$  and  $\|\tilde{\mathbf{v}} + \tau\mathbf{s}\|_\infty$  changes (both  $\rightarrow \infty$  as  $\tau \rightarrow \infty$ ). This must correspond either to a new zero residual ( $|\sigma| := |\sigma| + 1$ ), or a new extrema ( $|\nu| := |\nu| + 1$ ). If  $|\sigma|$  is increased then the argument can be repeated provided  $|\sigma| + |\nu| < n + 1$ . However, if  $|\nu| := |\nu| + 1$  then  $\|\tilde{\mathbf{v}} + \tau\mathbf{s}\|_\infty$  begins to increase and the objective function can be reduced by renormalization so that  $\mathbf{s}$  is downhill for minimizing  $\|Z\mathbf{v}\|_1$  at  $\tilde{\mathbf{v}} + \tau\mathbf{s}$ .

**Remark 3.2.** The existence of a downhill direction at a nearby point does not conflict with  $\tilde{\mathbf{v}}$  being a local minimum as this requires only that there is no downhill direction at  $\tilde{\mathbf{v}}$ . However, if  $\mathbf{s}$  is downhill at  $\tilde{\mathbf{v}} + \tau_0\mathbf{s}$  then  $\nu \cap \nu(\tau_0+) = \emptyset$ . The implication is that while it is possible to build up  $\sigma$  systematically,  $|\nu| > 1$  appears to happen only by chance, and it is argued that this corresponds to a degenerate situation.

The next theorem gives sufficient conditions for an isolated local solution when  $|\nu| = 1$ . The argument does not extend when  $|\nu| > 1$  (in fact it suggests how to construct a direction of nonuniqueness), and this provides additional evidence that the case  $|\nu| > 1$  at an isolated solution corresponds to a kind of degenerate behavior.

**THEOREM 3.3.** *Let  $|\nu| = 1$  and  $\hat{\mathbf{v}}$  be a local solution of (3.1). Then  $\hat{\mathbf{v}}$  is an isolated solution provided*

$$\text{rank}((\mathbf{z}_i, i \in \sigma, |y_i| < 1), \mathbf{w}) = n + 1,$$

where  $\mathbf{w} \in \partial\|\hat{\mathbf{v}}\|_\infty = \phi_k \mathbf{e}_k, k \in \nu$ .

*Proof.* Let  $s_k = 0$ , so that  $\tau\mathbf{s}$  is a feasible displacement at  $\hat{\mathbf{v}}$  for  $\tau > 0$  small enough. In this displacement

$$\|Z(\hat{\mathbf{v}} + \tau\mathbf{s})\|_1 - \|Z\hat{\mathbf{v}}\|_1 = \tau \left( \mathbf{g}^T \mathbf{s} + \sum_{i \in \sigma} |\mathbf{z}_i^T \mathbf{s}| \right).$$

Equation (2.15) applied to this case gives

$$\mathbf{g}^T \mathbf{s} + \sum_{i \in \sigma} y_i \mathbf{z}_i^T \mathbf{s} = \|Z\hat{\mathbf{v}}\|_1 \mathbf{w}^T \mathbf{s} = 0.$$

Thus

$$\mathbf{g}^T \mathbf{s} + \sum_{i \in \sigma} |\mathbf{z}_i^T \mathbf{s}| = \sum_{i \in \sigma} \{ |\mathbf{z}_i^T \mathbf{s}| - y_i \mathbf{z}_i^T \mathbf{s} \} \geq \sum_{i \in \sigma} \{ 1 - |y_i| \} |\mathbf{z}_i^T \mathbf{s}| > 0$$

as

$$|\mathbf{z}_i^T \mathbf{s}| = 0, \quad i \in \sigma \quad \text{and} \quad |y_i| < 1, \quad \text{and} \quad s_k = 0 \Rightarrow \mathbf{s} = 0$$

by the rank assumption. It is this last step which cannot always be applied when  $|\nu| > 1$ , and this provides the difficulty in extending the argument.

The above considerations lead us to define the usual case which will be the one for which a descent algorithm is developed in the next section.

DEFINITION 3.1. The point  $\mathbf{v}$  is *nondegenerate* if  $|\sigma| = k \leq n$ ,  $|\nu| = 1$  and

$$\text{rank}((\mathbf{z}_i, i \in \sigma), \mathbf{w}) = k + 1.$$

By Theorem 3.1 the search for nondegenerate minima is narrowed down to points at which  $|\sigma| = n$ . At a nondegenerate local minimum  $\hat{\mathbf{v}}$  only one column of  $E$  is different from zero.

To conclude this section we consider the resistance properties of the total  $l_1$  problem solution. Recall that in the  $l_1$  regression problem it is the  $f_i$  corresponding to the nonzero residuals that can be allowed to vary substantially. However, the above argument shows that there are comparatively few non-zero elements in  $\hat{E}$ . To show resistance, the analogous condition to (1.8) requires that we evaluate the rate of change of the solution quantities with respect to the elements of  $Z$ . To do this we note that  $\mathbf{v}$  and  $u_i, i \in \sigma$ , are determined by the conditions (2.15) which by (3.2), (3.3) are

$$(3.4) \quad \mathbf{g} + \sum_{i \in \sigma} u_i \mathbf{z}_i - \gamma \mathbf{w} = 0,$$

where

$$|\sigma| = n, \quad -1 \leq u_i \leq 1, \quad i \in \sigma, \quad \gamma = \|Z\hat{\mathbf{v}}\|_1, \quad \mathbf{w} = \phi_s \mathbf{e}_s, \quad \mathbf{z}_i^T \mathbf{v} = 0, \quad i \in \sigma,$$

and

$$\|\mathbf{v}\|_\infty = 1.$$

Let  $Z_{kl}$  be a general element of  $Z$  subject to the restriction that  $k \notin \sigma$ . Differentiating the above conditions with respect to  $Z_{kl}$  gives

$$\theta_k \mathbf{e}_l + \sum_{i \in \sigma} \frac{du_i}{dZ_{kl}} \mathbf{z}_i - \frac{d\gamma}{dZ_{kl}} \phi_s \mathbf{e}_s = 0,$$

$$\mathbf{z}_i^T \frac{d\mathbf{v}}{dZ_{kl}} = 0, \quad i \in \sigma,$$

$$\frac{dv_s}{dZ_{kl}} = 0.$$

In particular,  $d\mathbf{v}/dZ_{kl} = 0$  (this corresponds to the result obtained in the  $l_1$  regression problem), but  $(d/dZ_{kl})(\frac{u}{\gamma}) = \mathbf{a}$  constant vector  $\neq 0$  so the second of the resistance conditions (1.8) does not hold. The conditions that a perturbation

$$Z_{kl} \rightarrow Z_{kl} + \Delta_{kl}$$

leave  $\mathbf{v}$  unchanged become

$$(i) \quad \theta_k(\mathbf{z}_k^T \mathbf{v} + \Delta_{kl} v_l) > 0$$

and

$$(ii) \quad 1 - \left| u_i + \frac{du_i}{dZ_{kl}} \Delta_{kl} \right| > 0, \quad i \in \sigma.$$

It is this second condition that causes difficulty as it imposes strict limits on the size of possible perturbations.

**4. An algorithm for the  $l_1$  problem.** In this section we outline an algorithm for the total  $l_1$  problem. It is a modification of a reduced gradient algorithm for the  $l_1$  regression problem which can be shown to be equivalent to the well-known linear programming algorithms (this result does not seem to have been published, but the verification is just a matter of computation). For simplicity we again make appropriate nondegeneracy assumptions. Specifically, if  $\mathbf{v}$  is the current point then

- (i) the index set  $\sigma = \{i; \mathbf{z}_i^T \mathbf{v} = 0\}$  satisfies  $|\sigma| = k \leq n$ ,
- (ii) there is a unique index  $s$  such that

$$s = \arg \max (|v_i|, i = 1, 2, \dots, n + 1),$$

and

$$(iii) \quad \dim(H) = k + 1$$

where

$$H = \text{span}((\mathbf{z}_j, j \in \sigma), \mathbf{e}_s).$$

At the current point the algorithm consists of two main parts.

- (i) Let  $F(\mathbf{v}) = \|Z\mathbf{v}\|_1$ . Then a descent step is performed on the problem

$$\min_{v_s = \phi_s} F(\mathbf{v}).$$

Apart from this trivial modification to take account of the equality constraint this is just a step of the  $l_1$  solver. It has two components.

- (a) A descent vector  $\mathbf{t}$  is computed satisfying

$$F'(\mathbf{v}; \mathbf{t}) < 0, \quad t_s = 0.$$

If suitable  $\mathbf{t}$  cannot be found the computation is terminated.

- (b) Given  $\mathbf{t}$  we find  $\alpha$  to minimize  $F(\mathbf{v} + \alpha \mathbf{t})$  and set  $\tilde{\mathbf{v}} = \mathbf{v} + \alpha \mathbf{t}$ .
- (ii) It is possible that  $\|\tilde{\mathbf{v}}\|_\infty > 1$ . Thus  $\tilde{\mathbf{v}}$  must be renormalized. We have

$$\beta = 1 / \|\tilde{\mathbf{v}}\|_\infty \leq 1,$$

$$\mathbf{v} = \beta \tilde{\mathbf{v}},$$

$$F(\mathbf{v}) = \beta F(\tilde{\mathbf{v}}) \leq F(\tilde{\mathbf{v}}).$$

*The descent step.* The reduced gradient algorithm is used to generate the descent vector. Let  $B_\sigma : R^{n+1} \rightarrow R^{n+1}$  be given by

$$B_\sigma = [Z_\sigma | I_\sigma],$$

where

$$\kappa_i(Z_\sigma) = z_{\sigma(i)}, \quad i = 1, 2, \dots, k,$$

and  $I_\sigma$  consists of columns of the unit matrix including  $e_s$ , chosen to make  $\text{rank}(B_\sigma) = n + 1$ . It is convenient to make  $\kappa_{n+1}(B_\sigma) = e_s$ . Define

$$\mathbf{u} = \begin{bmatrix} \mathbf{u}_1 \\ \mathbf{u}_2 \end{bmatrix},$$

where  $\mathbf{u}_1 \in R^k$ ,  $\mathbf{u}_2 \in R^q$ ,  $q = n - k + 1$ , by

$$B_\sigma \mathbf{u} = -\mathbf{g},$$

where  $\mathbf{g}$  is given by (3.3). If  $\mathbf{u}_2 = \gamma \mathbf{e}_q$  then  $\mathbf{g} \in H$  and  $\mathbf{u}$  provides a tentative multiplier vector in (3.4). Thus the conditions for  $\mathbf{v}$  to be a nondegenerate, local solution of (3.1) are

$$(4.1) \quad \mathbf{g} \in H$$

and

$$(4.2) \quad -1 \leq u_i \leq 1, \quad i = 1, 2, \dots, k.$$

Otherwise there are two ways to compute a descent vector depending on which of (4.1) and (4.2) is violated.

(i)  $\mathbf{g} \notin H$  (so that  $\mathbf{u}_2 \neq \gamma \mathbf{e}_q$ ). Let

$$j = \arg \max (|u_i|, i = k + 1, \dots, n).$$

Then  $u_j \neq 0$  and

$$\mathbf{t} = \text{sgn}(u_j) \kappa_j(B_\sigma^{-T})$$

defines a descent direction. To verify this note that the directional derivative in the direction of  $\mathbf{t}$  is

$$F'(\mathbf{v}; \mathbf{t}) = \mathbf{g}^T \mathbf{t}$$

and

$$\mathbf{g}^T \mathbf{t} = -\text{sgn}(u_j) \rho_j(B_\sigma^{-1}) B_\sigma \mathbf{u} = -|u_j| < 0.$$

(ii)  $|u_j| - 1 > 0$  for some  $j$ ,  $1 \leq j \leq k$ . Then

$$\mathbf{t} = \text{sgn}(u_j) \kappa_j(B_\sigma^{-T})$$

defines a descent direction for

$$F'(\mathbf{v}; \mathbf{t}) = \mathbf{g}^T \mathbf{t} + |z_{\sigma(j)}^T \mathbf{t}| = -|u_j| + 1 < 0.$$

In this case

$$z_{\sigma(j)}^T \mathbf{t} = \text{sgn}(u_j) \neq 0$$

so that  $z_{\sigma(j)}^T \tilde{\mathbf{v}} \neq 0$  and  $\sigma := \sigma / \{\sigma(j)\}$ .

The second part of the descent step involves minimizing  $F$  in the direction determined by  $\mathbf{t}$  in order to compute  $\alpha$ . Now  $F$  is a piecewise linear function of  $\alpha$  with jumps in derivative at the points  $\alpha_i$  at which

$$z_i^T(\mathbf{v} + \alpha_i \mathbf{t}) = 0, \quad i \in \sigma^c.$$

Also, only values of  $\alpha_i > 0$  are of interest, and for  $\delta > 0$  small enough

$$F'(\mathbf{v} + (\alpha_j + \delta)\mathbf{t}; \mathbf{t}) - F'(\mathbf{v} + (\alpha_j - \delta)\mathbf{t}; \mathbf{t}) = 2|\mathbf{z}_j^T \mathbf{t}|.$$

Thus the point  $\alpha_p$  sought must correspond to the first  $p$  such that

$$F(\mathbf{v}; \mathbf{t}) + 2 \sum_{0 < \alpha_i \cong \alpha_p} |\mathbf{z}_i^T \mathbf{t}| \cong 0.$$

Efficient computation of  $\alpha_p$  has been discussed for the  $l_1$  regression problem. For example Bloomfield and Steiger [2] recommend Hoare's partitioning algorithm, while Clark and Osborne [3] find this satisfactory for the initial steps but note that the process settles down so that the number of terms in the summation is small after about  $n/2$  steps and suggest a comparison sort to order the small  $\alpha_i$ .

Given  $p, \sigma := \sigma \cup \{p\}$ .  $B_\sigma$  must now be updated by replacing either  $\mathbf{z}_{\sigma(j)}$  or  $\kappa_j(I_\sigma)$  by  $\mathbf{z}_p$  depending on the manner in which the descent vector is chosen. The exact mechanism of the updating and reorganization is determined by considerations of computational efficiency and stability (see Bartels and Golub [1], for example). It is a routine calculation to show that nonsingularity of  $B_\sigma$  is preserved in the updating.

Both possibilities (i) and (ii) can give adequate descent directions at the current point so that it may be necessary to choose between them. Assuming scale differences are not important our experience suggests a criterion of the form

$$j = \arg \max (|u_j| - 1, j = 1, 2, \dots, k, \mu |u_j|, j = k + 1, \dots, n)$$

where  $\mu > 0$  is chosen to favour deleting columns of  $I_\sigma$  unless there is a relatively large violation of the multiplier bound. This approach has the advantage that it does not require explicitly the rank of  $Z$  to be known in advance. In the rank defective case columns of  $I_\sigma$  in addition to the one corresponding to the norm constraint persist, and the corresponding components of  $\mathbf{v}$  are unchanged from their initial values as each descent vector is constructed to be orthogonal to the columns remaining in  $B_\sigma$ . Here this is a diagnostic device—without further assumptions the correct solution to the rank deficient problem is  $\hat{E} = 0$ .

*The renormalization step.* This step provides the main difference between our algorithm and a method for the  $l_1$  regression problem. The subgradient inequality gives

$$\|\mathbf{v} + \alpha \mathbf{t}\|_\infty \cong \|\mathbf{v}\|_\infty + \alpha \phi_s \mathbf{t}^T \mathbf{e}_s \cong \|\mathbf{v}\|_\infty,$$

so that if  $\|\mathbf{v} + \alpha \mathbf{t}\|_\infty > \|\mathbf{v}\|_\infty$  then

$$l = \arg \max (|v_i + \alpha t_i|, i = 1, 2, \dots, n + 1) \neq s.$$

Thus, apart from rescaling  $\tilde{\mathbf{v}}$  and  $F$ , it is necessary to make the update

$$\kappa_{n+1}(B_\sigma) \leftarrow \mathbf{e}_l \quad s := l.$$

Here the advantage of forcing the scaling constraint to be  $\kappa_{n+1}(B_\sigma)$  becomes clear because in the Bartels-Golub algorithm this step involves only the last two columns of  $B_\sigma$  (the scaling constraint having been moved to the penultimate column in the update which follows the descent step).

It is easy to take account of constraints of the form  $\kappa_i(E) = 0$  corresponding to columns of  $Z$  which are known exactly in the errors in variables application. It is necessary only to exclude these columns from consideration in the renormalization step so that components of  $\mathbf{v}$  associated with these columns are unconstrained.

*Remark 4.1.* If  $\mathbf{v}$  is chosen to be  $\mathbf{e}_s$  initially then at each subsequent step of the algorithm

$$\mathbf{v}^T B_\sigma \mathbf{u} = v_s u_{n+1} = \phi_s u_{n+1}$$

as  $\mathbf{v}^T \boldsymbol{\kappa}_i(Z_\sigma) = 0, i = 1, 2, \dots, k$  by the definition of  $\sigma$  and  $\mathbf{v}^T \boldsymbol{\kappa}(I_\sigma) = 0, i = 1, 2, \dots, q - 1$  by the choice of initial conditions. However,

$$\mathbf{v}^T B_\sigma \mathbf{u} = -\mathbf{v}^T \mathbf{g} = -F(\mathbf{v}).$$

Thus, at each step, the correct multiplier vector is given by the scaling constraint. This has value as a numerical check, but it also shows that information is not available from this source concerning the appropriateness or otherwise of the scaling constraint.

*Remark 4.2.* Finiteness of the algorithm in the nondegenerate case follows from the usual arguments. If the initial vector satisfies  $\mathbf{v} = \mathbf{e}_s$ , then at each subsequent step  $\mathbf{v}$  is uniquely determined by the linear equations

$$\mathbf{v}^T B_\sigma = \phi_s \mathbf{e}_{n+1}^T,$$

and each possible  $B_\sigma$  is associated with a particular value of  $F$ . As the algorithm reduces  $F$  at each step it follows that the configuration specified by  $B_\sigma$  cannot repeat. As there are only a finite number of such configurations finiteness is an immediate consequence.

To illustrate the use of the algorithm with Bartels–Golub updating it was applied to  $Z : R^5 \rightarrow R^{21}$  given in Table 4.1. This is the well-known stack loss data set considered in Daniel and Wood [4, Chap. 5]. In Table 4.2 values of  $F, \mathbf{v}, \mathbf{u}$  are given for the successive steps of the algorithm starting from  $\mathbf{v} = \mathbf{e}_5$ . In this case the same solution is found for each of the starting points  $\mathbf{v} = \mathbf{e}_j, j = 1, 2, \dots, 5$ , and the optimal scaling could be predicted by noting that the elements in  $\boldsymbol{\kappa}_1(Z)$  are small compared with those

TABLE 4.1  
The stack loss data set.

$i$		$M$		$\mathbf{f}$	
1	1	80	27	89	42
2	1	80	27	88	37
3	1	75	25	90	37
4	1	62	24	87	28
5	1	62	22	87	18
6	1	62	23	87	18
7	1	62	24	93	19
8	1	62	24	93	20
9	1	58	23	87	15
10	1	58	18	80	14
11	1	58	18	89	14
12	1	58	17	88	13
13	1	58	18	82	11
14	1	58	19	93	12
15	1	50	18	89	8
16	1	50	18	86	7
17	1	50	19	72	8
18	1	50	19	79	8
19	1	50	20	80	9
20	1	56	20	82	15
21	1	70	20	91	15

TABLE 4.2

Values of  $F$ ,  $\mathbf{v}$ ,  $\mathbf{u}$  for successive iterations of the reduced gradient algorithm applied to the data given in Table 4.1.

$i$	1	2	3	4	5	6	7	8	9	10
$F$	368.0	138.3	64.86	64.64	1.359	.9891	.9022	.8840	.8043	.7840
$v_1$	0	0	0	0	1	1	1	1	1	1
$v_2$	0	0	-1	-1	.0269	-.01522	-.00494	-.00878	-.00923	-.00845
$v_3$	0	0	0	.0934	.00034	-.02174	-.02612	-.002239	-.00848	-.01079
$v_4$	0	-.1724	.4982	.5201	.00190	0	-.00435	-.00321	-.00541	-.00524
$v_5$	1	1	.9774	.9933	.02688	.02174	.01306	.01592	.01200	.01031
$u_1$	-21.00	.3103	.7641	.7590	-1.370	-1.815	-2.204	-2.013	-1.439	-.4647
$u_2$	-1269.	-89.00	-2.328	4.834	.9542	-2.	-3.891	3.101	-.5001	.8594
$u_3$	-443.0	-40.86	5.760	-3.279	5.667	4.076	3.997	-1.601	1.214	.7616
$u_4$	-1812.	-.3103	-3.524	-3.145	7.434	-.2717	.2001	.3972	.5295	.6277
$u_5$	-368.0	-138.3	64.86	64.64	-1.359	-.9891	-.9022	-.8840	-.8043	-.7840

TABLE 4.3

Solutions produced by starting with different coordinate vectors as initial conditions when the intercept is not constrained. The solutions corresponding to  $\mathbf{e}_3$ ,  $\mathbf{e}_4$  are identical. The permutation of the components of  $\mathbf{u}$  is an implementation artifact.

$\mathbf{v}_0$	$\mathbf{e}_2$	$\mathbf{e}_3$	$\mathbf{e}_4$	$\mathbf{e}_5$
$F$	43.75	24.20	24.20	42.08
$v_1$	-34.83	-20.50	-20.50	39.69
$v_2$	1	.00334	.00334	-.8319
$v_3$	.3013	1	1	-.5739
$v_4$	-.1949	.05071	.05071	.06087
$v_5$	-.8558	-.2900	-.2900	1
$u_1$	-.1464	.5527	.04604	.1899
$u_2$	-.06796	.04604	.4184	-.5580
$u_3$	.4696	.4184	.5527	-.7290
$u_4$	.7447	-.0171	-.0171	.6391
$u_5$	-45.75	-24.20	-24.20	-42.08

in the other columns so that choosing  $\mathbf{v} = \mathbf{e}_j$ ,  $j > 1$ , would be likely to make  $v_1$  large. However, the first column corresponds to an intercept term and is known exactly. If this column is not included ( $\kappa_1(E) = 0$ ) then the results are rather different and several isolated local minima are found by varying the initial conditions so that  $\mathbf{v} = \mathbf{e}_j$ ,  $j = 2, 3, \dots, 5$ . The values of  $F$ ,  $\mathbf{v}$ ,  $\mathbf{u}$  obtained corresponding to each of these initial conditions are given in Table 4.3. It is readily checked that isolated local solutions are obtained. Also the solution with  $v_5 = 1$  gives the solution to the  $l_1$  regression problem.

The calculations were performed on an HP9845B programmed in BASIC.

**Acknowledgments.** The work of the second author was carried out while he was a Visiting Fellow in the Institute of Advanced Studies, Department of Statistics, and was supported by a grant from the Mathematics Research Centre, Australian National University. This visit was assisted by a grant from the Carnegie Trust for the Universities of Scotland.

## REFERENCES

- [1] R. H. BARTELS AND G. H. GOLUB, *The simplex method for linear programming using LU decomposition*, Comm. ACM, 12 (1969), pp. 266–268.
- [2] P. BLOOMFIELD AND W. STEIGER, *Least absolute deviation curve fitting*, this Journal, 1 (1980), pp. 290–301.
- [3] D. I. CLARK AND M. R. OSBORNE, *A descent algorithm for minimizing polyhedral convex functions*, this Journal, 4 (1983), pp. 757–786.
- [4] C. DANIEL AND F. S. WOOD, *Fitting Equations to Data*, John Wiley, New York, 1971.
- [5] G. H. GOLUB AND C. F. VAN LOAN, *An analysis of the total least squares problem*, SIAM J. Numer. Anal., 17 (1980), pp. 883–893.
- [6] J. B. HIRIART-URRUTY, *Tangent cones, generalized gradients and mathematical programming in Banach spaces*, Math. Oper. Res., 4 (1979), pp. 79–97.
- [7] P. A. P. MORAN, *Random processes in economic theory and analysis*, Sankhya, 21 (1959), pp. 99–126.
- [8] G. A. WATSON, *Approximation Theory and Numerical Methods*, John Wiley, Chichester, 1980.
- [9] ———, *The total approximation problem*, in Approximation Theory IV, L. L. Schumaker, ed., to appear.



## A QUASI-QUASI-NEWTON METHOD FOR GENERATING QUASI-CHOLESKI FACTORS\*

JOHN GREENSTADT†

**Abstract.** Previous attempts to derive quasi-Newton update *factors* by variational means have foundered on the nonlinearity (in the update increment  $D$ ) of the quasi-Newton condition. By neglecting the quadratic term in  $D$ , the QN condition is linearized, so that it becomes possible to derive a perspicuous recipe for least-norm factor-updates. However, in order to restore the robustness which is lost by this approximation,  $D$  is replaced by a suitable multiple of itself, based on minimizing a norm of the discrepancy in the QN condition. The resulting update (in two versions) is compared with the BFGS update for a small sample of "standard" functions. The upshot is that the new factor-update which is covariant under affine transformations is comparable in efficiency with the BFGS update, while the noncovariant factor-update is much worse. A table is given, indicating the numbers of gradient and function evaluations required to bring the function values down to the levels shown.

**Key words.** quasi-Newton methods, Choleski factors, unconstrained minimization

**1. Introduction.** In any method of quasi-Newton (QN) type, for the unconstrained minimization of a function  $F(x)$ , a recursive procedure is used to generate successive approximations to the Hessian  $H$  of the function. One of the most important desiderata for these methods is the maintenance of positive-definiteness in the approximants  $\{B_k\}$  when the true Hessian is itself positive-definite. If this property is not maintained, it is very easy for the Newton-step directions generated with the approximate Hessians to fail to be descent directions, and hence for the minimization algorithm to "hang up". It is well known that the DFP and the BFGS formulas do have this property of preserving positive-definiteness [1] (or at least semi-definiteness), which may in large measure account for their efficacy.

It has been of special interest to the author for some time, to find new quasi-Newton updating formulas by means of a variational approach [2], [3], [4]. This method for generating update formulas has resulted in some interesting results: For example, Goldfarb succeeded in deriving the DFP formula this way, and supplied one of the derivations of the BFGS formula [5]. Moreover, the Fletcher-dual form of Powell's Broyden-symmetric formula turned out to be included in the class of updates derived in [2]. (Some attention was also devoted there to the problem of insuring a downhill direction, even when the approximate Hessian is not positive-definite.)

In attempting to find QN methods which did not require the explicit calculation of the gradient  $g$  of  $F$  (either exactly, or by differences), it was found [3] that the lack of positive-definiteness impeded the convergence of the method very seriously, and in many cases caused it to fail altogether. In a revision of this method [4], whose aim principally was to rectify this defect, it was necessary to incorporate a very elaborate (and expensive) recursive procedure to build an approximate Hessian which was certain to be positive definite. This distinctly improved the performance of the method, but at a stiff price.

On another tack, an alteration was made in the way in which  $B$  is usually updated to form  $B^*$ . This is normally done by the *addition* of an increment  $D$ :

$$(1.1) \quad B^* = B + D.$$

An attempt was made to make use of the obvious fact that if, on the other hand,  $B$  is

---

\* Received by the editors February 22, 1983, and in final revised form December 28, 1983.

† IBM Corporation, Palo Alto Scientific Center, Palo Alto, California 94304.

updated by means of a product:

$$(1.2) \quad B^* = (I + D)^T B (I + D)$$

(where  $D$  is now the correction to the unit matrix  $I$ ), then the positive-definiteness of  $B$  implies that of  $B^*$  (provided that  $D$  is real and that  $I + D$  is nonsingular).

The "standard" way of deriving update formulas for symmetric  $D$ 's by variational means starts with the weighted Frobenius norm of  $D$ , viz.,

$$(1.3) \quad \Phi_0 \equiv \frac{1}{2} \text{Tr} \{P^{-1} D P^{-1} D^T\}$$

where  $P^{-1}$  is a symmetric, positive-definite weight matrix, and "Tr" means that the matrix trace is to be evaluated. Various terms incorporating constraints on  $D$  via Lagrange multipliers are then added, so as to form the complete Lagrangian function which is to be made stationary by the suitable choice of  $D$ . In general, we wish to add the so-called quasi-Newton condition on the updated matrix  $B^*$ , namely:

$$(1.4) \quad B^* s = y$$

where  $s$  is the difference  $x^* - x$ , between the new position vector  $x^*$  and the old one  $x$ , and  $y$  is the corresponding difference  $g^* - g$  between the computed gradients. When the correction to  $B$  is additive, the QN condition reduces to

$$(1.5) \quad Ds = y - Bs \equiv r.$$

To this may be adjoined, if  $B$  is to be kept symmetric, a symmetry condition on  $D$ , viz.,  $D^T - D = 0$ . The complete Lagrangian is then

$$(1.6) \quad \Phi = \frac{1}{2} \text{Tr} \{P^{-1} D P^{-1} D^T\} - \lambda^T (Ds - r) - \text{Tr} \{\Lambda (D^T - D)\}.$$

The solution to this variational problem is quite simple and the formula for  $D$  is given in [2]. The reason for this simplicity is almost solely a result of the fact that the Lagrangian function  $\Phi$  is *quadratic* in its arguments  $D$  and the lambdas. When the conditions for a stationary  $\Phi$  are derived by differentiating it with respect to its arguments, the resulting equations are *linear*.

The situation is quite different if the QN condition for the product form is used. This condition (on  $D$ ) takes the form

$$(1.7) \quad D^T B s + B D s + D^T B D s = y - B s \equiv r$$

which is *not* linear in  $D$ . (There is no symmetry condition on  $D$ , because the form of the update automatically preserves the symmetry of  $B$ .)

When  $D$  need not be symmetric, the Frobenius norm can contain two weight functions [6], so that

$$(1.8) \quad \Phi_0 = \frac{1}{2} \text{Tr} \{P^{-1} D Q^{-1} D^T\}$$

where  $Q^{-1}$  is another positive-definite weight matrix. The equation for  $D$  which results then has the form

$$(1.9) \quad P^{-1} D Q^{-1} - 2 B D s \lambda^T = B s \lambda^T + \lambda s^T B,$$

which has a bilinear term in the arguments  $(D, \lambda)$ . It is clear that, if  $P^{-1} \neq B$ , the matrix  $D$  is entangled in the left-hand side and cannot, in general, be solved for. However, even if  $P^{-1} = B$  and it is possible to solve for  $D$  in terms of  $\lambda$ , it turns out that when  $D$  is substituted back into (1.7) for the purpose of solving for  $\lambda$ , the result is a very complicated set of quartic equations for various inner products involving  $\lambda$ . It does

not seem worthwhile to go to such lengths to obtain a minimum-norm product-form update.

However, these efforts at that time were not entirely in vain. Through a lucky mistake, the *correct* result was obtained that the DFP update could be written in product form, and that the *wrongly-derived* formula was the right one. After A. R. Gourlay and K. W. Brodlie were apprised of this, they very rapidly derived the relationship of the product form with most of the best-known updates, including the DFP, the BFGS and the Murtagh-Sargent rank-one. We were independently able to characterize the changes in condition number of the sequence of approximants in terms of the determinants of the update factors. Our efforts were then pooled and published in [7], which also contains the classification of rank-two updates in terms of definiteness characteristics.

**2. Treatment of the binary product form.** More recently, in the course of collaborative discussions on the problem of sparse updates, Prof. John Dennis, of Rice University, proposed the possibility of deriving minimum norm updates for Choleski factors. If the matrix  $B$  is expressed in the form

$$(2.1) \quad B = LL^T$$

(where  $L$  is a lower-triangular matrix), and if the new approximant  $B^*$  is expressed in like manner

$$(2.2) \quad B^* = L^*L^{*T},$$

then the updating of  $B$  is done via a simple additive update of  $L$ , viz.,

$$(2.3) \quad L^* = L + D.$$

The QN condition for the correction  $D$  reduces to

$$(2.4) \quad LD^T s + DL^T s + DD^T s = y - LL^T s \equiv r$$

which is quite similar to (1.7) and is, of course, nonlinear in  $D$ .

Direct updates of Choleski factors have already been developed, but not by a variational approach. When the objection was raised to Dennis' suggestion that the inevitable nonlinear QN condition would present insuperable difficulties, he proposed "linearizing" the QN condition. As he later explained, he meant replacing the single nonlinear QN condition

$$(2.5) \quad L^*L^{*T} s = y$$

by two coupled linear ones. This would be done by defining the variable  $u$  by

$$(2.6a) \quad L^{*T} s \equiv u,$$

in which case (2.5) would become

$$(2.6b) \quad L^* u = y.$$

Of course, this linearization is achieved at the cost of introducing a new variable. In a 1981 paper [8], Dennis and Schnabel described the algorithm based on this decomposition of the QN condition.

Dennis' remark was *misinterpreted*, in that it was thought that he meant simply to neglect the nonlinear term in (2.4) and to take

$$(2.7) \quad LD^T s + DL^T s = r$$

as an *approximate* QN condition. This, of course, greatly simplifies the Lagrangian, which is now quadratic

$$(2.8) \quad \Phi = \frac{1}{2} \text{Tr} \{P^{-1} D Q^{-1} D^T\} - \lambda^T \{L D^T s + D L^T s - r\}.$$

It is very easy to solve the necessary conditions for stationary  $\Phi$  for  $D$  in terms of  $\lambda$ . The result is

$$(2.9) \quad D = P(s\lambda^T + \lambda s^T) L Q.$$

When this result is substituted into (2.7), for the purpose of finding  $\lambda$ , the equation to be solved is

$$(2.10) \quad [(s^T P s) R + (s^T R s) P] \lambda = -[(\lambda^T P s) R + (\lambda^T R s) P] s + r,$$

where  $R$  is defined as  $L Q L^T$ , and is symmetric.

The left-hand bracket contains known quantities only; hence we may denote it by  $Z$  for convenience ( $Z$  is also symmetric), and solve (2.10) for  $\lambda$ . The result is

$$(2.11) \quad \lambda = -Z^{-1} [(s^T P \lambda) R + (s^T R \lambda) P] s + Z^{-1} r,$$

which gives  $\lambda$  in terms of the two inner products involving  $\lambda$ . Therefore, we must first solve for these two quantities, so that we finally obtain  $D$  explicitly. This can be done by premultiplying (2.11) by  $s^T P$  and  $s^T R$  in turn, which yields

$$(2.12a) \quad \begin{aligned} (s^T P \lambda) &= -(s^T P Z^{-1} R s)(s^T P \lambda) - (s^T P Z^{-1} P s)(s^T R \lambda) \\ &\quad + (s^T P Z^{-1} r) \end{aligned}$$

and

$$(2.12b) \quad \begin{aligned} (s^T R \lambda) &= -(s^T R Z^{-1} R s)(s^T P \lambda) - (s^T R Z^{-1} P s)(s^T R \lambda) \\ &\quad + (s^T R Z^{-1} r). \end{aligned}$$

If we denote  $(s^T P Z^{-1} R s)$  by  $\alpha$ ,  $(s^T P Z^{-1} P s)$  by  $\beta$  and  $(s^T R Z^{-1} R s)$  by  $\gamma$ , then we have to solve the system of equations

$$(2.13) \quad \begin{pmatrix} 1 + \alpha & \beta \\ \gamma & 1 + \alpha \end{pmatrix} \begin{pmatrix} (s^T P \lambda) \\ (s^T R \lambda) \end{pmatrix} = \begin{pmatrix} (s^T P Z^{-1} r) \\ (s^T R Z^{-1} r) \end{pmatrix}$$

for  $(s^T P \lambda)$  and  $(s^T R \lambda)$ . We then substitute these into (2.11), thereby obtaining  $\lambda$ , and then substitute the result into (2.9) to get  $D$ . Thus, the linearization of the QN condition has enabled us to obtain an *explicit* solution for  $D$ . Unhappily, this solution is too inefficient to evaluate just to get an *approximate* QN update.

Fortunately, the most interesting special choice for  $P$  and  $Q$  has the effect of greatly simplifying  $Z$ , and hence all the formulas. To motivate this special choice, we shall consider certain general features of this way of generating updates. First, we take note of the fact that the exact Newton formula for the step  $p$ , defined by

$$(2.14) \quad p = -H^{-1} g,$$

where  $H$  is the true Hessian of  $F$ , is *covariant* under all linear (nonsingular) transformations of the independent variables, and also under a scaling transformation of  $F(x)$ . We need not consider the translatory part of these transformations on  $x$ , because we only make use of the difference of two values of  $x$  (to evaluate  $s$ ), so that we need consider only the group of *affine* transformations, i.e., those of the form

$$(2.15a) \quad \tilde{x} = Sx$$

where  $\tilde{x}$  represents the new set of coordinates, and  $S$  is any nonsingular matrix. In the Appendix, where these matters are discussed in more detail, it is shown that (2.15a) defines  $x$  as a *contravariant* vector. The gradient  $g$  transforms via

$$(2.15b) \quad \tilde{g} = S^{-T}g$$

and is thus defined as a *covariant* vector. (We define  $S^{-T}$  to mean  $(S^{-1})^T$ .) The Hessian transforms as follows:

$$(2.15c) \quad \tilde{H} = S^{-T}HS^{-1}$$

and is a covariant tensor of the second rank. If  $B$  is to be an approximation to  $H$  with the same covariance properties, then  $L$ , in (2.1), is permitted to transform according to

$$(2.15d) \quad \tilde{L} = S^{-T}LU$$

where  $U$  is any *orthogonal* matrix. It is clear that  $D$  ought to transform in like manner.

The QN condition itself is covariant under affine transformations. It therefore seems desirable to construct the original Frobenius norm in such a way that it is an invariant of the affine group. We shall show in the Appendix that the resulting necessary conditions for stationarity of the Lagrangian function are then covariant (the multipliers being defined to have the right transformation properties), and the update which satisfies these conditions is also covariant. Experience has already shown that the scaling (meaning multiplication by a factor) of variables and equations can have a decisive effect on the speed and robustness of a quasi-Newton method, and affine covariance can be regarded as the natural generalization of scaling invariance, in the quasi-Newton context. These affine transformations have already been recognized as having some significance, although they are called by the more restricted name of “scaling” transformations in the book by Gill, Murray and Wright [9].

It is a great misfortune that sparsity, which plays such an important role in the efficient solution of very large problems, is not affinely invariant, but is almost totally coordinate-dependent. A sparsity pattern is invariant only under the very restricted class of transformations consisting of row scalings, column scalings and row or column interchanges. Any other transformation of the affine group, outside of this very small subgroup, will spoil the sparsity pattern of the transformed matrix. For this reason, it is to be expected that the update increment  $D$ , generated by formula (2.9), will not retain any sparsity that  $L$  might start with. Hence, since triangularity is itself a form of sparsity, we cannot expect  $D$  to be triangular.

In fact, therefore, since  $L$  is not defined uniquely, but only up to an orthogonal transformation, we can opt for affine invariance of the sequence of updates and, after the minimization is completed, transform the final approximant to triangular form by orthogonal reduction. We can thus regard this “precursor” of the true Choleski factor as a kind of “quasi-Choleski” factor. (We should therefore think of the letter  $L$  as standing for “left”, instead of “lower”).

Because of the unusual transformation (2.15d) associated with  $L$ , we must pay special attention to the transformation properties of  $P$  and  $Q$ . Since these matrices are both symmetric, it is reasonable to assume that

$$(2.16a) \quad \tilde{P} = X^T P X,$$

$$(2.16b) \quad \tilde{Q} = Y^T Q Y,$$

where  $X$  and  $Y$  are transformation matrices to be determined. The trace used for the

norm should be formally invariant, which means that

$$\begin{aligned}
 \text{Tr} \{ \tilde{P}^{-1} \tilde{D} \tilde{Q}^{-1} \tilde{D}^T \} &= \text{Tr} \{ X^{-1} P^{-1} X^{-T} S^{-T} D U Y^{-1} Q^{-1} Y^{-T} U^T D^T S^{-1} \} \\
 (2.17) \qquad \qquad \qquad &= \text{Tr} \{ P^{-1} (XS)^{-T} D (UY^{-1}) Q^{-1} (UY^{-1})^T D^T (XS)^{-1} \} \\
 &= \text{Tr} \{ P^{-1} D Q^{-1} D^T \}.
 \end{aligned}$$

For (2.17) to be an identity, it is proved in the Appendix that the relations

$$(2.18) \qquad \qquad \qquad XS = \mu I \quad \text{and} \quad UY^{-1} = \pm \mu I$$

must hold, where  $\mu$  is an arbitrary constant. The factors  $\mu$  and  $\pm\mu$  may be absorbed into  $S$  and  $U$  respectively, so we may assume them both to be unity.

The relations (2.18) are clearly sufficient to guarantee (2.17) for all  $D$ ,  $P$  and  $Q$ , so that

$$(2.19a) \qquad \qquad \qquad \tilde{P} = S^{-T} P S^{-1}$$

and

$$(2.19b) \qquad \qquad \qquad \tilde{Q} = U^T Q U.$$

This means that  $P$  transforms like  $H$  (or  $LL^T$ ), and  $Q$  transforms according to the unitary subgroup of the full linear group. It is not clear what choice ought to be made for  $Q$ , but on the principle that only the quantities that occur naturally in the algorithm should appear in the update formulas, it would seem that  $Q = I$ , the identity matrix (which is “present” in any matrix context), is the only sensible choice. By the same token, a natural choice for  $P$  is  $LL^T$  which, in fact, yields the so-called “proportional” update. The Frobenius norm in this case has the form:  $\text{Tr} \{ (LL^T)^{-1} (DD^T) \}$ .

Looking back now at the form of  $Z$  that appears in (2.10), it is clear that requiring  $R$  to be equal (or at least proportional) to  $P$  results in a great simplification. This amounts to setting  $P = \mu L Q L^T$ . Equations (2.10) to (2.13) then simplify greatly, so that the formula for  $D$  becomes

$$(2.20) \qquad \qquad \qquad D = \frac{1}{2\rho} \left( (Psr^T + rs^T P) - \left( \frac{\sigma}{\rho} \right) P s s^T P \right) L^{-T}$$

where  $\rho \equiv s^T P s$  and  $\sigma \equiv s^T r$ .

This simplification adds to the already strong motivation for setting  $Q = I$  and  $P = LL^T$ . For the sake of comparison, we shall show, in § 4, results based also on the other “obvious” choice:  $P = Q = I$ ; this choice is interesting because the Frobenius norm then becomes just the Euclidean norm  $\text{Tr} (DD^T)$ , which corresponds to an “absolute” update, as compared with a “proportional” one.

**3. The last “quasi”.** The linearization of the QN condition yields a great benefit; viz., it makes feasible a simple, closed formula for the quasi-Choleski factor update. Unfortunately, it also exacts a stiff price. At the start of a minimization, when one is far from the solution, the magnitude of  $D$  (in any norm) may easily be far larger than that of  $L$ . Preliminary trials, using simple quadratic functions, gave very disappointing results; the algorithm almost never converged, but instead got “stuck” far from the solution.

The difficulty, of course, is that the QN condition is too far from being satisfied. Clearly, we cannot satisfy it exactly, because that would involve applying precisely the nonlinear exact condition that leads to intractable problems in solving for  $D$ . However, it is still possible, using the approximate solution for  $D$  derived in the

preceding section, to satisfy the exact QN condition “as closely as possible”. In mathematical terms, this means minimizing some norm of a suitably defined “discrepancy” in the QN condition.

Our strategy is precisely that used in the original minimization problem itself. The Newton “step”  $D$  is interpreted, not as a step, but as a search direction, and a parameter is introduced which measures the actual extent of the difference between the starting point of the search and the current point being examined. Insofar as the Newton step is derived by linearizing a nonlinear problem, and solving the resulting approximate problem exactly, the value of  $D$  may be regarded as a “Newton direction” in the  $N^2$ -dimensional space of  $N \times N$  matrices.

We can then consider the family of new factors  $L^*(\tau) \equiv L + \tau D$ , where  $\tau$  is the running parameter along the line thus defined. The exact QN condition is

$$(3.1) \quad L^*(\tau)L^{*T}(\tau)s = (L + \tau D)(L + \tau D)^T s = y,$$

which can never in general be satisfied by the  $D$  calculated in § 2, for any value of  $\tau$ . However, we can make the discrepancy  $\Delta$ , defined by

$$(3.2) \quad \Delta \equiv L^*L^{*T}s - y = (LD^T + DL^T)s\tau + (DD^T)s\tau^2 - r$$

as small as possible, in the sense that we can minimize some norm of  $\Delta$  by the appropriate choice of  $\tau$ .

The simplest norm to use for  $\Delta$  is a weighted Euclidean norm of the form

$$(3.3) \quad \Psi = \|\Delta\|^2 \equiv \Delta^T W \Delta.$$

For  $W = I$ ,  $\Psi$  is the Euclidean norm, and for  $W = (LL^T)^{-1}$ , it is the affinely invariant norm. If we substitute for  $\Delta$  from (3.2), the result is a quartic in  $\tau$ :

$$(3.4) \quad \Psi = c_1 + c_2\tau + c_3\tau^2 + c_4\tau^3 + c_5\tau^4.$$

For compactness, we define

$$(3.5) \quad \begin{aligned} u &\equiv (LD^T + DL^T)s, \\ v &\equiv DD^T s, \end{aligned}$$

so that the  $c$ 's can be written

$$(3.6) \quad \begin{aligned} c_1 &= r^T W r, \\ c_2 &= -2r^T W u, \\ c_3 &= -2r^T W v + u^T W u, \\ c_4 &= 2u^T W v, \\ c_5 &= v^T W v. \end{aligned}$$

The solution of this subproblem was done by finding a root of the equation

$$(3.7) \quad \frac{d\Psi}{d\tau} = 0$$

using Newton's method. Just starting with unity for  $\tau$ , and performing the raw Newton iteration (i.e., without line-searching or other protective devices) always gave the correct value for  $\tau$ , thereby obviating the usually mandatory line search. A programmed check of the progress of  $\|\Delta\|$  disclosed no cases when it did not decrease monotonically through the  $\tau$ -iterations. The ratio of final to initial norms varied from  $10^{-7}$  to unity,

indicating that at times,  $\Delta$  was being grossly overestimated, while at other times, it was very accurate (the latter usually occurred near the minimum of  $F$ ).

The complete "linearized" quasi-Newton, quasi-Choleski factor update algorithm, with a "best fit" to the QN condition, worked amazingly well. In the next section, the results of various numerical trials will be presented, showing the performance of this algorithm on a few representative functions.

**4. Numerical results.** A most important feature of any minimization algorithm is the line-search which it uses. We have used a rather simple one, which proceeds as follows:

- (1) Starting with the initial point  $x_0$  in each step, for which  $F_0$  and  $g_0$  are known, we denote the function  $F(x_0 + \lambda p)$  by  $f(\lambda)$  (where  $\lambda$  is a parameter along  $p$ ), and the directional derivative  $df/d\lambda$  at  $\lambda = 0$  by  $\dot{f}_0$ , which has the value  $g^T p$ .
- (2) We next evaluate  $f_1 \equiv f(\lambda_1)$  (where  $\lambda_1$  is initially set to unity), and combining it with the two pieces of information in step 1, we construct the second order approximation to  $f(\lambda)$  which matches  $f_0$ ,  $\dot{f}_0$  and  $f_1$ . This parabola has a minimum or maximum at  $\lambda_m$ , where its value is  $f_m \equiv f(\lambda_m)$ . We compute the ratio  $\beta \equiv f_1 \div f_m$ , which measures the extent to which  $f_1$  is near the estimated minimum of  $f(\lambda)$ . If  $f(\lambda)$  were exactly quadratic,  $\beta$  would be equal to unity.
- (3) If  $\beta$  falls above a predetermined lower bound, the value of  $\lambda_m$  is regarded as satisfactory, and the step based on it is accepted. (In practice, the value of .1 for this lower bound was found to work well).
- (4) If  $\beta$  is too small (including negative), and if  $\lambda_m$  lies between 0 and  $\lambda_1$ , then  $\lambda_1$  is set equal to  $\lambda_m$ ,  $f_1$  is set equal to  $f_m$ , and step (2) is repeated.
- (5) If  $\lambda_m$  is outside of this range, a "trap" sequence is initiated, which consists of an extrapolation of  $f(\lambda)$  until the middle value of three successive values is smaller than the other two. The line search is regarded as finished at this point.

With this search (which, be it noted, does not ensure "average positive curvature"), five functions were minimized, using the BFGS (with the inner product  $y^T s$  checked for positivity at every step), the "absolute" quasi-Choleski (QCHOL1) and the affinely invariant quasi-Choleski (QCHOL2) updating formulas. The weight  $W$ , used for finding  $\Delta$  in (3.3), was chosen to correspond to the type of update used; i.e.  $W = I$  for QCHOL1, and  $W = (LL^T)^{-1}$  for QCHOL2. The five functions tested were:

(a) A quadratic function in 5 variables, for which the Choleski factor of its (constant) Hessian matrix was generated randomly. The linear and constant parts were constructed so as to make (1, 1, 1, 1, 1) the minimum point, and to make the function vanish at that point. The starting value of  $x$  is denoted by  $x_0$ . The function is given by

$$F(x) = (Ax - b)^T (Ax - b),$$

$$A = \begin{bmatrix} 60 & 5 & 16 & -8 & 17 \\ & 62 & -72 & -10 & -53 \\ & & 82 & -43 & -48 \\ & & & 78 & -64 \\ & & & & 72 \end{bmatrix},$$
(3.8)

$$b = \{90, -73, -9, 14, 72\},$$

$$x_0 = \{1, 2, 3, 4, 5\}.$$



(b) Rosenbrock's function [11]. The standard starting value was used, as it was for all the rest of the "standard" functions tested. The function is

$$(3.9) \quad \begin{aligned} F(x) &= 100 \times (x_1^2 - x_2)^2 + (1 - x_1)^2, \\ x_0 &= \{-1.2, 1\}. \end{aligned}$$

(c) The "banana function" [12] in 10 variables.

$$(3.10) \quad \begin{aligned} F(x) &= \sum_{i=1}^9 (100 \times (x_{i+1} - x_i^2)^2 + (1 - x_i)^2), \\ x_0 &= \{-1.2, 1, -1.2, 1, -1.2, 1, -1.2, 1, -1.2, 1\}. \end{aligned}$$

(d) Powell's quartic function [11].

$$(3.11) \quad \begin{aligned} F(x) &= (x_1 + 10x_2)^2 + 5(x_3 - x_4)^2 + (x_2 - 2x_3)^4 + 10(x_1 - x_4)^4, \\ x_0 &= \{3, -1, 0, 1\}. \end{aligned}$$

(e) Wood's function [11].

$$(3.12) \quad \begin{aligned} F(x) &= 100(x_2 - x_1^2)^2 + (1 - x_1)^2 + 90(x_4 - x_3^2)^2 + (1 - x_3)^2 \\ &+ 10.1[(x_2 - 1)^2 + (x_4 - 1)^2] + 19.8(x_2 - 1)(x_4 - 1), \\ x_0 &= \{-3, -1, -3, -1\}. \end{aligned}$$

All of these functions vanish at the solution points. With double-precision used throughout (all the calculations were done in APL), the final value of  $F$  can be driven down to approximately the square of the machine precision (about  $10^{-32}$ ). However, it is felt by some workers in this field that, since many real problems are not formulated as sums of squares, it is generally not possible to drive the value of  $F$  below the basic machine precision (in our case,  $10^{-16}$ ). The two chief termination criteria used were (1) the increase of the directional derivative  $f_0$  to an insufficiently negative value and (2) the shrinking of the value of  $F$  to the point where the difference between  $f_1$  and  $f_0$  became subject to excessive relative rounding error. Both versions of the latter termination criterion were handled by evaluating the difference as  $(R + f_1) - (R + f_0)$ , where  $R$  was given the value 0 for the "double-precision" run and the value 1 for the "single-precision" one.

Table 1 shows the results of these runs. GS and FS are the numbers of gradient and of function evaluations at termination, and the minimum values achieved are listed.

**5. Discussion.** While no effort has been made to give anything like an exhaustive demonstration of the performance of these update formulas, an important inference can be drawn even from the small sample of tests shown here. It is obvious that the quasi-Choleski update which is not affinely invariant gives a very poor performance, while the affinely invariant one is comparable with the BFGS update (which is currently regarded as the "best" one). It would therefore appear that closer attention should be directed to wider invariance properties of quasi-Newton methods than just scaling invariance. It is remarkable that a unique (and efficient) update can be derived based on a variational principle and a strict requirement of affine invariance. It may be that the disappointing performance of a priori sparse quasi-Newton updates is due to their meager invariance properties.

While our tests have shown, once again, that the BFGS update is supreme (it also has a sparse form [13]), one of our original motivations was to broaden the range of choices for sparse quasi-Newton updates beyond those derivable from known nonsparse

TABLE I

Minimization results for three update formulas and five functions. The numbers of gradient and of function evaluations are given, as are the final values of the functions. The numbers in parentheses indicate powers of 10. Results for both double and single precision terminations are shown.

Results with double-precision minimum									
UD type →	BFGS			QCHOL1			QCHOL2		
Function ↓	GS	FS	Minimum	GS	FS	Minimum	GS	FS	Minimum
Quad 5	11	12	0.0	24	24	9.2 (-28)	20	24	1.2 (-26)
Rosenbrock	39	48	4.1 (-26)	44	57	2.1 (-29)	36	45	4.9 (-31)
Banana 10	75	92	7.3 (-27)	130	211	3.6 (-27)	81	116	1.8 (-27)
Powell	85	97	5.6 (-26)	261	328	2.7 (-23)	68	84	6.9 (-24)
Wood	85	117	1.0 (-28)	147	164	2.2 (-26)	90	152	4.7 (-29)

Results with single-precision minimum									
Function ↓	GS	FS	Minimum	GS	FS	Minimum	GS	FS	Minimum
Quad 5	10	12	2.3 (-22)	21	22	3.2 (-18)	18	23	5.3 (-17)
Rosenbrock	38	48	1.1 (-20)	39	53	8.2 (-17)	34	44	1.8 (-18)
Banana 10	69	87	1.1 (-16)	112	184	4.3 (-16)	76	112	1.1 (-16)
Powell	46	54	9.4 (-15)	180	215	8.9 (-14)	48	65	1.4 (-16)
Wood	82	115	1.5 (-16)	136	154	7.2 (-18)	87	150	3.5 (-19)

ones, and also to try to extend the results of Toint [14] on variationally derived updates. Since Toint relied on a unit weight matrix in the Frobenius norm, it was our hope to be able improve the efficiency of his update by extending the set of transformations under which that norm was invariant, subject to the sparsity constraints.

To some extent, we were successful, and have programmed a sparse updating scheme which, while still incomplete in some formal respects, is able to generate updates with prespecified sparsity, which are still covariant with respect to row and column scaling only. Moreover, it has been possible to extend this procedure, so that it can generate *true* Choleski factor updates, simply by prespecifying a triangular sparsity pattern for the update. Unfortunately, apart from the formal incompleteness of the technique (which, however, gives correct results), the updates are relatively inefficient, when compared with (nonsparse) affinely covariant ones.

Of course, if one starts with a triangular initial approximation for  $L$  (such as the unit matrix), one can always restore the triangularity of each update approximation by the methods of Gill, Golub, Murray and Saunders. The best such method [15] requires  $N^2$  multiplications for adding a rank-one correction, and  $\frac{3}{2}N^2$  for subtracting one. Since the quasi-Choleski update is of this type, it requires  $\frac{5}{2}N^2$  multiplications to retriangularize  $L$ .

6. Appendix.

**Demonstration of covariance properties.** We shall first show that in order that the Frobenius norm  $\Phi_0$  in (1.3) be invariant under transformation of its tensor components  $D, P$  and  $Q$  (as shown in (2.17)),  $P$  and  $Q$  must transform according to (2.16) and (2.18).

Since (2.17) is to be an identity for *any*  $D, P$  and  $Q$  (with  $P$  and  $Q$  symmetric), we may differentiate it with respect to  $D$ , and still expect an identity. We apply the general rules (which may be readily verified in terms of components)

$$(A1) \quad \frac{\partial}{\partial D} \text{Tr} (DR) = R^T, \quad \frac{\partial}{\partial D^T} \text{Tr} (D^T R) = R,$$

to the last two lines of (2.17). If we denote  $XS$  by  $A$ , and  $UY^{-1}$  by  $B$ , the result simplifies to

$$(A2) \quad A^{-1}P^{-1}A^{-T}DBQ^{-1}B^T = P^{-1}DQ^{-1}.$$

Next, we premultiply by  $A^T P A$  and postmultiply by  $Q$  to obtain

$$(A3) \quad A^T P A P^{-1} D = D B Q^{-1} B^T Q.$$

Recalling again that (A3) has to hold for any  $D$ , we simply set  $D$  to  $I$ , the unit matrix, to obtain

$$(A4) \quad A^T P A P^{-1} = B Q^{-1} B^T Q,$$

and we shall denote the common value of these two matrices by  $R$ . We are now ready to apply the second part of Schur's celebrated lemma, which we quote from [10] (with our own variable names):

*If a matrix  $R$  commutes with all the matrices of an irreducible system  $\{D\}$ , then  $R$  is a numerical multiple of the unit matrix.*

Inasmuch as the system  $\{D\}$  consists of the set of all nonsingular matrices, it is certainly an irreducible system. Therefore, we are entitled to assert that  $R = \mu I$ . We thus have

$$(A5) \quad A^T P A P^{-1} = \mu I, \quad B Q^{-1} B^T Q = \mu I.$$

Recalling again that  $P$  and  $Q$  are arbitrary symmetric matrices, we may set each of them to the unit matrix, to obtain

$$(A6) \quad A^T A = \mu I, \quad B B^T = \mu I,$$

so that both  $A$  and  $B$  are quasi-unitary; we may make them both unitary by absorbing  $\sqrt{\mu}$  into each of them. We then have  $A^T A = I$  and  $B B^T = I$ . This means that we can regard  $\mu$  as equal to unity in (A5).

If we therefore premultiply the first equation in (A5) by  $P^{-1} A$ , and postmultiply the second one by  $Q^{-1} B$ , the result is

$$(A7) \quad A P^{-1} = P^{-1} A, \quad B Q^{-1} = Q^{-1} B.$$

Recalling yet again that  $P$  and  $Q$  are arbitrary symmetric nonsingular matrices, and hence form an irreducible system, we can apply Schur's lemma once more, to obtain

$$(A8) \quad A = \mu I, \quad B = \nu I.$$

Referring back to the last two lines of (2.17), we replace  $XS (\equiv A)$  by  $\mu I$  and  $UY^{-1} (\equiv B)$  by  $\nu I$ , and we have

$$(A9) \quad \begin{aligned} \text{Tr} \{ P^{-1} \mu^{-1} D \nu Q^{-1} \nu D^T \mu^{-1} \} &= \left( \frac{\nu}{\mu} \right)^2 \text{Tr} \{ P^{-1} D Q^{-1} D^T \} \\ &= \text{Tr} \{ P^{-1} D Q^{-1} D^T \} \end{aligned}$$

so that  $\nu$  must be equal to  $\pm \mu$ . Hence finally,

$$(A10) \quad A \equiv XS = \mu I, \quad B \equiv UY^{-1} = \pm \mu I,$$

which is the same as (2.18).

We shall now show that an update increment derived from an invariant Lagrangian function has the correct covariance properties. We first consider the way in which the

quasi-Newton condition itself transforms. Let us consider the coordinate transformation

$$(A11) \quad \tilde{x}^i = \sum_j S_j^i x^j$$

which was written as  $\tilde{x} = Sx$  in (2.15a) and is defined in tensor analysis [16] as a *contravariant* transformation of the vector  $x$ . (The corresponding inverse is, of course,  $x = S^{-1}\tilde{x}$ .) From this transformation, we may derive

$$(A12) \quad \frac{\partial \tilde{x}^i}{\partial x^j} = S_j^i,$$

which is written in matrix form as  $\partial \tilde{x} / \partial x = S$ , etc. The function  $F(x)$  is assumed to be invariant in value under this transformation, i.e.,

$$(A13) \quad \tilde{F}(\tilde{x}) = F(x).$$

When we differentiate this relation with respect to  $x^i$ , we obtain

$$(A14) \quad \frac{\partial F}{\partial x^i} = \sum_j \frac{\partial \tilde{x}^j}{\partial x^i} \frac{\partial \tilde{F}}{\partial \tilde{x}^j} = \sum_j S_j^i \frac{\partial \tilde{F}}{\partial \tilde{x}^j},$$

so that, if the gradient of  $F$  is denoted by  $g$ , (A14) can be written in matrix form as  $g = S^T \tilde{g}$ . This kind of transformation is called *covariant*, and  $g$  is called a covariant vector. Obviously, we have  $\tilde{g} = S^{-T}g$ , which is the same as (2.15b).

In a similar manner, we can derive (2.15c), which shows the Hessian  $H$  to be covariant tensor of the second rank. In keeping with the point of view expressed in this paper, we wish the approximants  $B$  and  $B^*$  to have the same covariance properties as  $H$ , so we require

$$(A15) \quad \tilde{B} = S^{-T}BS^{-1}, \quad \tilde{B}^* = S^{-T}B^*S^{-1}.$$

Moreover, since  $B = LL^T$ , it is obvious that the most general transformation for  $L$  is that shown in (2.15d). This is rather an unusual transformation, which does not exactly fit into the classical framework of tensor analysis, but is certainly meaningful from a group-theoretic point of view. In any case, if  $L$  (and hence  $D$ ) transform in this way,  $B$  will transform correctly (as will  $B^*$ , etc). We need only note further that  $s$ , being the difference of  $x^*$  and  $x$ , transforms in the same way as  $x$ , and that  $y$ , being the difference of  $g^*$  and  $g$ , transforms like  $g$ . The "quasi-Newton discrepancy"  $\Delta$  is then found to transform as follows,

$$(A16) \quad \tilde{\Delta} \equiv \tilde{B}^* \tilde{s} - \tilde{y} = S^{-T}B^*S^{-1}Ss - S^{-T}y = S^{-T}(B^*s - y) = S^{-T}\Delta,$$

so that  $\Delta$  is also a covariant vector.

In order that the expression that we add to the invariant Frobenius norm also be invariant, so that the entire Lagrangian function in (2.8) be invariant, the multiplier  $\lambda$  must transform properly. Since  $\lambda$  is a vector, the most general transformation for it is:  $\tilde{\lambda} = A\lambda$ , where  $A$  is some matrix. If  $\lambda^T \Delta$  is to be invariant, we must have

$$(A17) \quad \tilde{\lambda}^T \tilde{\Delta} = \lambda^T A^T S^{-T} \Delta = \lambda^T \Delta$$

so that clearly,  $A = S$ , and  $\tilde{\lambda} = S\lambda$ , showing it to be a contravariant vector.

Let us now assume nothing about the transformation properties of  $D$ , but derive them from the variational formula (2.9), using only the transformations on  $P$ ,  $Q$ ,  $L$ ,  $s$

and  $\lambda$  which were necessary to make  $\Phi$  invariant. We have

$$\begin{aligned}
 \tilde{D} &= \tilde{P}(\tilde{s}\tilde{\lambda}^T + \tilde{\lambda}\tilde{s}^T)\tilde{L}\tilde{Q} \\
 &= S^{-T}PS^{-1}(Ss\lambda^T S^T + S\lambda s^T S^T)S^{-T}LUU^TQU \\
 (A18) \quad &= S^{-T}P(s\lambda^T + \lambda s^T)LQU \\
 &= S^{-T}DU,
 \end{aligned}$$

which shows that  $D$  transforms like  $L$ , so that their sum is consistent.

This result, albeit quite simple, illustrates the more general result that the solution to the equations resulting from the variation of an invariant functional has the property of being covariant.

**Acknowledgments.** My thanks are due to Prof. John Dennis for prompting this investigation, to Drs. P. Gill, W. Murray and M. Saunders for suggestions as to testing procedures and references, and to the Referees for many fascinating and extremely helpful comments, criticisms and corrections.

#### REFERENCES

- [1] J. E. DENNIS AND J. J. MORÉ, *Quasi-Newton methods, motivation and theory*, SIAM Rev., 19 (1977), pp. 46–89.
- [2] J. GREENSTADT, *Variations on variable-metric methods*, Math. Comp., 24 (1970), pp. 1–22.
- [3] ———, *A quasi-Newton method with no derivatives*, Math. Comp., 26 (1972), pp. 145–166.
- [4] ———, *Revision of a derivative-free quasi-Newton method*, Math. Comp., 32 (1978), pp. 201–221.
- [5] D. GOLDFARB, *A family of variable-metric methods derived by variational means*, Math. Comp., 24 (1970), pp. 23–26.
- [6] E. SPEDICATO AND J. GREENSTADT, *On some classes of variationally derived quasi-Newton methods for systems of nonlinear equations*, Numer. Math., 29 (1978), pp. 363–380.
- [7] K. W. BRODLIE, A. R. GOURLAY AND J. GREENSTADT, *Rank-one and rank-two corrections to positive-definite matrices expressed in product form*, J. Inst. Math. Applics., 11 (1973) pp. 73–82.
- [8] J.E. DENNIS AND R. B. SCHNABEL, *A new derivation of symmetric positive definite secant updates*, in Nonlinear Programming 4, O. L. Mangasarian et al., eds., Academic Press, New York, 1981.
- [9] P. E. GILL, W. MURRAY AND M. H. WRIGHT, *Practical Optimization*, Academic Press, New York, 1981.
- [10] H. BOERNER, *Representations of Groups*, North-Holland, Amsterdam, 1963.
- [11] P. E. GILL, W. MURRAY AND R. A. PITFIELD, *The implementation of two revised quasi-Newton algorithms for unconstrained optimization*, Report #NAC 11, National Physical Laboratory, Teddington, England, 1972.
- [12] S. S. OREN, *Self-scaling variable metric algorithms for unconstrained optimization*, Thesis, Dept. Engineering-Economic Systems, Stanford Univ., Stanford, CA, 1972.
- [13] J. E. DENNIS, JR, AND PHUONG VU, *Toward direct sparse updates of Choleski factors*, Rice Univ. Tech. Report 83-13, Houston, TX, April 1983.
- [14] PH. L. TOINT, *On sparse and symmetric matrix updating subject to a linear equation*, Math. Comp., 31 (1977), pp. 954–961.
- [15] P. E. GILL, W. MURRAY AND M. A. SAUNDERS, *Methods for computing and modifying the LDV factors of a matrix*, Math. Comp., 29 (1975), pp. 1051–1077.
- [16] J. L. SYNGE AND A. SCHILD, *Tensor Calculus*, Univ. Toronto Press, Toronto, 1949.

## ITERATIVE METHODS FOR SOLVING BORDERED SYSTEMS WITH APPLICATIONS TO CONTINUATION METHODS\*

TONY F. CHAN† AND YUCEF SAAD†

**Abstract.** Consider the linear system

$$M = \begin{bmatrix} A & b \\ c^T & d \end{bmatrix},$$

where  $A$  may be nearly singular but the vectors  $b$  and  $c$  are such that  $M$  is nonsingular. If  $A$  is large and sparse, use of iterative methods seems attractive. In this paper, a number of conjugate gradient like methods are considered. Estimates of eigenvalues of  $M$  based on those of  $A$  are derived. A primary issue is the exploitation of special properties of  $A$ , e.g. symmetry, in these algorithms. Often, a good preconditioning for  $A$  is available and we show various ways of exploiting it. Results of numerical experiments derived from the use of a continuation method for solving a nonlinear elliptic problem are presented and some general conclusions concerning the relative performance of these algorithms are drawn.

**Key words.** continuation methods, bordered systems, Krylov subspaces, iterative methods, preconditioning, constrained optimization

**1. Introduction.** Consider the linear system

$$(1) \quad M \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} A & b \\ c^T & d \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} f \\ g \end{bmatrix},$$

or

$$Mz = p,$$

where the  $n$  by  $n$  matrix  $A$  is bordered by the vectors  $b$  and  $c$  to form a larger system of dimension  $(n+1)$  by  $(n+1)$ .<sup>1</sup> In particular, we are primarily interested in the use of conjugate gradient type iterative methods for solving (1) which may be appropriate when the matrix  $A$  (and consequently  $M$ ) is large and sparse as for example when  $A$  is derived from discretizations of multi-dimensional partial differential equations. We shall present algorithms for solving systems of this form and perform numerical experiments to compare their relative efficiency.

Systems of this form arise in many applications. The matrix  $A$  often represents the regular part of the problem, whereas the vectors  $b$  and  $c$  represent either a constraint on the solution  $x$  or a coupling among the variables  $x$  and  $y$ . For example, these matrices are common in Lagrangian methods for constrained optimization [14]. We are primarily interested here in the class of path-following continuation methods for solving *nonlinear* problems. These include continuation procedures for solving nonlinear eigenvalue problems of the form  $G(u, \lambda) = 0$ , where  $u$  represents the usual "solution" and  $\lambda$  is a real parameter intrinsic to the problem [8], [18], [25]. Usually, one is interested in tracing the solution curves  $[u(\lambda), \lambda]$ . However, due to the fact that these solution curves may possess multiple solutions for a fixed value of  $\lambda$  and that they may admit singular points such as turning points and bifurcation points, it may not be best, or even possible, to parameterize the solution curves by the naturally

---

\* Received by the editors May 26, 1982, and in revised form November 28, 1983. This research was supported in part by the U.S. Department of Energy under contract DE-ACO2-81ER10996.

† Computer Science Department, Yale University, New Haven, Connecticut 06520.

<sup>1</sup> In general,  $b$  and  $c$  could be rectangular matrices although in this paper we shall only be concerned with the vector case. All the algorithms presented can be easily generalized to the higher dimensional cases.

occurring parameter  $\lambda$ . Often it is better to parameterize these solution curves either by another independent variable in  $u$  or by an arclength parameter. In these cases, the matrix  $A$  represents the Jacobian  $G_u$ , the vector  $b$  represents  $G_\lambda$ , and the last equation in (1) represents the "arclength constraint" on  $[u, \lambda]$ . For example, with the pseudo arclength method [18],  $(c^T, d)$  is a scalar multiple of the unit tangent  $(\dot{u}^T, \dot{\lambda})$  at a solution  $(u^T, \lambda)$ .

Another related application area is the class of homotopy continuation methods designed to improve the global convergence of iterative algorithms (e.g. Newton's method) for solving general nonlinear systems and fixed-point problems [13]. In these homotopy techniques, one transforms a nonlinear system  $F(x) = 0$  by a homotopy into a larger system, for example,  $H(x, t) = (1-t)(x-x_0) + tF(x) = 0$ . Note that  $H(x_0, 0) = 0$  and  $H(x, 1) = F(x)$ . Thus, one can start from the known solution  $x_0$  at  $t=0$  and trace the solution curve of  $H(x, t)$  until the solution of  $H(x, t) = 0$  at  $t=1$  is reached, which by construction is a solution of  $F(x) = 0$ .

The matrix  $M$  is partitioned in the way exhibited in (1) because in many applications, such as the ones mentioned above, the matrix  $A$  possesses special structures which one would like to exploit. Two such structures that we consider here are sparseness and symmetry. In the use of direct methods, the sparseness of  $A$  can be exploited in a variety of ways. Rheinboldt [26] has suggested a technique suitable for banded  $A$  and Keller [18] has employed a block elimination algorithm (see § 3.1). On the other hand, in addition to sparseness, the symmetry of the coefficient matrix often plays a critical role in both the efficiency and the convergence of iterative methods [15]. For example, for conjugate gradient type methods ([7], [12]), efficient methods and rather complete theories exist for symmetric problems, whereas the situation for nonsymmetric problems are not as well-understood. In many applications, although  $A$  is symmetric, the vectors  $b$  and  $c$  in (1) are unequal in general, resulting in a matrix  $M$  that is nonsymmetric. An obvious approach for solving (1) is to apply a nonsymmetric iterative method directly, without explicitly taking advantage of the symmetry of  $A$ . In this paper, we also consider algorithms for solving (1) that do exploit the symmetry of  $A$ . However, all of these algorithms require solving two symmetric systems for each system of the form (1). One of the main issues that we would like to address in this paper is the obvious trade-offs among these different approaches. A related issue is the construction of effective preconditioning techniques to be used with these algorithms, assuming that a preconditioning matrix is available for the matrix  $A$ . We note that symmetric conjugate gradient methods (without preconditionings) have been used by Abbott [1] and Mittelmann [21].

Another property of the matrix  $A$  that plays an important role in our discussion is its indefiniteness. In fact, in the path following applications mentioned above,  $A$  can actually become singular near the singular points. However, the vectors  $b$  and  $c$  are constructed so that the matrix  $M$  remains nonsingular. Since the convergence of conjugate gradient type methods depends critically on the distribution of the eigenvalues of the coefficient matrix, for example, positive definiteness and the spread of the eigenvalues, it is important to understand the relationship of the eigenvalues of  $M$  to those of  $A$ . In § 2, we shall present some general results in this direction and apply them to matrices arising from the use of continuation methods on nonlinear elliptic problems. In § 3, we shall present three algorithms for solving systems of the form (1), two of which exploit the symmetry of  $A$ . Preconditioning techniques will be discussed in § 4. Numerical experiments were carried out which apply these algorithms to a pseudo-arclength continuation procedure ([8], [18]) for tracing the solution curve of a two dimensional nonlinear elliptic eigenvalue problem that possesses a turning point.

The numerical results will be presented in § 5 and we attempt to draw some general conclusions from these results in § 6.

**2. Estimates of eigenvalues for bordered matrices.** In order to be able to interpret the behaviour of various iterative methods, in this section, we study the eigenvalues of matrices of the form

$$(2) \quad M = \begin{bmatrix} A & b \\ c^T & d \end{bmatrix}.$$

We assume that  $A$  is symmetric.

**2.1. General case.** Let us start by block factoring the matrix (2) as follows

$$M = \begin{bmatrix} A & b \\ c^T & d \end{bmatrix} = \begin{bmatrix} I & 0 \\ q^T & 1 \end{bmatrix} \begin{bmatrix} A & b \\ 0 & y \end{bmatrix}$$

with

$$(3) \quad q = A^{-1}c,$$

$$(4) \quad y = d - q^T b = d - c^T A^{-1}b.$$

Let the eigenvalues of  $A$  be  $\theta_i, i = 1, \dots, n$ , where we assume that  $\theta_1 \leq \theta_2 \leq \dots \leq \theta_n$ . We will denote by  $\mu_i$  the eigenvalues of  $M$ , and label them according to increasing real parts.

Since  $A$  is symmetric we can write  $A = Q^T \Lambda Q$ , where  $Q$  is orthonormal and  $\Lambda = \text{diag}(\theta_1, \theta_2, \dots, \theta_n)$ . We will denote by  $\beta_i$  (resp.  $\delta_i$ ) the  $i$ th component of the vector  $Q^T b$  (resp.  $Q^T c$ ). From the above factorization applied to  $M - \mu I$ , we get for all  $\mu$  not in the spectrum of  $A$

$$(5) \quad \text{Det}(M - \mu I) = \text{Det}(A - \mu I) \cdot \left[ d - \mu - \sum_{i=1}^n \frac{\beta_i \delta_i}{\theta_i - \mu} \right].$$

From (5) we can easily show the following:

PROPOSITION 1. 1) If  $\beta_i \delta_i = 0$ , then  $\theta_i$  is an eigenvalue of  $M$ .

2) If  $\beta_i \delta_i$  and  $\beta_{i+1} \delta_{i+1}$  have the same sign, then there is an eigenvalue of  $M$  between  $\theta_i$  and  $\theta_{i+1}$ . If  $\beta_n \delta_n$  (resp.  $\beta_1 \delta_1$ ) is positive, then  $M$  has an eigenvalue larger than  $\theta_n$  (resp. less than  $\theta_1$ ).

3) In particular if all  $\beta_i \delta_i$ 's are positive, then all the eigenvalues of  $M$  are real and interlace with those of  $A$ .

The assumption for the last property is in particular satisfied when  $c = b$ , i.e. when  $M$  is symmetric, which gives the well known Cauchy interlace theorem for symmetric bordered matrices, see, e.g., [24, p. 186].

**2.2. The case when  $b$  or  $c$  is an eigenvector of  $A$ .** Next we consider the particular case, when either  $b$  or  $c$  is an eigenvector of  $A$ . This situation may arise around certain turning points [18] in continuation methods. If  $c$  happens to be an eigenvector of  $A$  associated with  $\theta_i$ , then all the  $\delta_j$ 's are zero, except for  $\delta_i$ . Likewise if  $b$  is an eigenvector of  $A$  associated with  $\theta_i$ , all of the  $\beta_j$ 's are zero except for  $\beta_i$ . Hence in either case, the first point of Proposition 1 yields the following result.

PROPOSITION 2. Assume that  $b$  (resp.  $c$ ) is an eigenvector of  $A$  associated with the eigenvalue  $\theta$  and let  $\gamma = c^T b$ . Then all eigenvalues of  $A$ , other than  $\theta$ , are eigenvalues of  $M$ . Moreover  $M$  has two extra eigenvalues which are roots of the following equation in  $\mu$ :

$$(6) \quad \mu^2 - (\theta + d)\mu + d\theta - \gamma = 0.$$

Furthermore, when  $\gamma$  is nonnegative, the two roots are real.



It follows from the last part of the proposition that, for some iterative methods, it may pay to scale the last row by a sign change so that the extra eigenvalues are real.

**2.3. Application to arclength continuation methods for nonlinear elliptic problems.** The above proposition determines completely the spectrum of  $M$  in the situation when either  $b$  or  $c$  is an eigenvector of  $A$ . In the context of pseudo-arclength continuation methods, the matrix  $A$  becomes singular near a *turning point* and the vector  $c$  (a scalar multiple of  $\dot{u}$ ) becomes an eigenvector of  $A$  associated with the eigenvalue zero [8], [18]. Moreover, the scalar  $d$  (a multiple of  $\dot{\lambda}$ ) also goes to zero. Hence the result of Proposition 2 applies. The spectrum of  $M$  consists of all the nonzero eigenvalues of  $A$ , plus the two eigenvalues that are solutions of (6). Of particular interest in the use of iterative methods is the size of these eigenvalues as compared to that of the nonzero eigenvalues of  $A$ , as the rate of convergence of most iterative methods depends on the spread of the eigenvalues of  $M$ .

We shall use the following two-dimensional nonlinear elliptic problem as a typical example:

$$(7) \quad \Delta u + \lambda e^u = 0$$

with zero Dirichlet boundary conditions on a unit square. Consider a typical second order finite difference discretization of (7) with mesh spacing  $h$ . This problem has a turning point at  $\lambda \approx 6.8$ , see [8]. In a typical continuation algorithm, the matrix  $A$  and the vector  $b$  of the matrix  $M$  correspond to the linearized and discretized version of (7) and  $e^u$  respectively and the last row of the matrix  $M(c^T, d)$  is usually a scaled multiple of  $(h^2 \dot{u}^T, \dot{\lambda})$ . The scaling of the last row is arbitrary and we choose to scale it to be  $(\dot{u}^T, h^{-2} \dot{\lambda})$  because this makes the vectors  $b$  and  $c$  have the same order of magnitude and the matrix  $M$  more symmetric. With this scaling, we can easily verify that  $\theta_1 = O(1)$ ,  $\theta_2 = O(1)$ ,  $\theta_n = O(h^{-2})$  and  $\gamma = O(h^{-2})$ . Moreover, except near the turning point,  $d = O(h^{-2})$ . On the lower branch all the eigenvalues of  $A$  are negative and  $d$  is positive. As the turning point is crossed over to the upper branch,  $\theta_1$  becomes positive and  $d$  becomes negative.

Based on these order of magnitude estimates, we shall now try to estimate the size of the two extra eigenvalues  $\mu_+$  and  $\mu_-$  as the turning point is approached, as given by the solution of the quadratic equation (6). First, we shall assume that  $d(-\theta) = -d(\theta)$  which is at least qualitatively correct. With this simplifying assumption, it easily follows that  $\mu_+(-\theta) = -\mu_-(\theta)$  and thus we can limit our attention to the lower branch where  $\theta_1$  is negative. Next, we consider two regions on the solution branch: one near the turning point and the other away from it.

- **Region I:** Near the turning point, we have the estimates  $\theta_1 \approx 0$  and  $\gamma = O(h^{-2}) \gg d \gg \theta_1$ . The quadratic equation (6) then reduces to  $\mu^2 - d\mu - \gamma \approx 0$  from which one can easily derive the estimates  $\mu_+ \approx \sqrt{\gamma + d/2} = O(h^{-1})$ ,  $\mu_- \approx -\sqrt{\gamma + d/2} = O(h^{-1})$ .

- **Region II:** Away from the turning point, we have the estimates  $d = O(h^{-2})$ ,  $\theta_1 = O(1)$  and thus  $d \gg \theta_1$ . The quadratic equation (6) then reduces to  $\mu^2 - d\mu + d\theta_1 - \gamma = 0$  from which one can easily derive the estimates  $\mu_+ \approx d = O(h^{-2})$ ,  $\mu_- \approx \theta_1 - \gamma/d = O(1)$ .

Based on these estimates and (6), we can deduce the behaviour of the two new eigenvalues as  $\theta_1$  goes to zero, as illustrated in Fig. 2.1. There are two things worth noting in Fig. 2.1. First, the extra eigenvalues are always in the range  $[O(1), O(h^{-2})]$ , i.e., they are of the same order of magnitude as the other eigenvalues of  $A$ . Thus they do not in general increase the spread of the spectrum of  $A$ . Second, the minimum absolute value taken on by these two eigenvalues occurs when  $\theta_1$  is near, but not at,

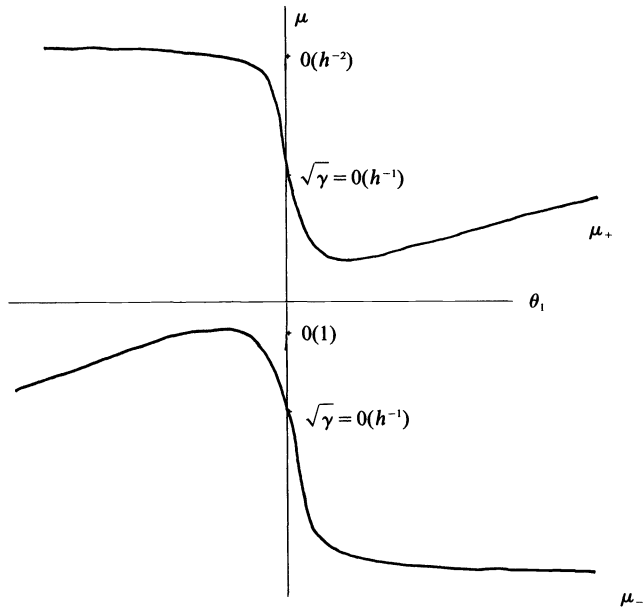


FIG. 2.1. Behaviour of the two extra eigenvalues of  $M$ .

zero. Although this minimum value is  $O(1)$  in general, its actual value can be considerably smaller than  $\theta_2$ , as is verified by numerical experiments in § 5. In the symmetric case such a small isolated eigenvalue does not usually affect the convergence too much. However, in the nonsymmetric case, it is observed that the situation may be quite different, especially in the nonpreconditioned case.

**3. Iterative method.** We are interested in several ways of applying Krylov subspace methods to system (1). A classical approach widely used in conjunction with direct methods is the block elimination algorithm which requires the solution of two systems with  $A$  as is described in § 3.1.

An interesting alternative to this approach is to work directly with  $M$ , which unlike  $A$ , is not singular near the singular point. Here, only one system with  $M$  has to be solved but the problem becomes unsymmetric. This will be described in § 3.2.

Somewhere in between these two approaches lies a whole class of methods in which one attempts to split  $M$  into the sum of a symmetric matrix and a low rank matrix. These will be described in § 3.3.

**3.1. Block-elimination.** An algorithm which is very useful in the context of *direct* methods is the following block-elimination algorithm:

ALGORITHM BE [8], [18]. (1) Solve

$$(8, 9) \quad Av = b, \quad Aw = f.$$

(2) Compute

$$(10) \quad y = \frac{(g - c^T w)}{(d - c^T v)}.$$

(3) Compute

$$(11) \quad x = w - yv.$$

With direct methods, the work consists mainly of one factorization of  $A$  and two backsolves with the  $LU$  factors of  $A$ . Moreover, for problems with many right-hand sides, the factorization need be computed only once. Since the factorization usually costs much more than a backsolve, the cost of Algorithm BE is, in general, approximately the same as that of factoring the matrix  $M$  directly. Furthermore, Algorithm BE is modular in the sense that special structures or solvers for  $A$ , when available, can be easily exploited. For example, in some cases, a fast elliptic solver can be applied to (8) and (9) although this is limited to separable problems on special domains. When such is not the case iterative methods may become competitive. With iterative methods, however, *two* linear systems of dimension  $n$  have to be solved for each system of the form (1). On the other hand, Algorithm BE does exploit fully any special properties that may be possessed by  $A$ , for example, symmetry or positive definiteness, which are important for the convergence of the iterative methods but are usually not inherited by the matrix  $M$ .

In situations where  $A$  is nearly singular, the iterative methods may encounter some convergence difficulties in solving (8) and (9). In application to the path following continuation methods mentioned in § 1, this usually does not present great difficulties for the purpose of tracing the solution curves, unless one is actually interested in computing the singular points themselves accurately [1], [9], [20], [22]. For direct methods, Algorithm BE can be shown to be unstable if  $A$  is nearly singular but can be stabilized through deflation techniques [6], [29], while retaining most of its desirable properties with minimal overhead [5]. We are currently investigating similar deflation techniques to be used with iterative methods.

### 3.2. Methods that work on $M$ directly.

Consider the linear system

$$(12) \quad Mz = p,$$

where  $M$  is unsymmetric. If  $z_0$  is an initial approximation of  $z$ , and  $r_0$  the corresponding residual vector  $r_0 = b - Mz_0$ , then one defines the  $j$ th Krylov subspace  $K_j$  as the linear span of the finite sequence  $r_0, Mr_0, \dots, M^{j-1}r_0$ . The Krylov subspace methods consist of finding an approximate solution to (12) belonging to the affine subspace  $z_0 + K_j$ , such that the residual vector  $r_j$  of  $z_j$  satisfies certain Galerkin conditions. Among such methods let us mention the ORTHOMIN ( $k$ ) methods studied by Vinsome [30] and by Eisenstat, Elman and Schlutz [11], the method of Axelsson [3], the ORTHODIR and ORTHORES methods due to Jea and Young [17] and the incomplete orthogonalization method (IOM) described by Saad in [28]. ORTHOMIN ( $k$ ) and ORTHORES ( $k$ ) require that the symmetric part of  $M$  be positive definite, in order that they do not breakdown.

In the numerical experiments of this paper we will use the IOM method, a full description of which may be found in [28]. The main properties characterizing the method are the following:

$$(13) \quad z_j \in z_0 + \{r_0, Mr_0, \dots, M^{j-1}r_0\},$$

$$(14) \quad (r_i, r_j) = 0, \quad j - k \leq i < j,$$

where  $z_i$  and  $r_i$  denote the iterate and the corresponding residual at the  $i$ th iteration and  $k$  is an integer parameter larger than one.<sup>2</sup> In other words the residual vector  $r_j$

<sup>2</sup> When  $M$  is symmetric any value of  $k$  larger than one leads to the classical conjugate gradient method [28]. The value  $k = 1$  would lead to the steepest method and is not considered.

is orthogonal to the previous  $k$  residuals. Note that  $k$  vectors from the previous iterations have to be stored.

In the Krylov subspace methods, the only operations performed with  $M$  are operations of the form  $y = Mz$ , i.e. matrix-vector multiplications. Such operations are easy to perform for bordered matrices and cost at most  $2n$  more multiplications than the corresponding operation with  $A$ . This feature makes it possible to take full advantage of sparsity.

**3.3. Symmetric splittings.** Consider the matrix  $M$  in (1). If  $A$  is symmetric, then clearly  $M$  is a low rank perturbation of a symmetric matrix, and one would like to take advantage of this fact. Let us split  $M$  as follows:

$$(15) \quad M = \begin{bmatrix} A & c \\ c^T & d \end{bmatrix} + \begin{bmatrix} b-c \\ 0 \end{bmatrix} e_{n+1}^T.$$

We will denote by  $S_1$  the first matrix on the right-hand side of (15). Then writing the second matrix of (15) as  $uv^T$ , the solution of (1) can easily be obtained via the Sherman and Morrison formula [16],

$$(16) \quad (S_1 + uv^T)^{-1}p = S_1^{-1}p - \sigma S_1^{-1}u$$

with

$$\sigma = \frac{v^T S_1^{-1}p}{1 + v^T S_1^{-1}u}.$$

To apply the above formula, one needs to solve two systems with  $S_1$ , namely  $S_1^{-1}p$  and  $S_1^{-1}u$ . Note, however, that as opposed to methods where  $M$  is used directly (§ 3.2) these systems are symmetric.

Since the matrix  $S_1$  is symmetric but not, in general, positive definite, we must resort to some generalization of the conjugate gradient similar to the SYMMLQ algorithm for solving the systems involving  $S_1$  [23]. We have used a method based on the IOM algorithm which is equivalent to the SYMMLQ algorithm, but slightly less expensive [28].

Note that there are other ways of splitting  $M$ . Here are two other possibilities:  $M = S_2 - vu^T$ , with  $u$  and  $v$  defined earlier and

$$(17) \quad S_2 = \begin{bmatrix} A & b \\ b^T & d \end{bmatrix}$$

and  $M = S_3 + (M - S_3)$ , where  $S_3$  is

$$(18) \quad S_3 = \begin{bmatrix} A & e \\ e^T & d \end{bmatrix}$$

with  $e = \frac{1}{2}(b + c)$ .

Only the splitting  $S_1$  defined by (15) will be considered in this paper.

Although symmetry is an important factor in iterative methods, it is not clear a priori whether solving two symmetric linear systems instead of one unsymmetric linear system of the same dimension will be more costly. The main objective of the numerical experiments to be described in § 5 is to provide some empirical evidence on this issue.

An interesting observation concerning the computational work of block elimination and symmetric splitting is that in both cases we have to solve two linear systems with matrices of dimensions differing by one. Note also that  $S_1$  is a low rank perturbation

of  $A$  and therefore we may expect the methods to converge in approximately the same number of steps. There is however a big difference in the context of pseudo-arclength continuation methods, which is that  $A$  becomes nearly singular near the singular point, while  $S_1$  is nonsingular as can be easily seen from Proposition 2 and the fact that  $c$  is an eigenvector of  $A$  associated with the eigenvalue zero. Thus, in theory, the symmetric splitting algorithm should be more robust.

**3.4. Conjugate gradient method on the normal equations.** Another classical way of preserving symmetry, is via the normal equations

$$(19) \quad M^T M z = M^T p.$$

The regular conjugate gradient algorithm can then be applied to (19) which is positive definite. However, not only is the amount of work per step doubled but, as is well known, the condition number of  $M^T M$  is the square of that of  $M$ , thus rendering the method slowly convergent.

**4. Preconditionings.** The use of a good preconditioning is often essential for the successful application of Krylov subspace based iterative methods. In this section, we shall discuss the use of preconditioning techniques in the algorithms presented in § 3. For this purpose, we shall assume that a good preconditioning is available for the matrix  $A$  in the form of a matrix  $B$  such that  $B^{-1} \approx A^{-1}$  and such that the matrix-vector product  $B^{-1}x$  is easy to compute. Since we wish to exploit the symmetry of  $A$ , we shall also assume that  $B$  is symmetric, so that a symmetric method can be used with the preconditioned systems in some of the methods.

The use of preconditioning in the block-elimination algorithm is straightforward, because the preconditioning  $B^{-1}$  can be applied directly to the systems with  $A$  as coefficient matrix. A possible difficulty with this approach occurs when  $A$  is *indefinite*. The reason is that in order to use the symmetric preconditioned conjugate gradient algorithm, the preconditioner must be symmetric positive definite [7]. When  $A$  is indefinite, it may not be easy to find a positive definite preconditioner  $B^{-1}$  that is “close” to  $A^{-1}$  in some sense. If  $A$  is not too indefinite, however, the situation may not be too serious because one can use a *shifted* incomplete factorization of  $A$  to obtain a reasonably good preconditioner [19].

For the other two algorithms presented in § 3, the construction of a preconditioner is not as straightforward. We shall only consider the more general unsymmetric case here (§ 3.2), as the same techniques can be applied to the symmetric splittings as well. One way to obtain a preconditioning for  $M$  based on one for  $A$  is to first express the exact inverse of  $M$  in terms of  $A^{-1}$  and then replace  $A^{-1}$  by  $B^{-1}$ . Thus, we have

$$(20) \quad M^{-1} = \left[ \begin{array}{c|c} A^{-1}(I - bu^T) & v \\ \hline u^T & -y^{-1} \end{array} \right],$$

where

$$(21) \quad y = c^T A^{-1} b - d, \quad u = \frac{A^{-1} c}{y}, \quad v = \frac{A^{-1} b}{y}.$$

Replacing  $A^{-1}$  by  $B^{-1}$  in (20), one obtains the following preconditioner for  $M$ :

$$(22) \quad P1 = \left[ \begin{array}{c|c} B^{-1}(I - b\tilde{u}^T) & \tilde{v} \\ \hline \tilde{u}^T & -\tilde{y}^{-1} \end{array} \right],$$

where the “ $\tilde{\sim}$ ” quantities are defined analogous to (21), with  $A^{-1}$  replaced by  $B^{-1}$ . We

assume here that the preconditioner  $B$  is chosen so that  $\tilde{y}$  is nonzero. Note that the preconditioning is nonsymmetric.

The above preconditioning requires some preprocessing to compute the quantities  $y$ ,  $u$  and  $v$  and the matrix-vector product  $P_1 z$  requires a few more inner-products to compute than does  $B^1 z$ . For this reason, we shall also consider the following simpler (also nonsymmetric) preconditioning:

$$(23) \quad P_2 = \begin{bmatrix} B^{-1} & 0 \\ 0 & 1 \end{bmatrix}.$$

Since  $P_1$  is nonsymmetric it should only be used with  $M$  and not with the symmetric  $S_i$ 's.

We note that a preconditioning similar to  $P_1$  has been used in a slightly different context by Bristeau et al. [4] for transonic flow problems. It is of interest to compare the two preconditionings  $P_1$  and  $P_2$ , assuming that we have the same preconditioning  $B$  for  $A$ . Let  $E = I - AB^{-1}$  and consider first the "error"  $I - MP_2$ , in the preconditioning  $P_2$ . We clearly have

$$(24) \quad I - MP_2 = \begin{bmatrix} E & -b \\ -c^T B^{-1} & 1 - d \end{bmatrix}.$$

For  $P_1$  a similar but somewhat more complicated computation leads to the equality

$$(25) \quad I - MP_1 = \begin{bmatrix} E(I - b\tilde{u}^T) & \tilde{y}^{-1}Eb \\ 0 & 0 \end{bmatrix}.$$

A comparison of (24) and (25) indicates that if  $E$  is small then the preconditioning  $P_1$  will be more accurate than  $P_2$  in general. This is not surprising because of the way this preconditioning is constructed. In general, however, the norm of  $E$  will not be small. On the other hand the effect of a preconditioning  $P$  of  $M$  is not to provide a small error  $E$  in the inverse but rather to transform the eigenvalues of  $P^{-1}M$  in such a way that most of them will be close to one. From this point of view the two preconditioned matrices  $P_1 M$  and  $P_2 M$  should not behave too differently as they only differ by a low rank perturbation. This fact is confirmed by the numerical experiments.

**5. Numerical experiments.** When the preconditioning techniques presented in § 4 are combined with the basic algorithms of § 3, a large number of methods result. In order to focus our discussions on a few representative methods and to facilitate the presentation of numerical results, we shall limit our attention to the combinations in Table 5.1.

The algorithms in Table 5.1 were implemented in a path-following continuation program package written by the authors for tracing solution curves of parameterized

TABLE 5.1  
*List of algorithms.*

---

BE:	Block-elimination, symmetric conjugate gradient for $A$
M:	Nonsymmetric conjugate gradient for $M$
SS:	Symmetric splitting, symmetric conjugate gradient for $S_1$
NE:	Normal equation on $M$ , symmetric positive definite conjugate gradient
PBE:	Preconditioned BE, symmetric conjugate gradient for $B^{-1}A$
P1M:	Nonsymmetric conjugate gradient for $P_1 M$
P2M:	Nonsymmetric conjugate gradient for $P_2 M$
P2SS:	Preconditioned symmetric splitting, symmetric conjugate gradient for $P_2 S_1$

---

nonlinear eigenvalue problems. All our numerical experiments were performed in the context of applying this program package to solve the following model nonlinear elliptic eigenvalue problem:

$$(26) \quad G(u, \lambda) \equiv \Delta u + \lambda e^u = 0,$$

on  $[0, 1] \times [0, 1]$  with zero Dirichlet boundary conditions. This problem is discretized by a standard five-point finite difference formula on a uniform  $m$  by  $m$  grid, resulting in a system of nonlinear equations of size  $n = (m-1)^2$ . The solution curve for this problem has one simple turning point at  $[\lambda \approx 6.808, u(0.5, 0.5) \approx 1.3]$  (for  $m = 20$ ), where the Jacobian  $G_u$  is singular [8]. The matrix  $A \equiv G_u$  is symmetric, and sparse (banded and no more than 5 nonzeros per row). On the lower branch of the solution curve  $A$  is negative definite, whereas on the upper branch, it is indefinite, with one eigenvalue being positive. For the preconditioner, we choose  $B \equiv \Delta$ , which is symmetric positive definite. For the basic continuation procedure, we use the pseudo-arclength parameterization of Keller [8], [18], corresponding to using the unit tangent vector  $[\dot{u}^T, \dot{\lambda}]$  for the vector  $[c, d]$ . At each step of the continuation procedure, a Newton iteration is used to bring a predicted solution to converge to the solution curve. At each step of the Newton iteration, a linear system of form (1) has to be solved.

The iterative methods that we have used are the usual conjugate gradient method for symmetric positive definite matrices, a method similar to SYMMLQ [23], [28] for symmetric indefinite problems, and the incomplete orthogonalization method (IOM) for general nonsymmetric systems [28].

A form of truncated Newton method was used for the corrector [10]. The conjugate gradient like inner iteration was stopped when the norm of the residual was reduced by a factor less than  $10^{-3}$ , and the Newton iteration was stopped when the norm of the residual for the nonlinear equations was less than  $10^{-4}$ . All computations were performed on a VAX-780 with mantissa of 24 bits, corresponding to a relative machine precision of about  $10^{-7}$ .

The experiments were carried out by starting at the trivial solution  $[0, 0]$  and tracing the solution curve slightly past the turning point. In the case of block elimination methods this represents the total of the two solves with  $A$ . For each of the methods listed in Table 5.1, we record the total number of inner iterations used by the iterative method. We note that exactly the same continuation steps are taken in the outer iteration independent of which iterative methods is used in the inner Newton iteration. The continuation steps are tabulated in Table 5.2. They are automatically chosen by

TABLE 5.2  
Continuation steps.

	$\lambda$	$\ u\ _\infty$
I	0.0	0.0
	1.0	0.078
	2.996	0.270
	4.991	0.555
	5.984	0.792
	6.475	0.990
II	6.594	1.065
	6.804	1.404
	6.697	1.660
	6.466	1.897

TABLE 5.3  
Total number of iterations.

method	$m = 10$		$m = 20$	
	I	II	I	II
BE	305	465	633	1554
M ( $k = 9$ )	206	328	623	2217
SS	350	533	737	1193
NE	433	1426	large	large
PBE	72	134	72	143
P1M ( $k = 9$ )	49	85	49	88
P1M ( $k = 4$ )	49	87	49	150
P2M ( $k = 9$ )	49	84	49	80
P2M ( $k = 4$ )	49	84	49	83
P2SS	100	186	106	201

an adaptive strategy analogous to those used in ODE solvers. The results are tabulated in Table 5.3 for  $m = 10$  and 20. Since the matrix  $M$  is qualitatively different around the turning point (due to the fact that  $A$  is nearly singular), the number of iterations taken near the turning point (denoted by II in Tables 5.2, 5.3) is presented separately from the part away from it (denoted by I in Tables 5.2, 5.3).

All the methods encountered some convergence difficulties near the turning point. From the results of § 2, this may be partly due to the near-singularity of  $A$  which introduces a small eigenvalue for  $M$ . We have tabulated in Table 5.4 some of the eigenvalues of the matrix  $M$  as the turning point is approached, together with the estimates for the two extra eigenvalues  $\bar{\mu}_1$  and  $\bar{\mu}_2$  as given by the solution of (6), which is strictly applicable only at the turning point where  $c$  is an eigenvector of  $A$ . The eigenvalues of  $M$  are computed by a version of Arnoldi's method [2], [27]. We note that the estimates are rather accurate, especially around the singular point, showing that  $c$  is actually approaching an eigenvector of  $A$ . Moreover, these values confirm the qualitative behaviour described in Fig. 2.1. Observe that the value of the extra eigenvalues of  $M$  changes drastically in the neighbourhood of the turning point ( $\theta_1 \approx 0$ ). For this particular problem, the minimum absolute value for the extra eigenvalues is an order of magnitude smaller than  $\theta_2$ . We can also observe a correlation between the larger number of iterations in Table 5.3 and the small value of the extra eigenvalue in Table 5.4. For direct methods this is not as important since this effect manifests itself through the loss of a few digits, which usually is not so drastic as to make the Newton diverge.

TABLE 5.4  
Eigenvalues of  $M$ ,  $m = 20$ ,  $\mu_n \leq \dots \leq \mu_2 \leq \mu_1$

$\lambda$	$\theta_1$	$\mu_n$	$\mu_2$	$\mu_1$	$\bar{\mu}_2$	$\bar{\mu}_1$
3.0	-16.00	-3179.0	-16.0	399.0	-45.5	400.0
6.0	-8.62	-3169.0	-9.2	398.0	-8.8	398.0
6.47	-5.89	-3166.0	-6.5	395.0	-6.1	395.0
6.73	-2.95	-3163.0	-3.6	364.0	-3.7	364.0
6.8	-0.55	-3162.0	-2.2	311.0	-2.2	311.0
* 6.77	+2.23	-3162.0	-31.4	11.8	-90.2	11.8

\*  $\mu$  computed directly by Arnoldi's method,

$\bar{\mu}$  computed as solution of (6),

$\bar{\mu}_2$  is not the second largest eigenvalue of  $M$ ,  $\mu_2$  corresponds to  $\theta_2$ .



TABLE 5.5  
Work and storage per iteration.

Method	Work			*Storage
	$Ax, Mx$	Mult.	$B^{-1}$	Vector
BE	1	$7n$	0	$6n$
M	1	$(3k+2)n$	0	$(2k+2)n$
SS	1	$7n$	0	$6n$
NE	2	$5n$	0	$4n$
PBE	1	$7n$	1	$6n$
P1M	1	$(3k+2)n$	1	$(2k+2)n$
P2M	1	$(3k+2)n$	1	$(2k+2)n$
P2SS	1	$7n$	1	$6n$

\* Storage does not include  $A$  or  $B$ .

Although we have presented the total number of inner iterations for each method in Table 5.3, the work per iteration is different for each method. In Table 5.5, we tabulated the work per step and the storage requirement of each of the competing methods.

**6. Conclusions.** In this section, we wish to draw some conclusions based on the numerical results presented in § 5.

First observe that when no preconditioning is used, a large part of the computational effort is spent near the turning point. In this case, method ( $M$ ) takes less iterations in all but the last column, i.e., near the turning point. This shows the importance of symmetry near a singular point if the systems are not preconditioned. By comparing the corresponding preconditioned methods (PBE), (P2SS) and (P2M,  $k=4$ ), and their corresponding operation counts in Table 5.5, we find that, when a preconditioning is available, it seems best to work directly with an iterative method on the unsymmetric matrix  $M$ . Note that from Table 5.5, each step of method (P2M) with  $k=4$  is less expensive than with  $k=9$ , and is not much more expensive than each step of method (PBE) or (P2SS).

Also of importance is the observation that despite the fact that the matrices do not have positive definite symmetric parts, a simple preconditioning based on the direct solver associated with a shift of the matrix can be quite effective. The preconditioning continues to work around the singular point for which the matrix is badly conditioned. The two preconditionings  $P_1$  and  $P_2$  yield results that are almost identical, except for a slight difference around the turning point. Surprisingly, the less accurate  $P_2$  always does better than  $P_1$ . Their performances may be more equal if a better approximation  $B$  of the inverse of  $A$  is available. Note that for  $k=4$ , P1M has some difficulties around the turning point for the case  $m=20$ . One possible explanation for this is that  $P_1$  is not symmetric positive definite while  $P_2$  is. In general, it seems that using a small value for  $k$  may be unreliable.

Finally we can assess the various methods tested as follows.

- If a good preconditioning is available then the preconditioned unsymmetric conjugate gradient method (P2M) gives the best results in execution time.
- Symmetry is not as important for well conditioned problems as for ill-conditioned ones. In particular as shown in Table 5.3, the iterative methods that do not use preconditioning are slow and sensitive to symmetry near the singular point.
- The normal equations approach is to be avoided if unpreconditioned.

**Acknowledgments.** The authors would like to thank Prof. Stanley Eisenstat for his helpful suggestions throughout this project and the referees for their comments and corrections.

## 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] W. E. ARNOLDI, *The principle of minimized iteration in the solution of the matrix eigenvalue problem*, Quart. Appl. Math., 9 (1951), pp. 17–29.
- [3] O. AXELSSON, *Conjugate gradient type methods for unsymmetric and inconsistent systems of linear equations*, Linear Algebra Appl., 29 (1980), pp. 1–16.
- [4] M. O. BRISTEAU, R. GLOWINSKI, J. PERRIAUX AND G. POIRIER, *Nonunique solutions of the transonic equation by arc length continuation techniques and finite element least squares methods*, Proc. 5th International Conference on Finite Elements and Flow Problems, Austin, Texas, Jan. 23–26, 1984.
- [5] T. F. CHAN, *Deflation techniques and block-elimination algorithms for solving bordered singular systems*, Tech. Rep. 226, Computer Science Dept., Yale Univ., New Haven, CT, 1982; this Journal, 5 (1984), pp. 121–134.
- [6] ———, *Deflated decomposition of solutions of nearly singular systems*, Tech. Rep. 225, Computer Science Dept., Yale Univ., New Haven, CT, 1982; SIAM J. Numer. Anal., 21 (1984), pp. 738–754.
- [7] R. CHANDRA, *Conjugate gradient methods for partial differential equations*, Ph.D. thesis, Computer Science Dept., Yale Univ., New Haven, CT, 1978.
- [8] T. F. CHAN AND H. B. KELLER, *Arc length continuation and multi-grid techniques for nonlinear eigenvalue problems*, this Journal, 3 (1982), pp. 173–194.
- [9] T. F. CHAN, *Newton-like pseudo-arclength methods for computing simple turning points*, Tech. Rep. 233, Computer Science Dept., New Haven, CT, 1982; this Journal, 5 (1984), pp. 135–148.
- [10] R. S. DEMBO, S. EISENSTAT AND T. STEIHAUG, *Inexact Newton methods*, SIAM J. Numer. Anal., 18 (1982), pp. 400–408.
- [11] 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.
- [12] 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.
- [13] C. B. GARCIA AND W. I. ZANGWILL, *Pathways to solutions, fixed points and equilibria*, Prentice-Hall, Englewood Cliffs, NJ, 1981.
- [14] P. E. GILL, W. MURRAY AND M. WRIGHT, *Practical Optimization*, Academic Press, New York, 1981.
- [15] A. L. HAGEMAN AND D. M. YOUNG, *Applied Iterative Methods*, Academic Press, New York, 1981.
- [16] A. S. HOUSEHOLDER, *Theory of Matrices in Numerical Analysis*, Blaisdell, Johnson, CO, 1964.
- [17] K. C. JEA AND D. M. YOUNG, *Generalized conjugate gradient acceleration of nonsymmetrizable iterative methods*, Lin. Algebra and Appl., 34 (1980), pp. 159–194.
- [18] 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.
- [19] T. A. MANTEUFFEL, *An incomplete factorization technique for positive definite linear systems*, Math. Comp., 34 (1980), pp. 473–497.
- [20] R. G. MELHEM AND W. C. RHEINBOLDT, *A comparison of methods for determining turning points of nonlinear equations*, Computing, 29 (1982), pp. 201–226.
- [21] H. D. MITTELMANN, *An efficient algorithm for bifurcation problems of variational inequalities*, Tech. Rep. NA-81-14, Stanford University, Stanford, CA, 1981; Math. Comp., 41 (1983), pp. 473–485.
- [22] 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., Workshop on Bifurcation Problems and their Numerical Solution, January 15–17, Birkhauser, Dortmund, 1980, pp. 1–45.
- [23] C. C. PAIGE AND M. A. SAUNDERS, *Solution of sparse indefinite systems of linear equations*, SIAM J. Numer. Anal., 12 (1975), pp. 617–624.
- [24] B. N. PARLETT, *The Symmetric Eigenvalue Problem*, Prentice Hall, Englewood Cliffs, NJ, 1980.
- [25] W. C. RHEINBOLDT, *Solution fields of nonlinear equations and continuation methods*, SIAM J. Numer. Anal., 17 (1980), pp. 221–237.
- [26] ———, *Numerical analysis of continuation methods for nonlinear structural problems*, Computers and Structures, 13 (1981), pp. 103–113.

- [27] Y. SAAD, *Variations on Arnoldi's method for computing eigenelements of large unsymmetric matrices*, Lin. Algebra and Appl., 34 (1980), pp. 269-295.
- [28] ———, *Practical use of some Krylov subspace methods for solving indefinite and unsymmetric linear systems*, Tech. Rep. 214, Computer Science Dept., Yale Univ., New Haven, CT, 1982; this Journal, 5 (1984), pp. 203-228.
- [29] G. W. STEWART, *On the implicit deflation of nearly singular systems of linear equations*, this Journal, 2 (1981), pp. 136-140.
- [30] 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.

## ***m*-STEP PRECONDITIONED CONJUGATE GRADIENT METHODS\***

LOYCE ADAMS†

**Abstract.** This paper discusses preconditioners for the conjugate gradient method which are based on several iterations of stationary iterative methods. Necessary and sufficient conditions are given for the applicability of these preconditioners to symmetric and positive definite systems of linear equations. Efficient computer implementations of these methods are discussed and results on the CYBER 203 and the Finite Element Machine under construction at NASA Langley Research Center are included.

**Key words.** multi-color ordering, preconditioned conjugate gradient, SSOR (symmetric successive over-relaxation)

**1. Introduction.** In this paper we are concerned with the solution of a sparse  $N \times N$  system of symmetric and positive definite linear equations

$$(1.1) \quad K\mathbf{u} = \mathbf{f}$$

by preconditioned conjugate gradient (PCG) methods. For a detailed description of these methods see Concus, Golub, O'Leary (1976) and Chandra (1978).

The PCG method solves the system,  $\hat{K}\hat{\mathbf{u}} = \hat{\mathbf{f}}$ , where

$$(1.2) \quad \hat{K} = Q^T M^{-1} K Q^{-T}, \quad \hat{\mathbf{u}} = Q^T \mathbf{u}, \quad \hat{\mathbf{f}} = Q^{-1} \mathbf{f}.$$

$Q$  is a nonsingular matrix, and the symmetric and positive definite preconditioning matrix is given by  $M = Q Q^T$ . The algorithm for the solution of  $\mathbf{u}$  directly is described in Chandra (1978) and is given below where  $u$ ,  $r$ ,  $\hat{r}$  and  $p$  are vectors and  $(x, y)$  denotes the inner product  $x^T y$ .

**ALGORITHM 1.** *Preconditioned conjugate gradient algorithm.*

- (1) Choose  $u^0$ ,
- (2)  $r^0 = f - K u^0$ ,
- (3)  $M \hat{r}^0 = r^0$ ,
- (4)  $p^0 = \hat{r}^0$ ,
- (5) For  $k = 0, 1, \dots, k_{\max}$ .
  - (1)  $\alpha = (\hat{r}^k, r^k) / (p^k, K p^k)$ ,
  - (2)  $u^{k+1} = u^k + \alpha p^k$ ,
  - (3) If  $\|u^{k+1} - u^k\|_{\infty} < \epsilon$  then stop, otherwise continue,
  - (4)  $r^{k+1} = r^k - \alpha K p^k$ ,
  - (5)  $M \hat{r}^{k+1} = r^{k+1}$ ,
  - (6)  $\beta = (\hat{r}^{k+1}, r^{k+1}) / (\hat{r}^k, r^k)$ ,
  - (7)  $p^{k+1} = \hat{r}^{k+1} + \beta p^k$ .

The standard conjugate gradient algorithm results by choosing  $M = I$ .

In the next section preconditioners that are based on taking  $m$  steps of an iterative method are described, conditions for their applicability to and effectiveness for symmetric and positive definite systems are given, and their relationship to the preconditioners of Dubois, Greenbaum, Rodrigue (1979) and Johnson, Micchelli and Paul (1982)

---

\* Received by the editors April 19, 1983, and in revised form October 11, 1983. The research reported in this paper was supported in part by the National Aeronautics and Space Administration under contract NAS1-46 while the author was at the University of Virginia, Charlottesville, and in part by the National Aeronautics and Space Administration under contracts NAS1-15810, NAS1-17070 and NAS1-17130 while the author was in residence at ICASE, NASA Langley Research Center, Hampton, VA 23665.

† Institute for Computer Applications in Science and Engineering, NASA Langley Research Center, Hampton, Virginia 23665.

is discussed. In § 3, the implementation of the  $m$ -step SSOR preconditioner on parallel machines is discussed and results of this preconditioner on the CYBER 203/205 and the Finite Element Machine are included.

**2.  $m$ -step preconditioners.**

**2.1. Choosing  $M$ .** Algorithm 1 of the last section requires a symmetric and positive definite preconditioning matrix  $M$  to be specified or computed. The question arises as how to choose  $M$  so that the condition number of  $\hat{K}$ ,

$$\kappa(\hat{K}) = \frac{\max_i \lambda_i}{\min_i \lambda_i},$$

where  $\lambda_i$  are the eigenvalues of  $M^{-1}K$ , is as small as possible since the error in the  $j$ th conjugate gradient step (see Concus, Golub, O’Leary (1976)), is bounded by

$$(2.1) \quad \frac{\|\hat{u} - \hat{u}^{(j)}\|_A}{\|\hat{u} - \hat{u}^{(0)}\|_A} \leq 2 \left( \frac{\alpha - 1}{\alpha + 1} \right)^j$$

where  $\alpha = (\kappa(\hat{K}))^{1/2}$  and  $\|v\|_A = (v^T A v)^{1/2}$ .

The best choice for  $M$  in the sense of minimizing  $\kappa(\hat{K})$  is  $M = K$  but this gains nothing since  $K\hat{r} = r$  is just as difficult to solve as  $Ku = f$ . One approach that has been taken in the literature is to choose  $M$  to be an incomplete Cholesky factorization of  $K$  (Manteuffel (1979)). Another approach is to choose  $M$  to be a symmetric and positive definite splitting of  $K$  that describes a linear stationary iterative method (refer to Concus, Golub, O’Leary (1976) and the references therein).

The question of interest here is whether it would be beneficial to take more than one step of a linear stationary iterative method to produce a preconditioner  $M$  that more closely approximates  $K$ . We begin by deriving an expression for  $M$ . Let  $K = P - Q$  be a splitting of  $K$  that is associated with the linear stationary iterative method with iteration matrix  $G = P^{-1}Q$ . Then the  $m$ -step iterative method applied to  $K\hat{r} = r$  is

$$(2.2) \quad P(I + G + \dots + G^{m-1})^{-1}\hat{r}^{(m)} = [P(I + G + \dots + G^{m-1})^{-1} - (P - Q)]\hat{r}^{(0)} + r.$$

By choosing  $\hat{r}^{(0)} = 0$ , (2.2) yields

$$(2.3) \quad M = P(I + G + \dots + G^{m-1})^{-1}.$$

Rutishauser (1959) describes how to implement an inner-outer iterative method where the outer method is the conjugate gradient method.

Before we establish the necessary and sufficient conditions for  $M$  to be symmetric and positive definite, we prove the following lemma.

**LEMMA 1.** *If  $A = BC$  is symmetric,  $B$  is symmetric positive definite and  $C$  has positive eigenvalues, then  $A$  is positive definite.*

*Proof.* Multiplying  $A$  by  $B^{-1/2}$  on the left and right, we find

$$(2.4) \quad B^{-1/2}AB^{-1/2} = B^{1/2}CB^{-1/2}.$$

The matrix on the right of (2.4) is similar to  $C$  and hence has the same eigenvalues, while the matrix on the left is congruent to  $A$  and hence has the same number of positive eigenvalues; see Gantmacher (1959). Thus  $A$  has the same number of positive eigenvalues as  $C$  and so is positive definite.  $\square$

The necessary and sufficient conditions for  $M$  to be positive definite are given in Theorem 1.

**THEOREM 1.** *Let  $K = P - Q$  be a symmetric positive definite matrix and let  $P$  be a symmetric nonsingular matrix. Then*

- (1) *The matrix  $M$  of (2.3) is symmetric.*
- (2) *For  $m$  odd,  $M$  is positive definite if and only if  $P$  is positive definite.*
- (3) *For  $m$  even,  $M$  is positive definite if and only if  $P + Q$  is positive definite.*

*Proof.* To prove symmetry, we write  $M^{-1}$  as

$$(2.5) \quad M^{-1} = P^{-1} + P^{-1}QP^{-1} + P^{-1}QP^{-1}QP^{-1} + \dots + \underbrace{P^{-1}QP^{-1}Q \dots P^{-1}QP^{-1}}_{m-1 \text{ terms}}.$$

Now since  $P$  and  $K$  and hence  $Q$  are symmetric, each term in (2.5) is symmetric. Thus  $M$  is symmetric.

The matrix  $G = P^{-1}Q$  can be expressed as  $G = K^{-1/2}(I - K^{1/2}P^{-1}K^{1/2})K^{1/2}$ . Since  $P^{-1}$  is symmetric the eigenvalues of the congruence transformation  $K^{1/2}P^{-1}K^{1/2}$  are real. Hence, the eigenvalues of  $G$  are real.

To prove (2), let  $m$  be odd. If  $g$  is any eigenvalue of  $G$  other than 1, the corresponding eigenvalue of

$$R = (I + G + \dots + G^{m-1})$$

is

$$1 + g + \dots + g^{m-1} = \frac{1 - g^m}{1 - g},$$

which is positive since  $m$  is odd. If  $g = 1$ , the corresponding eigenvalue of  $R$  is equal to  $m$  and is also positive. Now, since  $P = MR$  and  $M$  is symmetric and  $R$  has positive eigenvalues, it follows from Lemma 1 that if  $P$  is positive definite then  $M$  must also be positive definite. Conversely,  $M$  can be written as  $M = PR^{-1}$ . Since  $R^{-1}$  has positive eigenvalues and  $P$  is symmetric, we conclude from Lemma 1 that if  $M$  is positive definite then  $P$  is also positive definite.

Next, to prove (3) let  $m$  be even. It is sufficient to consider  $M^{-1}$  since any conclusions about the definiteness of  $M^{-1}$  will apply to  $M$ . Since  $m$  is even,  $M^{-1}$  from (2.5) can be written as

$$M^{-1} = P^{-1}(P + PG + PG^2 + PG^3 + \dots + PG^{m-1})P^{-1},$$

or

$$M^{-1} = P^{-1}[(P + PG) + (P + PG)G^2 + (P + PG)G^4 + \dots + (P + PG)G^{m-2}]P^{-1}.$$

Now, since  $PG = Q$ ,  $M^{-1}$  can be written as

$$(2.6) \quad M^{-1} = P^{-1}(P + Q)(I + G^2 + G^4 + \dots + G^{m-2})P^{-1}.$$

Since  $P$  is nonsingular and symmetric,  $M^{-1}$  is positive definite if and only if the symmetric matrix

$$(2.7) \quad S = (P + Q)(I + G^2 + G^4 + \dots + G^{m-2})$$

is positive definite.

Assume  $P + Q$  is positive definite. Since  $S$  is symmetric and the matrix  $(I + G^2 + G^4 + \dots + G^{m-2})^{-1}$  has positive eigenvalues,  $S$  is positive definite by Lemma 1. Conversely, if  $S$  is positive definite, since  $P + Q$  is symmetric and the series  $I + G^2 + G^4 + \dots + G^{m-2}$  has positive eigenvalues,  $P + Q$  is positive definite by Lemma 1.  $\square$

Dubois, Greenbaum, and Rodrique (1979) considered a truncated Neumann series for  $K^{-1}$  as a preconditioner. Their preconditioner is equivalent to that of (2.3) if

$K = P - Q$  corresponds to a Jacobi splitting where  $P = \text{diag}(K)$ , but they do not consider more complicated splittings that result from other iterative methods. Theorem 1 extends their main result. Under the hypothesis that  $K$  and  $P$  are both symmetric and positive definite matrices and  $\rho(G) < 1$ , they prove that  $M$  is symmetric and positive definite for all  $m$ . For odd  $m$  the condition that  $\rho(G) < 1$  is not needed. The relationship between the condition  $\rho(G) < 1$  and the positive definiteness of  $P + Q$  is given in Theorem 2.

**THEOREM 2.** *Let  $K = P - Q$  be a symmetric positive definite matrix and let  $P$  be symmetric and nonsingular. Then  $\rho(P^{-1}Q) < 1$  if and only if  $P + Q$  is positive definite.*

*Proof.* First, assume  $P + Q$  is positive definite. Since  $K$  is symmetric positive definite and  $P$  is nonsingular,  $K = P - Q$  is a  $p$ -regular splitting. Hence, from Ortega's  $p$ -regular splitting theorem, Ortega (1972),  $\rho(P^{-1}Q) < 1$ .

Now, assume that  $\rho(G) < 1$ . Then  $(I - G)^{-1}$  exists and since  $G$  has real eigenvalues, it easily follows that the matrix  $H$  defined by

$$(2.8) \quad H = (I - G)^{-1}(I + G)$$

has real eigenvalues. But we know from Young (1971, p. 82) that  $H$  is  $N$ -stable. Hence  $H$  has positive eigenvalues. Now, we can write  $H$  as

$$(2.9) \quad H = K^{-1}(P + Q),$$

or equivalently,

$$(2.10) \quad K = (P + Q)H^{-1}.$$

Finally, since  $K$  is symmetric and positive definite and  $H^{-1}$  has positive eigenvalues and  $P + Q$  is symmetric, we conclude from Lemma 1 that  $P + Q$  is positive definite.  $\square$

We note that the Jacobi convergence theorem given in Young (1971) is a special case of Theorem 2.

Theorem 1 and Theorem 2 are helpful in choosing a splitting of  $K$  that will produce an  $m$ -step preconditioner that is symmetric and positive definite. For example, if the Jacobi splitting of  $K$  ( $P = D$  and  $Q = D - K$  where  $D$  is the diagonal of  $K$ ) were considered, part (3) of Theorem 1 says that if  $m$  is even,  $P + Q$  must be positive definite, and by Theorem 2 this is only true when the Jacobi method is convergent. However, for problems of interest to us, the Jacobi method is not guaranteed to be convergent since we only know that  $K$  will be symmetric and positive definite; therefore, for these problems, only odd values of  $m$  will yield  $m$ -step Jacobi preconditioning matrices that are guaranteed to be positive definite.

**2.2. Analysis of the condition number.** In the last section, we gave conditions for  $M$  to be symmetric and positive definite and hence to be considered as a preconditioner for the conjugate gradient method. In this section we determine if increasing  $m$  will, in fact, produce a better conditioned system. For this purpose, we now denote by  $M_m$  the matrix of (2.3).

As a first step towards answering this question, we derive an expression for  $\kappa(\hat{K}_m)$ . Recall from (1.2) that  $\hat{K}$  is similar to  $M_m^{-1}K$  so that  $\kappa(\hat{K}_m)$  is the same as the ratio of the largest to smallest eigenvalue of  $M_m^{-1}K$ . An expression for  $M_m^{-1}K$  as a polynomial in  $G$  is

$$(2.11) \quad M_m^{-1}K = (I + G + \dots + G^{m-1})P^{-1}(P - Q),$$

or

$$M_m^{-1}K = I - G^m,$$

where  $G = P^{-1}Q$ .

We wish to compare  $\kappa(\hat{K}_m)$  to  $\kappa(\hat{K}_{m+1})$ , when both  $M_m$  and  $M_{m+1}$  are symmetric and positive definite. By Theorem 1, this implies that  $P$  and  $P + Q$  are positive definite and thus by Theorem 2,  $\rho(G) < 1$ . Under the hypothesis of Theorem 1 the eigenvalues  $\lambda_i$  of  $G$  are real, and can be ordered as

$$-1 < \lambda_1 \leq \lambda_2 \leq \dots \leq \lambda_n < 1.$$

Furthermore, let  $\delta$  be the eigenvalue with the smallest absolute value. Then the condition number of  $\hat{K}_m$  is

$$(2.12) \quad \kappa(\hat{K}_m) = \begin{cases} \frac{1 - \lambda_1^m}{1 - \lambda_n^m}, & m \text{ odd,} \\ \frac{1 - \delta^m}{1 - \lambda_n^m}, & \lambda_n \geq |\lambda_1|, \quad m \text{ even,} \\ \frac{1 - \delta^m}{1 - \lambda_1^m}, & |\lambda_1| \geq |\lambda_n|, \quad m \text{ even.} \end{cases}$$

If  $\lambda_1 < 0$  and  $|\lambda_1| \geq |\lambda_n|$ , it is impossible to decide whether  $\kappa(\hat{K}_{m+1}) \leq \kappa(\hat{K}_m)$  without knowledge of the values of  $\lambda_1$ ,  $\lambda_n$  and  $\delta$ . The conditions for the remaining two cases are stated below:

If  $\lambda_1 \geq 0$ ,

(2.13a)  $\kappa(\hat{K}_m)$  is a nonincreasing function for all  $m$ .

If  $\lambda_n \geq |\lambda_1|$  and  $\lambda_1 < 0$ ,

(2.13b) (a) for  $m$  odd,  $\kappa(\hat{K}_{m+1}) \leq \kappa(\hat{K}_m)$ ,  
 (b) for  $m$  even,  $\kappa(\hat{K}_{m+1}) \leq \kappa(\hat{K}_m)$  if and only if  $(1 + |\lambda_1|^{m+1})(1 - \lambda_n^m) \leq (1 - \delta^m)(1 - \lambda_n^{m+1})$ .

As an application of (2.13a) consider the SSOR splitting of a symmetric and positive definite matrix. Recall from the basic convergence theorem for SSOR that if  $K$  is a symmetric matrix with positive diagonal elements, the SSOR method converges if and only if  $K$  is positive definite and  $0 < \omega < 2$ . Therefore,  $\rho(G) < 1$  for this splitting and from Young (1971) we know that all the eigenvalues of  $G$  are real and nonnegative. Since  $P$  is symmetric, it follows from Theorems 1 and 2 that  $M_m$  is symmetric and positive definite and from (2.13a) it follows that  $\kappa(\hat{K}_m)$  is a nonincreasing function of  $m$ .

Results of the  $m$ -step SSOR preconditioned conjugate gradient method on a  $1536 \times 1536$  symmetric and positive definite matrix derived from a finite element discretization (triangles with linear basis functions) of a plate in plane stress are given in Table 1 and the results on a  $768 \times 768$  matrix derived from the 5-star discretization of Laplace's equation are given in Table 2. For these problems, results are given for both the natural rowwise ordering and the multi-color ordering (see Adams and Ortega (1982)) of the grid. The convergence criterion was  $\|u^{k+1} - u^k\|_\infty < \epsilon$ , where  $\epsilon = 10^{-6}$  for both problems. The conjugate gradient results with no preconditioning are indicated by  $m = 0$ .

The results in Tables 1 and 2 show that the number of iterations is a decreasing function of  $m$  as was predicted by (2.13a). The results also indicate that there will be



TABLE 1  
*m*-step SSOR PCG for 1536 × 1536 plane stress problem.

<i>m</i>	<i>R/B/G</i>	Natural	
	# iterations ( $\omega = 1$ )	# iterations ( $\omega = 1$ )	# iterations ( $\omega = 1.6$ )
0	363	363	363
1	139	111	93
2	99	80	66
3	82	65	54
4	71	57	47

TABLE 2  
*m*-step SSOR PCG for 768 × 768 Laplace's equation.

<i>m</i>	<i>R/B</i>	Natural	
	# iterations ( $\omega = 1$ )	# iterations ( $\omega = 1$ )	# iterations ( $\omega = 1.8$ )
0	56	56	56
1	30	28	17
2	22	21	13
3	18	17	10
4	16	15	9

an optimal value of *m*, say  $m_{opt}$ ; since for  $m > m_{opt}$ , the reduction in the number of CG iterations is not enough to balance the increase in the time required for the iterations of the SSOR preconditioner. The actual relative cost of the CG and SSOR iterations on a computer will be a function of the amount of arithmetic and communication operations in each algorithm as well as the times to perform these operations on the machine. Therefore, the optimal value of *m* will depend on the architecture of the machine and the problem size as indicated by the results in § 3.

As an example of an application of (2.13b) we consider the Jacobi splitting of any symmetric and positive definite matrix *K* that has Property *A* (see Young (1971)). For this splitting,  $P = D$  where *D* is the diagonal of *K* and therefore *P* is symmetric and positive definite. Now, since *K* has Property *A*, the eigenvalues  $\lambda_i$  of *G* occur in  $\pm\lambda_i$  pairs and  $\lambda_n = |\lambda_1|$  and  $\delta = 0$ . From (2.13b) we conclude that going from *m* (even) to *m* + 1 (odd) is advantageous if and only if

$$(2.14) \quad (1 + \lambda_n^{m+1})(1 - \lambda_n^m) < (1 - \lambda_n^{m+1}),$$

or equivalently,

$$\lambda_n^{m+1} - 2\lambda_n + 1 > 0.$$

As *m* increases, the inequality in (2.14) reduces asymptotically to

$$(2.15) \quad \lambda_n < \frac{1}{2}.$$

For  $m = 2$  and  $m = 3$ , the exact conditions are  $\lambda_n < .62$  and  $\lambda_n < .53$  respectively, but for problems of interest to us,  $\lambda_n$  will be closer to 1 and we can conclude that it is not advantageous to increase *m* from *m* (even) to *m* + 1 (odd). This fact has been verified by numerical experiments for the *m*-step Jacobi preconditioner on an 89 × 89 symmetric and positive definite system that had Property *A*. The results are given in Table 3. Note from Table 3 that increasing *m* from even to odd increases the number of

TABLE 3  
*m*-step Jacobi results  $89 \times 89$ .

<i>m</i>	# iterations
0	45
1	45
2	23
3	36
4	21
5	30
6	18
7	26
8	16

iterations. On the other hand, observe that increasing  $m$  from an odd to a consecutive even number always reduces the number of iterations. Dubois, Greenbaum, Rodrique (1979) reported similar results for Poisson's equation. Their results may also be explained by (2.13b). Also note from Table 3 that the number of iterations is a decreasing function of  $m$  if we restrict  $m$  to be even. In fact this can easily be shown to be true for all three cases in (2.12).

So far we have only addressed the question of whether a better conditioned system results by increasing  $m$ . We now turn to the question of how much improvement over  $m = 1$  can be made by taking  $m > 1$  steps of the preconditioner. Dubois, Greenbaum and Rodrique (1979) proved that the  $m$ -step PCG method can at most reduce the number of iterations needed by the 1-step PCG method by a factor of  $m$ . In practice, this theoretical bound may not be reached and for a given distribution of eigenvalues it may be sharper for some values of  $m$  than for others. The results of Dubois et al. (1979) show this for the  $m$ -step Jacobi PCG for Laplace's equation. Tables 1 and 2 show for the  $m$ -step SSOR PCG method applied to both the plane stress problem and Laplace's equation that the bound is best for  $m = 2$ . Table 3 shows that for the  $m$ -step Jacobi PCG applied to a problem with Property *A* that the bound is extremely sharp for  $m = 2$  and extremely poor for odd values of  $m$ .

The above results show that the  $m$ -step PCG method is not very effective as  $m$  increases. However, by parametrizing the preconditioner by using the techniques of Johnson, Micchelli, and Paul (1982), the method is more effective. They have suggested symmetrically scaling the matrix  $K$  to have unit diagonal and then taking  $m$  terms of a parametrized Neumann series for  $K^{-1} = (I - G)^{-1}$  as the value for  $M^{-1}$ . This corresponds to a symmetric preconditioning matrix that is a polynomial of degree  $m - 1$  in  $G$ ,

$$(2.16) \quad M_m^{-1} = \alpha_0 I + \alpha_1 G + \alpha_2 G^2 + \cdots + \alpha_{m-1} G^{m-1}$$

derived from the Jacobi splitting,

$$(2.17) \quad K = I - G$$

of  $K$ . Also the values of  $\alpha$  are chosen so that  $M_m$  is positive definite and the eigenvalues of  $M_m^{-1}K$  are close to those of  $I$ . This same idea applied to the splitting

$$(2.18) \quad K = P - Q,$$

with  $G = P^{-1}Q$  yields

$$(2.19) \quad M_m^{-1} = (\alpha_0 I + \alpha_1 G + \alpha_2 G^2 + \cdots + \alpha_{m-1} G^{m-1}) P^{-1}$$

as the inverse of the parametrized preconditioner corresponding to (2.3). The appropriate values of the  $\alpha_i, i = 0, 1, \dots, m - 1$  for the SSOR splitting are given in Adams (1983). In the next section we discuss the efficient implementation of the  $m$ -step SSOR preconditioner and the choice for the relaxation parameter  $\omega$  for the SSOR method if the grid points are ordered by a multi-color ordering.

**3. Implementation and results.**

**3.1. Implementation considerations.** In order to implement efficiently the  $m$ -step SSOR preconditioner on vector computers, the equations at the grid points of the problem domain must be colored, see Adams and Ortega (1982), so that any two equations at points on the same grid point stencil are different colors. The equations are then ordered by colors with the equations of the same color being ordered left to right, top to bottom (for a rectangular grid). In particular, if three colors are used, the system  $K\hat{\mathbf{r}} = \mathbf{r}$  has the decoupled form,

$$(3.1) \quad \begin{bmatrix} D_{11} & B_{12} & B_{13} \\ B_{12}^T & D_{22} & B_{23} \\ B_{13}^T & B_{23}^T & D_{33} \end{bmatrix} \begin{bmatrix} \hat{\mathbf{r}}_1 \\ \hat{\mathbf{r}}_2 \\ \hat{\mathbf{r}}_3 \end{bmatrix} = \begin{bmatrix} \mathbf{r}_1 \\ \mathbf{r}_2 \\ \mathbf{r}_3 \end{bmatrix},$$

where  $D_{ii}, i = 1$  to 3 are diagonal matrices. The vector length will be the number of equations of each color.

For parallel arrays, the grid points (not equations) are colored for the purpose of assigning an equal number of points of each color to the processors. Once this is done, the equations at a given grid point are considered different colors so that the matrix  $K$  has the form given in (3.1).

The  $m$ -step SSOR iteration is implemented as a forward followed by a backward multi-color SOR iteration (Adams and Ortega (1982)) but care is taken to save results from the forward pass in an auxiliary vector to be used in the reverse pass so that the cost of one SSOR iteration is no more expensive than the cost of one SOR iteration (Conrad and Wallach (1979)). Specific details on this implementation (in conjunction with Algorithm 1) for the CYBER 203 and the Finite Element Machine can be found in Adams (1983).

In addition to the computational work saved by using the auxiliary vector, the multi-color ordering permits even more savings. To explain this, we begin by writing a 3-color SOR iteration matrix,  $\mathcal{L}_\omega$ , in the following factored form:

$$(3.2) \quad \mathcal{L}_\omega = G_\omega B_\omega R_\omega,$$

where  $R_\omega, B_\omega$  and  $G_\omega$  are the matrix operators for the Red, Black and Green equations respectively. Nicolaides (1974) discussed the factorization of an  $n \times n$  SOR iteration matrix  $\mathcal{L}_\omega$  into  $n$  operator matrices, one for each equation, and then showed how these factors combine for matrices with Property  $A$  into two factors,  $\mathcal{L}_\omega = B_\omega R_\omega$ , corresponding to the red and black equations respectively. Young (1971) also gives the factorization of  $\mathcal{L}_\omega$  for these 2-colored matrices. Equation (3.2) is a straightforward continuation of these ideas. To be precise, if the matrix  $K$  is given by

$$(3.3) \quad K = \begin{bmatrix} I_1 & -X_{12} & -X_{13} \\ -X_{12}^T & I_2 & -X_{23} \\ -X_{13}^T & -X_{23}^T & I_3 \end{bmatrix},$$

with no loss in generality by assuming  $D = I$  on the diagonal, the  $R_\omega, B_\omega$  and  $G_\omega$

matrices in (3.2) are

$$R_\omega = \begin{bmatrix} (1-\omega)I_1 & \omega X_{12} & \omega X_{13} \\ 0 & I_1 & 0 \\ 0 & 0 & I_2 \end{bmatrix}, \quad B_\omega = \begin{bmatrix} I_1 & 0 & 0 \\ \omega X_{12}^T & (1-\omega)I_2 & \omega X_{23} \\ 0 & 0 & I_3 \end{bmatrix},$$

and

$$(3.4) \quad G_\omega = \begin{bmatrix} I_1 & 0 & 0 \\ 0 & I_2 & 0 \\ \omega X_{12}^T & \omega X_{23}^T & (1-\omega)I_3 \end{bmatrix},$$

respectively.

Similarly, the backward multi-color SOR iteration matrix may be written in the factored form

$$(3.5) \quad \mathcal{U}_\omega = R_\omega B_\omega G_\omega,$$

where  $R_\omega, B_\omega, G_\omega$  are the same as those of (3.2). Now, the multi-color SSOR iteration matrix may be written as

$$(3.6) \quad \mathcal{S}_\omega = R_\omega B_\omega G_\omega G_\omega B_\omega R_\omega.$$

A trivial calculation shows that  $G_\omega G_\omega = G_{\omega(2-\omega)}$  and  $R_\omega R_\omega = R_{\omega(2-\omega)}$ . Hence,

$$(3.7) \quad \mathcal{S}_\omega = R_\omega B_\omega G_{\omega(2-\omega)} B_\omega R_\omega.$$

From (3.7), we see that the green equations only need to be calculated on the forward pass with relaxation factor  $\omega' = \omega(2 - \omega)$ . Likewise, the  $R_\omega$  operators combine from the backward pass and the next forward pass so that the red equations should be updated on the first forward pass with relaxation factor  $\omega$  and on the last backward pass with relaxation factor  $\omega$ . For the intermediate forward passes, the red equations should be updated with  $\omega' = \omega(2 - \omega)$ . The black equations, however, must be updated on both the forward and backward passes with relaxation parameter  $\omega$  but part of this calculation can be saved by the use of the auxiliary vector mentioned earlier. By organizing the computation in this fashion, with  $c$  colors,  $2m(c - 1) + 1$  rather than  $2mc$  operation matrices need to be applied. Also, this computational organization is not affected by the introduction of  $\alpha_i, i = 1, 2, \dots, m$  since the parameter  $\alpha_i$  multiplies only the right-hand side vector  $r$  on step  $m - i + 1$  of the preconditioner.

We now briefly discuss the choice for  $\omega$ . From Young's (1971) theory of matrices with Property A (2-colored) we know that the optimal  $\omega$  for SSOR is  $\omega = 1$ . In fact, Young's proof shows that

$$(3.8) \quad \mathcal{S}_\omega = R_\omega B_\omega B_\omega R_\omega,$$

and

$$(3.9) \quad \mathcal{S}_\omega \sim B_{\omega(2-\omega)} R_{\omega(2-\omega)} = \mathcal{L}_{\omega(2-\omega)}$$

and, for matrices with Property A,  $\mathcal{L}_{\omega(2-\omega)}$  has the smallest spectral radius whenever  $\omega = 1$ . In particular,  $\mathcal{S}_1 \sim \mathcal{L}_1$ . Now, for multi-color matrices,  $\mathcal{S}_\omega$  and  $\mathcal{L}_{\omega(2-\omega)}$  do not necessarily have the same eigenvalues, since from (3.7) with 3 colors, we see that

$$(3.10) \quad \mathcal{S}_\omega \sim B_\omega G_{\omega(2-\omega)} B_\omega R_{\omega(2-\omega)},$$

and for  $\omega = 1$ ,

$$(3.11) \quad \mathcal{S}_1 \sim B_1 \mathcal{L}_1.$$

In general, when the number of colors is equal to  $c$  and  $C_\omega^{(k)}$  denotes the matrix associated with color  $k$ ,

$$(3.12) \quad \mathcal{L}_\omega \sim C_\omega^{(2)} C_\omega^{(3)} \dots C_\omega^{(c-1)} C_{\omega(2-\omega)}^{(c)} C_\omega^{(c-1)} \dots C_\omega^{(3)} C_\omega^{(2)} C_{\omega(2-\omega)}^{(1)},$$

and for  $\omega = 1$ ,

$$(3.13) \quad \mathcal{L}_1 \sim C_1^{(2)} C_1^{(3)} \dots C_1^{(c-1)} \mathcal{L}_1.$$

(In the above equations,  $X \sim Y$  means that  $X$  and  $Y$  have the same eigenvalues.) Assume that (3.12) represents an equal number of equations of each color and let  $\omega > 1$  so that  $\omega(2 - \omega) < 1$ . For two colors, (3.9) shows that all equations are underrelaxed. For three colors, (3.10) shows that we can regard only the black equations as being overrelaxed (once on the forward and once on the reverse pass). In general, (3.12) shows that the equations of  $c - 2$  colors can be regarded as overrelaxed and the equations of 2 colors as underrelaxed. When the number of colors equals the number of equations, all but two equations can be regarded as being overrelaxed. Although not a proof, this observation suggests that overrelaxation becomes more worthwhile as the number of colors increases and choosing  $\omega = 1$  when a small number of colors is used is a good choice. This was the case for the results in Table 1, where for the  $R/B/G$  ordering of nodes (really six colors—two unknowns per node)  $\omega = 1$  was optimal for  $m$ -step SSOR PCG. Results in Adams (1982) show that  $\omega = 1$  was also optimal for the SSOR method (used alone) for this same problem.

**3.2. Results on parallel computers.** We now give results of the  $m$ -step SSOR PCG method for a square plate in plane stress on both the CYBER 203 and the Finite Element Machine. These results were discussed in detail in Adams (1983) and are only included here to show that the method is effective on these machines. Table 4 gives the number of iterations,  $I$ , and the time,  $T$ , in seconds to solve this problem using  $m = 0, 1, 2, 3, 4, 5, 6, 7, 8, 9$  and 10. The parametrized preconditioner results are denoted by  $P$ , the number of rows in the plate by  $a$ , and the maximum vector length by  $v$ .

We now give the Finite Element Machine results. The same problem with 6 rows and 6 columns of nodes (60 equations) was solved on a 1, 2, and then on a 5-processor Finite Element Machine using the  $m$ -step SSOR PCG method (as more processors become available on this machine the solution of larger problems will be possible). Each processor was assigned equations at an equal number of  $R$ ,  $B$  and  $G$  nodes. Therefore, in the absence of communication time and any differences in processor speeds, a speedup of 2 (5) over the one processor case should be realized whenever 2 (5) processors are used respectively. The number of iterations,  $I$ , and the time,  $T$ , in seconds as well as the respective speedups are given in Table 5.

**4. Summary and conclusions.** Preconditioners for a symmetric and positive definite system of linear equations based on taking  $m$  steps of an iterative method that is derived from a symmetric splitting of the coefficient matrix have been described. Necessary and sufficient conditions were given for these preconditioners to be symmetric and positive definite for  $m$  both odd and even in Theorem 1, and the relationship between a splitting and its associated iteration matrix was given in Theorem 2.

The  $m$ -step SSOR preconditioner was shown to lead to a system whose condition number was a nonincreasing function of  $m$ ; however, for small problems, the actual decrease in the number of iterations is not enough to balance the extra work involved in the preconditioner as shown in Tables 4 and 5. By parametrizing this preconditioner, the number of iterations is reduced enough so that larger values of  $m$  should be used

for smaller problems as well. The optimal number of steps of the preconditioner is seen from Tables 4 and 5 to be a function of the architecture as well as the problem. The more expensive the inner products of the outer CG iteration become, the more likely  $m$  should be increased.

TABLE 4  
CYBER 203 iterations and timings,  $m$ -step SSOR PCG.

$m$	$v=22$ $a=8$		$v=41$ $a=11$		$v=132$ $a=20$		$v=561$ $a=41$		$v=1282$ $a=62$		$v=2134$ $a=80$	
	$I$	$T$	$I$	$T$	$I$	$T$	$I$	$T$	$I$	$T$	$I$	$T$
0	112	.133	157	.213	271	.565	536	3.293	788	11.845	929	22.780
1	52	.129	66	.184	111	.454	214	2.373	311	7.832	395	17.194
2	38	.143	50	.208	79	.478	152	2.428	221	7.773	280	17.380
2P	<u>31</u>	<u>.116</u>	40	.167	61	.369	118	1.885	172	6.052	218	13.534
3	31	.155	39	.216	65	.520	124	2.585	181	8.174	229	18.469
3P	24	.121	30	.167	46	.369	88	1.836	129	5.828	163	13.151
4P	22	.138	<u>24</u>	<u>.166</u>	35	.350	67	1.726	99	5.471	124	12.306
5P	19	.143	20	.167	<u>29</u>	<u>.347</u>	56	1.716	82	5.345	104	12.260
6P	18	.159	18	.175	25	.348	47	1.670	70	5.263	88	12.011
7P					26	.413	43	1.739	64	5.451	80	12.410
8P					21	.375	<u>36</u>	<u>1.634</u>	54	5.139	69	11.985
9P							33	1.660	<u>48</u>	<u>5.056</u>	61	11.731
10P							31	1.709	44	5.070	<u>55</u>	<u>11.594</u>

TABLE 5  
FEM iterations, timings, speedups,  $m$ -step SSOR PCG.

$m$	$p=1$		$I$	$p=2$		Speedup	$I$	$p=5$		Speedup
	$I$	$T$		$T$	$T$			$T$	Speedup	
0	48	63.35	49	33.70	1.92	48	17.70	3.58		
1	19	47.90	19	25.85	1.85	19	14.85	3.23		
2	13	48.75	13	26.65	1.83	13	15.50	3.15		
2P	11	41.95	11	22.95	1.83	11	13.30	3.15		
3	11	54.95	11	30.15	1.82	11	17.65	3.11		
3P	8	41.25	8	22.75	1.81	8	13.25	3.11		
4	10	62.40	10	34.30	1.82	10	20.20	3.09		
4P	<u>6</u>	<u>39.80</u>	<u>6</u>	<u>22.00</u>	1.81	<u>6</u>	<u>12.90</u>	3.09		
5P	5	40.60	5	22.50	1.80	5	13.25	3.06		
6P	5	47.05	5	26.20	1.80					

We noted that although a theoretical optimal value of  $\omega$ , the relaxation parameter for the SSOR method, can not be found, the choice  $\omega = 1$  (when the nodes are ordered by the multi-color ordering) was optimal for our plane stress test problem (6 colors). It is well known that  $\omega = 1$  is optimal for SSOR for matrices that have Young's Property A (Red/Black), but in general this theory does not extend beyond two colors. However, we conjectured that if the number of colors is small, choosing  $\omega = 1$  is a good choice.

A problem still remains in applying the method to irregular regions since the grid must be colored and for array machines must also be distributed to the processors in light of this coloring.

**Acknowledgments.** The author would like to thank Professor James M. Ortega for his useful discussions and suggestions concerning this manuscript. The author also acknowledges useful suggestions by the referees.

## REFERENCES

- [1] L. ADAMS AND J. ORTEGA, *A multi-color SOR method for parallel computation*, Proc. 1982 IEEE Conference on Parallel Processing, Bellaire, MI, August 1982, pp. 53–58.
- [2] L. ADAMS, *Iterative algorithms for large sparse linear systems on parallel computers*, Ph.D. thesis, Dept. Applied Mathematics and Computer Science, Univ. Virginia, January 1983; NASA Contractor Report 166027, NASA Langley Research Center, Langley, VA 1982.
- [3] ———, *An  $m$ -step preconditioned conjugate gradient method for parallel computation*, Proc. 1983 IEEE Conference on Parallel Processing, Bellaire, MI, August 1983, pp. 36–43.
- [4] R. CHANDRA, *Conjugate gradient methods for partial differential equations*, Ph.D. thesis, Research Report #129, Dept. Computer Science, Yale Univ., New Haven, CT, 1978.
- [5] P. CONCUS, G. GÖLUB AND D. O'LEARY, *A generalized conjugate gradient method for the numerical solution of elliptic partial differential equations* in Sparse Matrix Computations, J. Bunch and D. Rose, eds., Academic Press, New York, pp. 309–332.
- [6] V. CONRAD AND Y. WALLACH, *Alternating methods for sets of linear equations*, Numer. Math., 32 (1979), pp. 105–108.
- [7] P. DUBOIS, A. GREENBAUM AND G. RODRIQUE, *Approximating the inverse of a matrix for use in iterative algorithms on vector processors*, Computing, 22 (1979), pp. 257–268.
- [8] F. GANTMACHER, *The Theory of Matrices*, Vol. 1, Chelsea, New York, 1959.
- [9] O. JOHNSON, C. MICCHELLI AND G. PAUL, *Polynomial preconditioners for conjugate gradient calculations*, IBM Research Report #40444#, IBM Thomas J. Watson Research Center, Yorktown Heights, NY, 1982.
- [10] T. MANTEUFFEL, *An incomplete factorization technique for positive definite linear systems*, Math. Comp., 34 (1979), pp. 473–479.
- [11] R. NICOLAIDES, *On a geometrical aspect of SOR and the theory of consistent ordering for positive definite matrices*, Numer. Math., 27 (1974), pp. 99–104.
- [12] J. ORTEGA, *Numerical Analysis: A Second Course*, Academic Press, New York, 1972.
- [13] H. RUTISHAUSER, *Theory of gradient methods*, in Refined Iterative Methods for Computation of the Solution and Eigenvalues of Self-Adjoint Boundary Value Problems, T. Engeli, H. Ginsburg, H. Rutishauser and E. Stiefel, Mitteilungen aus dem Institut für angewandte Mathematik, No. 8, Birkhäuser-Verlag, Berlin, Chapter II, pp. 24–45.
- [14] D. YOUNG, *Iterative Solution of Large Linear Systems*, Academic Press, New York, 1971.

## THE VECTOR HARMONIC ANALYSIS OF LAPLACE'S TIDAL EQUATIONS\*

PAUL N. SWARZTRAUBER† AND AKIRA KASAHARA†

**Abstract.** A modal analysis is presented of the linearized shallow-water equations on the sphere called Laplace's tidal equations, using the spherical vector harmonics. The present approach is more compact and conceptually simpler than past analyses in which the order of the differential equation was raised. The modes, called Hough harmonics, are expressed as a series in the spherical vector harmonics whose coefficients are then computed as the eigenvectors of an infinite, banded, symmetric, linear system of equations. The frequencies of the modes are determined as the eigenvalues of the banded system. New zonal rotational modes for zonal wavenumber  $m=0$  are obtained as the limit of Hough vector harmonics as  $m$  tends to zero. Although the zonal rotational modes are not unique, this new set of functions shares many properties with the nonzonal rotational modes, including orthogonality of characteristic functions. Some limiting cases of the Hough harmonics are also discussed, including the Haurwitz modes, which are determined as the limit of solutions to the untransformed shallow-water equations as the equivalent height tends to infinity. Finally, a description of the software for computing the Hough harmonics as well as the Haurwitz modes is presented. This software package is available from the National Center for Atmospheric Research and consists of four user-entry FORTRAN subroutines. Subroutine SIGMA computes the frequencies of the normal modes. Subroutine ABCOEF computes the coefficients in the expansion of the normal modes in terms of the spherical vector harmonics. Subroutine UVH tabulates the components of the meridional structure of the Hough vector functions and subroutine UVHDER tabulates certain derivatives of the components.

**Key words.** spherical vector harmonics, Hough functions, Laplace tidal equations, shallow water equations

**1. Introduction.** The motions of a thin layer of incompressible, homogeneous, and hydrostatic fluid over a rotating sphere are described by the shallow-water equations, which are of fundamental importance to meteorology and oceanography [21]. The linearized version of the inviscid shallow-water equations with respect to the resting basic state are called Laplace's tidal equations. (See references in [15].) Eigensolutions of Laplace's tidal equations have been used to solve atmospheric tidal problems [1].

More recently, eigensolutions of free oscillations described by Laplace's tidal equations, referred to as the normal modes, have been applied in the analysis of global meteorological data [7] and in the prediction of global atmospheric motion [13]. The representation of meteorological data in terms of the normal modes is used to identify the characteristics of the wave motions which evolve from the global initial data. This property of normal mode expansion has been used extensively to initialize the input data for global numerical weather prediction models. (See references in [3].)

Various aspects of the eigensolutions of Laplace's tidal equations, including the methods of solution, their asymptotic characters, and tables of their eigenvalues and eigenfunctions, are discussed in [17], [10], [6], [16], [12] and [4]. Because Hough [10] was the first to solve the normal mode problem by means of spherical harmonics, the eigensolutions of Laplace's tidal equations are now referred to as *Hough functions* [24].

One of the main purposes of this article is to describe the software that has been developed at the National Center for Atmospheric Research to calculate the Hough functions. In § 2, we describe Laplace's tidal equations and derive their form in dimensionless variables. Solutions of Laplace's tidal equations are presented in the case of positive fluid depth, and the orthogonality of normal modes with distinct frequencies is discussed.

\* Received by the editors August 9, 1983, and in final revised form February 13, 1984.

† National Center for Atmospheric Research, Boulder, Colorado 80307. The center is sponsored by the National Science Foundation.



In § 3, we describe the method of solving Laplace's tidal equations used in the software. Earlier, Kasahara [12] presented a method which is similar to, but different from that discussed by Longuet-Higgins [16]. We show that this earlier approach can be rewritten compactly using spherical vector harmonics [19]. The use of spherical vector harmonics in global meteorology and aeronomy was suggested by Moses [20]. A similar approach was also used by Jones [11] to develop a general theory of atmospheric oscillations. We express the normal modes as a series expansion in terms of the spherical vector harmonics. By substituting the series into Laplace's tidal equations, we obtain an infinite, symmetric, homogeneous, pentadiagonal linear system of equations. The development of these equations is simplified by using the identities given by Swartrauber [28], which enable one to represent the Coriolis term as a linear combination of the spherical vector harmonics. The frequencies are computed as the eigenvalues of the matrix, and the coefficients in the expansion series are computed as the eigenvectors. The orthogonality of the modes corresponding to distinct frequencies follows from the symmetric property of the pentadiagonal matrix.

The zonal modes are treated separately in § 4, since their frequencies are not all distinct. All the rotational modes in this case are steady with zero frequencies and, therefore, are not necessarily orthogonal. Earlier Kasahara [14] formulated a method of solution for the zonal mode problem which requires the orthogonalization of the geostrophic modes. In this article, however, we propose a different method after an approach suggested by Shigehisa [23]. These new zonal modes are computed as a limit set. This approach is attractive from the point of view that these zonal modes exhibit the same fundamental characteristics that are displayed by the nonzonal modes. This method for computing the zonal modes is similar to that for the nonzonal modes, except that the coefficients in the expansion of the gravity and rotational modes are computed from two distinct symmetric *tridiagonal* systems of equations.

Certain limiting cases are presented in § 5, including the case of the nonrotating sphere and the calculation of the Haurwitz modes [9]. A computational program for the Hough vector functions is presented in § 6. The program consists of four subroutines for computing the frequencies, coefficients, Hough vector functions and related quantities.

**2. Laplace's tidal equations.** Small-amplitude motions of an incompressible, homogeneous, hydrostatic and inviscid fluid with free surface over a rotating sphere may be described by the following form of the shallow-water equations:

$$(2.1) \quad \frac{\partial u}{\partial t} - 2\Omega \sin \phi v = -\frac{g}{a \cos \phi} \frac{\partial h}{\partial \lambda},$$

$$(2.2) \quad \frac{\partial v}{\partial t} + 2\Omega \sin \phi u = -\frac{g}{a} \frac{\partial h}{\partial \phi},$$

$$(2.3) \quad \frac{\partial h}{\partial t} + \frac{h_0}{a \cos \phi} \left[ \frac{\partial u}{\partial \lambda} + \frac{\partial}{\partial \phi} (v \cos \phi) \right] = 0,$$

where  $\lambda$ ,  $\phi$  and  $t$  represent, respectively, longitude, latitude and time;  $u$ ,  $v$  the eastward and northward components of velocity;  $h$  the vertical displacement of the free surface from the mean height of the fluid  $h_0$ ; and  $\Omega$ ,  $a$  and  $g$  denote, respectively, the angular velocity, radius and the acceleration of gravity of the earth, all assumed to be constant. Equations (2.1) through (2.3) are often referred to as Laplace's tidal equations without the tide-generating terms [15].

The system of equations (2.1)–(2.3) appears as that of the horizontal structure equations when the linearized model of a compressible atmosphere at rest is resolved into its vertical and horizontal parts using the method of separation of variables. In this case, the mean free surface height  $h_0$  is interpreted as the constant of separation which links the vertical and horizontal operators and is known as the equivalent height [29].

For cases of forced oscillations, such as a tidal problem with a prescribed frequency, (2.1)–(2.3) may admit a negative value of  $h_0$  as an eigenvalue [1]. However, for the case of free oscillations, we are normally concerned with the solutions for positive values of  $h_0$ . Hence, in this article, we restrict ourselves to the case of positive  $h_0$ .

If we introduce the dimensionless variables

$$(2.4) \quad \tilde{u} = \frac{u}{\sqrt{gh_0}}, \quad \tilde{v} = \frac{v}{\sqrt{gh_0}}, \quad \tilde{h} = \frac{h}{h_0}, \quad \tilde{t} = 2\Omega t,$$

then (2.1)–(2.3) can be written in the form

$$(2.5) \quad \frac{\partial \mathbf{W}}{\partial \tilde{t}} + \mathbf{L}\mathbf{W} = 0,$$

where  $\mathbf{W}$  denotes the vector dependent variable

$$(2.6) \quad \mathbf{W} = (\tilde{u}, \tilde{v}, \tilde{h})^T.$$

Here  $\mathbf{L}$  is the linear differential matrix operator

$$(2.7) \quad \mathbf{L} = \begin{bmatrix} 0 & -\sin \phi & \frac{\gamma}{\cos \phi} \frac{\partial}{\partial \lambda} \\ \sin \phi & 0 & \gamma \frac{\partial}{\partial \phi} \\ \frac{\gamma}{\cos \phi} \frac{\partial}{\partial \lambda} & \frac{\gamma}{\cos \phi} \frac{\partial}{\partial \phi} [\cos \phi (\cdot)] & 0 \end{bmatrix}$$

in which

$$(2.8) \quad \gamma = \frac{\sqrt{gh_0}}{2a\Omega} (= \varepsilon^{-1/2})$$

is a single dimensionless constant that characterizes the nature of shallow-water flows. The related quantity,  $\varepsilon = \gamma^{-2}$ , is called Lamb's parameter [16].

Since (2.5) is a linear system with respect to  $\tilde{t}$ , the solution  $\mathbf{W}$  can be expressed as a linear combination of functions that have the form

$$(2.9) \quad \mathbf{W}(\lambda, \phi, \tilde{t}) = \mathbf{H}(\lambda, \phi) \exp(-i\sigma \tilde{t}),$$

where  $\mathbf{H}(\lambda, \phi)$  is the horizontal structure of the normal mode and  $\sigma$  is the corresponding dimensionless frequency scaled by  $2\Omega$ . Substituting (2.9) into (2.5), we see that the problem reduces to finding  $\mathbf{H}(\lambda, \phi)$  and  $\sigma$  such that

$$(2.10) \quad (\mathbf{L} - i\sigma)\mathbf{H}(\lambda, \phi) = 0.$$

Because (2.5) is linear with respect to longitude  $\lambda$ , the horizontal structure function  $\mathbf{H}(\lambda, \phi)$  can be expressed in the form

$$(2.11) \quad \mathbf{H}(\lambda, \phi) = \Theta(\phi) e^{im\lambda},$$

where  $m$  denotes the zonal wavenumber and  $\Theta(\phi)$  is the meridional modal function which depends only on latitude  $\phi$ . There are a number of meridional modal functions corresponding to each zonal wavenumber  $m$ . We use index  $l$  to denote the  $l$ th meridional normal mode. Hence, the horizontal structure of normal mode  $\mathbf{H}(\lambda, \phi)$  and the associated frequency  $\sigma$  depend on the zonal wavenumber  $m$  and meridional index  $l$  in addition to Lamb's parameter  $\epsilon$ .

In §§ 3 and 4, we present a method of calculating the frequencies  $\sigma$  and their associated horizontal structure function  $\mathbf{H}(\lambda, \phi)$ . In the case of  $m \geq 1$ , two different kinds of motion with distinct frequencies exist: eastward and westward propagating gravity-inertia waves (first kind) and westward propagating rotational waves of the Rossby-Haurwitz type (second kind). (For an example see [16].)

In the case of  $m = 0$ , the frequencies of gravity-inertia motion (first-kind) appear as pairs of positive and negative values with the same magnitudes. Although the meaning of eastward and westward propagations is lost in the case of  $m = 0$ , we shall use the term eastward (westward) to indicate positive (negative) frequency. On the other hand, the frequencies of the rotational motion (second-kind) as well as the gravity frequencies corresponding to the lowest meridional index  $l = 0$  are zero [16].

The dimensionless frequency  $\sigma$  is tabulated for zonal wavenumbers  $m = 0, 1, 2$  and  $3$ , in Tables 5 through 8, respectively, with the exception of certain entries in Table 5. Instead of tabulating zeros for all the rotational frequencies as well as the eastward gravity frequencies corresponding to the lowest meridional index  $l = 0$ , the asymptotic rate at which the frequencies go to zero is tabulated. The computation of the gravity frequencies and the asymptotic rates of the rotational frequencies for the case  $m = 0$  is quite different from the case  $m > 0$  and is discussed in detail in § 4.

All of the tabulated values are obtained from subroutine SIGMA described in § 5 and they are in agreement with those computed by Longuet-Higgins [16], except for the asymptotic rates which do not appear in [16]. The first column indicates the

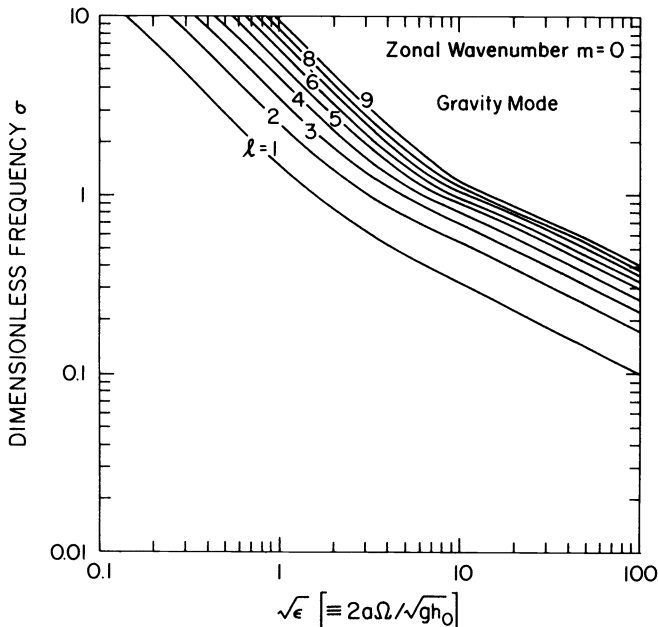


FIG. 1. Curves of dimensionless frequency  $|\sigma|$ , for gravity waves for zonal wavenumber  $m = 0$ , plotted against  $\sqrt{\epsilon}$ .

meridional modal index  $l$ , in which  $l=0$  denotes the lowest meridional mode. As shown later, the meridional modal functions  $\Theta(\phi)$  for the eastward and westward gravity modes are symmetric (antisymmetric) with respect to the equator for even (odd) meridional index  $l$ . On the other hand, for the rotational modes,  $\Theta(\phi)$  are symmetric (antisymmetric) for odd (even)  $l$ . Figure 1 shows the dimensionless frequencies  $\sigma$  of the gravity waves for the case of  $m = 0$  plotted against  $\sqrt{\epsilon}$  on the abscissa. The different curves correspond to the cases of meridional index  $l = 1$  through 9.

In Fig. 2(a) the dimensionless frequency  $\sigma$  for the case  $m = 1$  eastward propagating gravity waves is plotted versus  $\sqrt{\epsilon}$  on the abscissa. The mode  $l=0$  is symmetric and is referred to as a Kelvin wave. Figure 2(b) shows the same, but for the westward propagating gravity and rotational waves. The rotational  $l=0$  mode behaves like a rotational mode for small values of  $\epsilon$ , but behaves more like a gravity mode for large values of  $\epsilon$ . For this reason, this particular mode is referred to as a mixed Rossby gravity wave. The reader is referred to similar diagrams in Longuet-Higgins [16] for zonal wavenumbers up to  $m = 5$ .

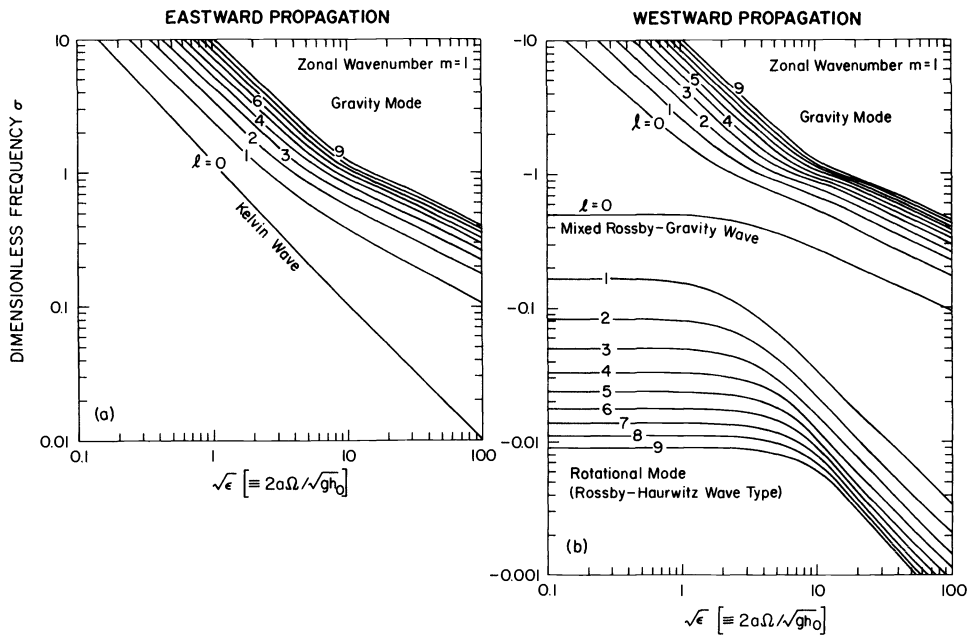


FIG. 2. (a) Curves of dimensionless frequency  $\sigma$ , for eastward propagating gravity waves for zonal wavenumber  $m = 1$  plotted against  $\sqrt{\epsilon}$  on the abscissa. (b) Curves of dimensionless frequency  $\sigma$ , for westward propagating gravity waves and rotational waves for zonal wavenumber  $m = 1$  plotted against  $\sqrt{\epsilon}$  on the abscissa.

We now show that for real  $\gamma$  or positive  $\epsilon$ ,  $L$  is skew Hermitian. Let  $u$  and  $v$  be arbitrary vector functions defined on the surface of the sphere. It can be shown by direct substitution followed by integration by parts that

$$(2.12) \quad \langle u, Lv \rangle = -\langle Lu, v \rangle,$$

where the innerproduct is defined by

$$(2.13) \quad \langle u, v \rangle = \int_0^{2\pi} \int_{-\pi/2}^{\pi/2} (u)^* v \cos \phi \, d\phi \, d\lambda$$

and  $(u)^*$  is the conjugate transpose of  $u$ .

The significance of (2.12) is that it can be used to determine that all the frequencies  $\sigma$  of  $\mathbf{L}$  are real and that any modes corresponding to distinct frequencies are orthogonal ([5], [18], [22] and [12]). Let  $\mathbf{H}_1$  and  $\mathbf{H}_2$  be modes that correspond to frequencies  $\sigma_1$  and  $\sigma_2$ , respectively. In (2.12) set  $u = \mathbf{H}_1$  and  $v = \mathbf{H}_2$ ; then

$$(2.14) \quad \langle \mathbf{H}_1, i\sigma_2 \mathbf{H}_2 \rangle = -\langle i\sigma_1 \mathbf{H}_1, \mathbf{H}_2 \rangle$$

or

$$(2.15) \quad (\sigma_1 - \bar{\sigma}_2) \langle \mathbf{H}_1, \mathbf{H}_2 \rangle = 0,$$

where the overbar denotes a complex conjugate.

If  $\mathbf{H}_1 = \mathbf{H}_2$ , then  $\langle \mathbf{H}_1, \mathbf{H}_2 \rangle$  is nonzero and therefore  $\sigma_1 = \bar{\sigma}_1$ . Thus, all the frequencies  $\sigma$  are real. On the other hand, if the frequencies are real and distinct, then  $\sigma_1 - \bar{\sigma}_2$  is nonzero, which implies that  $\langle \mathbf{H}_1, \mathbf{H}_2 \rangle = 0$ , meaning that the modes are orthogonal.

In the case of  $m > 0$ , the frequencies are distinct and hence orthogonal. However, for the case  $m = 0$ , the frequencies of the rotational modes are all zero and hence the modes are not necessarily orthogonal. However, in § 4 we derive an orthogonal set of rotational modes for  $m = 0$ . These modes are also orthogonal to the modes for  $m > 0$  and therefore all of the horizontal structure functions  $\mathbf{H}(\lambda, \phi)$  are orthogonal for  $m \geq 0$ . That is:

$$(2.16) \quad \frac{1}{2\pi} \int_0^{2\pi} \int_{-\pi/2}^{\pi/2} (\mathbf{H}_j)^* \mathbf{H}_k \cos \phi \, d\phi \, d\lambda = \delta_{jk},$$

where the right-hand side is unity if  $j = k$  and zero otherwise.

In this article, we introduce the subscript  $\alpha$ , which takes on the values of 1, 2 and 3 corresponding to the cases of eastward gravity, westward gravity and rotational mode, respectively. Hence, the normal modes are expressed by using three indices, namely  $m, l$  and  $\alpha$  in the form

$$(2.17) \quad \mathbf{H}_{l,\alpha}^m(\lambda, \phi) = \Theta_{l,\alpha}^m(\phi) e^{im\lambda},$$

where we designate  $\mathbf{H}_{l,\alpha}^m(\lambda, \phi)$  as the Hough vector harmonics and  $\Theta_{l,\alpha}^m(\phi)$  as the Hough vector functions which depend on latitude  $\phi$ . The Hough vector function has three components: zonal velocity  $U_{l,\alpha}^m$ , meridional velocity  $V_{l,\alpha}^m$  and height  $Z_{l,\alpha}^m$ , and is expressed by

$$(2.18) \quad \Theta_{l,\alpha}^m(\phi) = \begin{bmatrix} U_{l,\alpha}^m(\phi) \\ iV_{l,\alpha}^m(\phi) \\ Z_{l,\alpha}^m(\phi) \end{bmatrix}.$$

The factor  $i$  in front of  $V$  is introduced to account for a phase shift of  $\pi/2$ .

By substituting (2.17) into (2.16), the orthonormality of the Hough vector functions is seen:

$$(2.19) \quad \int_{-\pi/2}^{\pi/2} (\Theta_l^m)^* \Theta_l^m \cos \phi \, d\phi = \int_{-\pi/2}^{\pi/2} (U_l^m U_l^m + V_l^m V_l^m + Z_l^m Z_l^m) \cos \phi \, d\phi = \delta_{ll'},$$

where the subscript  $\alpha$  is suppressed.

**3. The vector harmonic analysis of the shallow-water equations.** In this section we will describe a method for computing the frequencies  $\sigma$  and corresponding Hough vector harmonics  $\mathbf{H}(\lambda, \phi)$ . The method consists of expanding the eigensolutions of (2.10) in terms of the spherical vector harmonics. When the expansion is substituted into the differential equations, an infinite, symmetric and pentadiagonal matrix is

obtained. The eigenvalues of this matrix correspond to the frequencies, and the eigenvectors correspond to the coefficients in the expansion of  $\mathbf{H}(\lambda, \phi)$ . We begin with a review of the spherical vector harmonics.

For  $n = 0, \dots$  and  $m = -n, \dots, n$ , the spherical vector harmonics are given by

$$(3.1) \quad y_{n,1}^m = \begin{bmatrix} \frac{im}{\cos \phi} P_n^m \\ \frac{dP_n^m}{d\phi} \\ 0 \end{bmatrix} \frac{e^{im\lambda}}{\sqrt{n(n+1)}}, \quad y_{n,2}^m = \begin{bmatrix} -\frac{dP_n^m}{d\phi} \\ \frac{im}{\cos \phi} P_n^m \\ 0 \end{bmatrix} \frac{e^{im\lambda}}{\sqrt{n(n+1)}},$$

$$y_{n,3}^m = \begin{bmatrix} 0 \\ 0 \\ P_n^m \end{bmatrix} e^{im\lambda},$$

where  $P_n^m$  are the normalized associated Legendre functions

$$(3.2) \quad P_n^m = \frac{1}{2^n n!} \sqrt{\frac{2n+1}{2} \frac{(n-m)!}{(n+m)!}} (\cos \phi)^m \frac{d^{m+n}}{dx^{m+n}} (x^2-1)^n, \quad x = \sin \phi.$$

Use of the normalized associated Legendre functions induces the normalization  $\langle y_{n,j}^m, y_{n,j}^m \rangle = 2\pi$  for  $j = 1, 2$  and  $3$ . The components of the vectors in (3.1) can be computed easily from the identities

$$(3.3) \quad \frac{dP_n^m}{d\phi} = \frac{1}{2} [\sqrt{(n-m)(n+m+1)} P_n^{m+1} - \sqrt{(n+m)(n-m+1)} P_n^{m-1}],$$

$$(3.4) \quad \frac{m}{\cos \phi} P_n^m = \frac{1}{2} \sqrt{\frac{2n+1}{2n-1}} [\sqrt{(n+m)(n+m-1)} P_{n-1}^{m-1} + \sqrt{(n-m)(n-m-1)} P_{n-1}^{m+1}].$$

Apart from a scale factor, the vectors (3.1) correspond to the spherical vector harmonics  $B_{mn}, C_{mn}$  and  $P_{mn}$ , respectively, given in [19]. They form a complete set under the inner product (2.13) for vector functions defined on the surface of the sphere [26], [27].

We proceed now to list the identities for the spherical vector harmonics that will be used in the analysis of  $\mathbf{L}$ . Let  $U^T = (u, v, h)$  be an arbitrary vector and define

$$(3.5) \quad \operatorname{div} U = \frac{1}{\cos \phi} \left[ \frac{\partial u}{\partial \lambda} + \frac{\partial}{\partial \phi} (\cos \phi v) \right],$$

$$(3.6) \quad \operatorname{rot} U = \frac{1}{\cos \phi} \left[ \frac{\partial v}{\partial \lambda} - \frac{\partial}{\partial \phi} (\cos \phi u) \right].$$

Then it is known that

$$(3.7) \quad \operatorname{div} y_{n,1}^m = -\sqrt{n(n+1)} Y_n^m, \quad \operatorname{div} y_{n,2}^m = 0, \quad \operatorname{div} y_{n,3}^m = 0,$$

and

$$(3.8) \quad \operatorname{rot} y_{n,1}^m = 0, \quad \operatorname{rot} y_{n,2}^m = \sqrt{n(n+1)} Y_n^m, \quad \operatorname{rot} y_{n,3}^m = 0,$$

where  $Y_n^m$  are the scalar spherical harmonics given by  $Y_n^m = P_n^m e^{im\lambda}$ . Furthermore, using the identities in [28], it can be shown that

$$(3.9) \quad \sin \phi y_{n,1}^m = p_n^m y_{n-1,1}^m + p_{n+1}^m y_{n+1,1}^m + iq_n^m y_{n,2}^m,$$

$$(3.10) \quad \sin \phi y_{n,2}^m = p_n^m y_{n-1,2}^m + p_{n+1}^m y_{n+1,2}^m - iq_n^m y_{n,1}^m,$$

where

$$(3.11) \quad p_n^m = \sqrt{\frac{(n-1)(n+1)(n-m)(n+m)}{n^2(2n-1)(2n+1)}}, \quad q_n^m = \frac{m}{n(n+1)}.$$

Let  $f$  be an arbitrary scalar function on the sphere and define

$$(3.12) \quad \nabla f = \begin{bmatrix} 1 & \frac{df}{d\lambda} \\ \cos \phi & \frac{df}{d\phi} \end{bmatrix}.$$

Then

$$(3.13) \quad \nabla Y_n^m = \sqrt{n(n+1)} y_{n,1}^m.$$

Using these identities we can express  $\mathbf{L}y_{n,j}^m$  as a linear combination of the  $y_{n,j}^m$  for  $j = 1, 2$  and  $3$ . From (2.7) and (3.1),

$$(3.14) \quad \mathbf{L}y_{n,1}^m = \begin{bmatrix} -\sin \phi \frac{dP_n^m}{d\phi} \\ \sin \phi \frac{im}{\cos \phi} P_n^m \\ \gamma \sqrt{n(n+1)} \operatorname{div} y_{n,1}^m \end{bmatrix} \frac{e^{im\lambda}}{\sqrt{n(n+1)}}.$$

Using (3.1) and (3.7), we can rewrite (3.14) as

$$(3.15) \quad \mathbf{L}y_{n,1}^m = \sin \phi y_{n,2}^m - \gamma \sqrt{n(n+1)} y_{n,3}^m.$$

Substituting (3.10) into the first term on the right, we obtain

$$(3.16) \quad \mathbf{L}y_{n,1}^m = p_n^m y_{n-1,2}^m + p_{n+1}^m y_{n+1,2}^m - iq_n^m y_{n,1}^m - \gamma \sqrt{n(n+1)} y_{n,3}^m.$$

Next, from (2.7) and (3.1),

$$(3.17) \quad \mathbf{L}y_{n,2}^m = \begin{bmatrix} -\sin \phi \frac{im}{\cos \phi} P_n^m \\ -\sin \phi \frac{dP_n^m}{d\phi} \\ 0 \end{bmatrix} \frac{e^{im\lambda}}{\sqrt{n(n+1)}}$$

or

$$(3.18) \quad \mathbf{L}y_{n,2}^m = -\sin \phi y_{n,1}^m.$$

But from (3.9),

$$(3.19) \quad \mathbf{L}y_{n,2}^m = -(p_n^m y_{n-1,1}^m + p_{n+1}^m y_{n+1,1}^m + iq_n^m y_{n,2}^m).$$

From (2.7) and (3.1),

$$(3.20) \quad \mathbf{L}y_{n,3}^m = \begin{bmatrix} \gamma \frac{im}{\cos \phi} P_n^m \\ \gamma \frac{dP_n^m}{d\phi} \\ 0 \end{bmatrix} e^{im\lambda}$$

or

$$(3.21) \quad \mathbf{L}y_{n,3}^m = \gamma\sqrt{n(n+1)}y_{n,1}^m.$$

Equations (3.21), (3.16) and (3.19) provide the identities that are necessary for the analysis of the operator  $\mathbf{L}$ . If we assume the expansion

$$(3.22) \quad \mathbf{H}(\lambda, \phi) = \Theta_l^m e^{im\lambda} = \sum_n (iA_n^m y_{n,1}^m + B_n^m y_{n,2}^m - C_n^m y_{n,3}^m),$$

and substitute (3.22) into (2.10), then

$$(3.23) \quad -i\sigma \sum (iA_n^m y_{n,1}^m + B_n^m y_{n,2}^m - C_n^m y_{n,3}^m) + \sum (iA_n^m \mathbf{L}y_{n,1}^m + B_n^m \mathbf{L}y_{n,2}^m - C_n^m \mathbf{L}y_{n,3}^m) = 0.$$

Substituting (3.21), (3.16) and (3.19) into (3.23) and equating coefficients of  $y_{n,j}^m$  for  $j=1, 2$  and  $3$ , respectively, we obtain

$$(3.24) \quad (\sigma + q_n^m)A_n^m = r_n C_n^m + p_n^m B_{n-1}^m + p_{n+1}^m B_{n+1}^m,$$

$$(3.25) \quad (\sigma + q_n^m)B_n^m = p_n^m A_{n-1}^m + p_{n+1}^m A_{n+1}^m,$$

$$(3.26) \quad \sigma C_n^m = r_n A_n^m,$$

where  $r_n = \gamma\sqrt{n(n+1)}$ .

This system of equations can be divided into two subsystems. The first subsystem consists of the equations that are satisfied by the coefficients  $A_n^m, B_{n+1}^m$  and  $C_n^m$  for  $n = m, m+2, m+4, \dots$ . For this case, the free surface height and zonal velocity component are symmetric with respect to the equator and the meridional velocity component is antisymmetric. We call this case *symmetric*. Let  $\mathbf{X}$  be the vector

$$(3.27) \quad \mathbf{X} = (C_m^m, A_m^m, B_{m+1}^m, C_{m+2}^m, A_{m+2}^m, B_{m+3}^m, \dots)^T$$

and  $\mathbf{A}$  be the matrix

$$(3.28) \quad \mathbf{A} = \begin{bmatrix} 0 & r_m & 0 & & & & \\ r_m & -q_m^m & p_{m+1}^m & 0 & & & \\ 0 & p_{m+1}^m & -q_{m+1}^m & 0 & p_{m+2}^m & & \\ & 0 & 0 & 0 & r_{m+2} & 0 & \\ & & p_{m+2}^m & r_{m+2} & -q_{m+2}^m & p_{m+3}^m & \\ & & & 0 & p_{m+3}^m & -q_{m+3}^m & \\ & & & & & & \ddots \end{bmatrix}.$$

Then the symmetric case is written as

$$(3.29) \quad \mathbf{A}\mathbf{X} = \sigma\mathbf{X}.$$

The second subsystem consists of the equations from (3.24) through (3.26) that are satisfied by the coefficients  $A_{n+1}^m, B_n^m$  and  $C_{n+1}^m$  for  $n = m, m+2, m+4, \dots$ . For this case, the free surface height and zonal velocity component are antisymmetric with respect to the equator and the meridional velocity component is symmetric. We call this case *antisymmetric*. Let  $\mathbf{Y}$  be the vector

$$(3.30) \quad \mathbf{Y} = (B_m^m, C_{m+1}^m, A_{m+1}^m, B_{m+2}^m, C_{m+3}^m, A_{m+3}^m, \dots)^T$$



and **B** be the matrix

$$(3.31) \quad \mathbf{B} = \begin{bmatrix} -q_m^m & 0 & p_{m+1}^m & & & & & & \\ 0 & 0 & r_{m+1} & 0 & & & & & \\ p_{m+1}^m & r_{m+1} & -q_{m+1}^m & p_{m+2}^m & 0 & & & & \\ & 0 & p_{m+2}^m & -q_{m+2}^m & 0 & p_{m+3}^m & & & \\ & & 0 & 0 & 0 & r_{m+3} & & & \\ & & & p_{m+3}^m & r_{m+3} & -q_{m+3}^m & & & \\ & & & & & & \dots & & \end{bmatrix}.$$

Then the antisymmetric case is written as

$$(3.32) \quad \mathbf{B}\mathbf{Y} = \sigma\mathbf{Y}.$$

The problem thus reduces to computing the eigenvalues  $\sigma$  and corresponding eigenvectors of the real symmetric pentadiagonal matrices **A** and **B**. We note that the eigenvalues must be real, which is in agreement with the results that were obtained in the previous section.

We can restrict our attention to the case  $m \geq 0$ , for if  $A_n^m, B_n^m, C_n^m$  and  $\sigma_n^m$  comprise an eigensolution of (3.24)-(3.26), then  $-A_n^{-m}, B_n^{-m}, C_n^{-m}$  and  $-\sigma_n^{-m}$  also comprise an eigensolution of (3.24)-(3.26) with  $m$  replaced by  $-m$ . This implies that  $\mathbf{H}_{l,\alpha}^{-m} = (-1)^m \bar{\mathbf{H}}_{l,\alpha}^m$  since  $y_{nj}^{-m} = (-1)^m \bar{y}_{nj}^m$  for  $j = 1, 2$  and  $3$ .

Once the coefficients are computed,  $\mathbf{H}(\lambda, \phi)$  can be determined from (3.22). In addition, several related quantities can also be determined. From (3.7), (3.8) and (3.22)

$$(3.33) \quad \text{div } \mathbf{H}(\lambda, \phi) = -\sum i\sqrt{n(n+1)} A_n^m Y_n^m,$$

$$(3.34) \quad \text{rot } \mathbf{H}(\lambda, \phi) = \sum \sqrt{n(n+1)} B_n^m Y_n^m.$$

The velocity potential  $\Phi$  and stream function  $\Psi$  are defined by  $\nabla^2\Phi = \text{div } \mathbf{H}(\lambda, \phi)$  and  $\nabla^2\Psi = \text{rot } \mathbf{H}(\lambda, \phi)$ . Therefore,

$$(3.35) \quad \Phi = \sum \frac{iA_n^m}{\sqrt{n(n+1)}} Y_n^m,$$

$$(3.36) \quad \Psi = -\sum \frac{B_n^m}{\sqrt{n(n+1)}} Y_n^m,$$

using the relationship  $\nabla^2 Y_n^m = -n(n+1) Y_n^m$ .

**4. The zonal modes.** The case  $m = 0$  is unique since the frequencies of the gravity waves (first kind) appear as pairs of positive and negative values of the same magnitudes and the frequencies of the rotational motions (second kind) are all zero [16]. The meaning of eastward and westward propagation is lost in this case. The meridional structure functions of the gravity modes  $\Theta_l^0$  are symmetric (antisymmetric) with respect to the equator for even (odd) meridional index  $l$ . Eastward and westward gravity modes that correspond to the same meridional index are the complex conjugates of one another. Also, because the frequencies of the gravity waves are all distinct, the  $\Theta_l^0$  are orthogonal in the sense of (2.19). However, the frequencies of the rotational modes are all zero, and therefore not distinct, so that the modes are not necessarily orthogonal. In this section we will present a computational method for determining an orthogonal set of rotational modes.

We now proceed to the analysis of the zonal normal modes. The notation can be simplified by deleting the superscript  $m = 0$ . To this end, we define

$$(4.1) \quad A_n = A_n^0, \quad B_n = B_n^0, \quad C_n = C_n^0,$$

$$(4.2) \quad p_n = p_n^0 = \sqrt{\frac{(n-1)(n+1)}{(2n-1)(2n+1)}} \quad \text{and} \quad q_n^0 = 0.$$

For  $m = 0$  the system (3.24) through (3.26) has the form

$$(4.3) \quad \sigma A_n = r_n C_n + p_n B_{n-1} + p_{n+1} B_{n+1},$$

$$(4.4) \quad \sigma B_n = p_n A_{n-1} + p_{n+1} A_{n+1},$$

$$(4.5) \quad \sigma C_n = r_n A_n.$$

If  $\sigma$  is a frequency that corresponds to a mode with coefficients  $A_n$ ,  $B_n$  and  $C_n$ , then from (4.3) through (4.5),  $-\sigma$  is a frequency that corresponds to the mode with coefficients  $A_n$ ,  $B_n$  and  $C_n$ . Hence the nonzero frequencies occur in pairs. We now show that these frequencies can be determined as the eigenvalues of two symmetric tridiagonal systems rather than from the pentadiagonal systems (3.29) and (3.32). If we multiply (4.3) by  $\sigma$  and eliminate  $\sigma B_{n-1}$ ,  $\sigma C_n$  and  $\sigma B_{n+1}$  from the result by using (4.4) and (4.5), we obtain

$$(4.6) \quad p_{n-1} p_n A_{n-2} + (r_n^2 + p_n^2 + p_{n+1}^2 - \sigma^2) A_n + p_{n+1} p_{n+2} A_{n+2} = 0.$$

The eigenvalues  $\sigma^2$  of this system can be computed efficiently from two distinct tridiagonal systems that are obtained by separating (4.6) into systems with even and odd subscripts. The frequencies can then be computed as plus and minus the square root of the eigenvalues. Consider first the symmetric case which corresponds to (4.6) with  $n = 0, 2, 4, \dots$ . Let  $\mathbf{U}$  be the vector

$$(4.7) \quad \mathbf{U} = (A_0, A_2, A_4, \dots)^T$$

and  $\mathbf{C}$  be the matrix

$$(4.8) \quad \mathbf{C} = \begin{bmatrix} r_0^2 + p_1^2 & p_1 p_2 & & & & & \\ p_1 p_2 & r_2^2 + p_2^2 + p_3^2 & p_3 p_4 & & & & \\ & p_3 p_4 & r_4^2 + p_4^2 + p_5^2 & p_5 p_6 & & & \\ & & p_5 p_6 & & \dots & & \\ & & & & & \dots & \\ & & & & & & \dots \end{bmatrix}.$$

Then the symmetric case is written as

$$(4.9) \quad \mathbf{C}\mathbf{U} = \sigma^2 \mathbf{U}.$$

The antisymmetric case consists of the equations from (4.6) corresponding to  $n = 1, 3, 5, \dots$ . Let  $\mathbf{V}$  be the vector

$$(4.10) \quad \mathbf{V} = (A_1, A_3, A_5, \dots)^T$$

and  $\mathbf{D}$  be the matrix

$$(4.11) \quad \mathbf{D} = \begin{bmatrix} r_1^2 + p_1^2 + p_2^2 & p_2 p_3 & & & & & \\ p_2 p_3 & r_3^2 + p_3^2 + p_4^2 & p_4 p_5 & & & & \\ & p_4 p_5 & r_5^2 + p_5^2 + p_6^2 & p_6 p_7 & & & \\ & & p_6 p_7 & & \dots & & \\ & & & & & \dots & \\ & & & & & & \dots \end{bmatrix}.$$

Then the antisymmetric case is written as

$$(4.12) \quad \mathbf{D}\mathbf{V} = \sigma^2\mathbf{V}.$$

Once the frequencies have been determined as plus and minus the square root of the eigenvalues of  $\mathbf{C}$  and  $\mathbf{D}$ , the coefficients of the mode corresponding to  $\sigma$  can be determined by first computing the coefficients  $A_n$  as the eigenvectors of  $\mathbf{C}$  and  $\mathbf{D}$  and then  $B_n$  and  $C_n$  from (4.4) and (4.5). As mentioned above, the coefficients corresponding to  $-\sigma$  are given by  $-A_n$ ,  $B_n$  and  $C_n$ .

There are two modes with zero frequency which are classified as gravity modes. The first is obtained by noting that  $\mathbf{U}^T = (1, 0, 0, \dots)$  is an eigenvector of  $\mathbf{C}$  which corresponds to the eigenvalue  $\sigma^2 = 0$ . Therefore the first of these modes is identified by the indices  $n, \alpha = 0, 2$  for which  $A_0 = 1$  and all other coefficients are zero. Since  $y_{0,1}^0$  is identically zero, this mode is also identically zero, which corresponds to solid rotation with respect to the nonrotating coordinate system. The second mode will be identified by the indices  $n, \alpha = 0, 1$ . It will be determined at the end of this section. The remaining zonal modes are steady with  $\sigma = 0$ .

The case  $m = 0$  and  $\sigma = 0$  is quite different from the previous cases, since there are an infinite number of modes that correspond to the same frequency. If  $\sigma = 0$ , then from (4.4) and (4.5) it can be seen that  $A_n = 0$  and hence the flow is strictly rotational. In addition, from (4.3),

$$(4.13) \quad r_n C_n + p_n B_{n-1} + p_{n+1} B_{n+1} = 0.$$

However, this equation alone does not determine a unique mode. For example, if we let  $N$  be any positive integer, then we can determine a sequence of coefficients whose elements are solutions of (4.13). If we set  $B_n = 0$  for all  $n$  except  $B_N = 1$ , then (4.13) is satisfied if  $C_{N-1} = -p_N/r_{N-1}$ ,  $C_{N+1} = -p_{N+1}/r_{N+1}$  and  $C_n = 0$  for all other  $n$ . This method of solution was adopted in [14]. However, we note that the resulting modes are not orthogonal and although they could be orthogonalized using the Gram-Schmidt process, a more satisfactory approach has been suggested by Shigehisa [23].

In this new approach, the modes for  $m = 0$  are determined as the limit of modes for which  $m \geq 0$ . This seems like a reasonable approach since the limit of an orthogonal set can also be expected to be orthogonal. However, it is interesting to note that the limit is taken through noninteger (infinitesimal) values of  $m$  for which the normal modes have not been defined. Nevertheless, the approach does work in the sense that an orthogonal set is obtained, and although any linear combination of the resulting modes is also a mode, this particular set exhibits some basic characteristics that are shared with the  $m \neq 0$  modes.

Near  $m = 0$  we assume that the coefficients and  $\sigma$  have the following asymptotic forms.

$$(4.14) \quad \begin{aligned} A_n^m &= A_n m + O(m^2), \\ B_n^m &= B_n + O(m), \\ C_n^m &= C_n + O(m), \\ \sigma &= \sigma_a m + O(m^2). \end{aligned}$$

Substituting these forms into (3.24) and equating coefficients of  $m^0 = 1$ , we obtain (4.13) as expected. If we then substitute the forms (4.14) into (3.25) and (3.26), we obtain the following additional equations which will enable us to compute the desired

modes

$$(4.15) \quad \left( \sigma_a + \frac{1}{n(n+1)} \right) B_n = p_n A_{n-1} + p_{n+1} A_{n+1},$$

$$(4.16) \quad \sigma_a C_n = r_n A_n.$$

Eliminating  $A_n$  and  $C_n$  from (4.13), (4.15) and (4.16) we obtain

$$(4.17) \quad \frac{p_n p_{n-1}}{r_{n-1}^2} B_{n-2} + \left[ 1 + \left( \frac{p_n}{r_{n-1}} \right)^2 + \left( \frac{p_{n+1}}{r_{n+1}} \right)^2 + \frac{1}{n(n+1)\sigma_a} \right] B_n + \frac{p_{n+1} p_{n+2}}{r_{n+1}^2} B_{n+2} = 0.$$

If we define

$$(4.18) \quad \begin{aligned} \tilde{\sigma} &= -\frac{1}{\sigma_a}, \\ \tilde{B}_n &= \frac{1}{\sqrt{n(n+1)}} B_n, \\ d_n &= \sqrt{n(n-1)} \frac{p_n}{r_n} = \frac{(n-1)}{\gamma \sqrt{(2n-1)(2n+1)}}, \\ e_n &= \sqrt{(n+1)(n+2)} \frac{p_{n+1}}{r_n} = \frac{(n+2)}{\gamma \sqrt{(2n+1)(2n+3)}}, \end{aligned}$$

then (4.17) takes the form

$$(4.19) \quad e_{n-1} d_{n-1} \tilde{B}_{n-2} + [n(n+1) + e_{n-1}^2 + d_{n+1}^2 - \tilde{\sigma}] \tilde{B}_n + d_{n+1} e_{n+1} \tilde{B}_{n+2} = 0.$$

From this equation we obtain two independent symmetric tridiagonal equations for  $\tilde{B}_n$  and  $\tilde{\sigma}$  from which the desired modes can be determined. Consider first the symmetric case which corresponds to (4.19) with  $n = -1, 1, 3, \dots$ . Let  $\mathbf{S}$  be the vector

$$(4.20) \quad \mathbf{S} = (\tilde{B}_{-1}, \tilde{B}_1, \tilde{B}_3, \dots)^T$$

and  $\mathbf{E}$  be the matrix

$$(4.21) \quad \mathbf{E} = \begin{bmatrix} d_0^2 & d_0 e_0 & & & \\ d_0 e_0 & 2 + e_0^2 + d_2^2 & d_2 e_2 & & \\ & d_2 e_2 & 12 + e_2^2 + d_4^2 & d_4 e_4 & \\ & & d_4 e_4 & & \ddots \\ & & & & \ddots \end{bmatrix}.$$

Then the symmetric case is written as

$$(4.22) \quad \mathbf{E}\mathbf{S} = \tilde{\sigma}\mathbf{S}.$$

The antisymmetric case consists of the equations (4.19) that correspond to  $n = 2, 4, 6, \dots$ . Let  $\mathbf{T}$  be the vector

$$(4.23) \quad \mathbf{T} = (\tilde{B}_2, \tilde{B}_4, \tilde{B}_6, \dots)^T$$



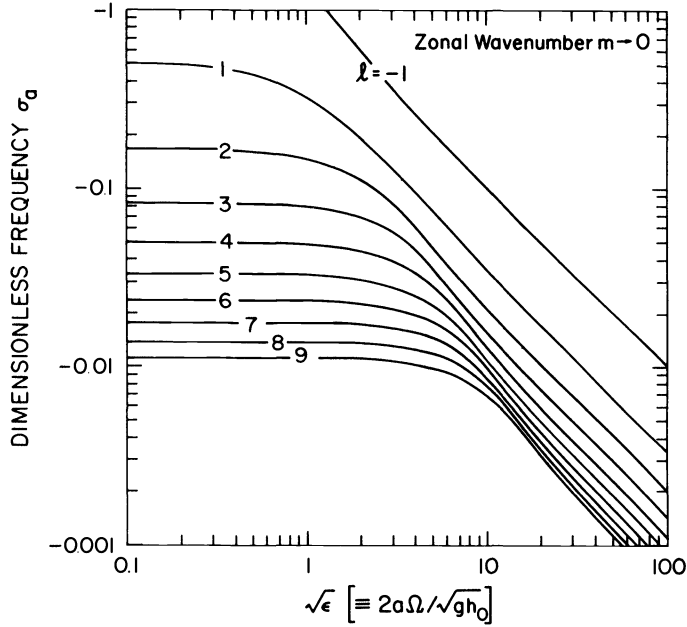


FIG. 3. Curves of asymptotic dimensionless frequency  $\sigma_a$ , which is defined in (4.14) in the case of  $m \rightarrow 0$  and calculated from  $-1/\tilde{\sigma}$ , plotted against  $\sqrt{\epsilon}$ .

From (4.18) and (4.26)

$$(4.28) \quad \langle \mathbf{H}_1, \mathbf{H}_2 \rangle = \sum_{\substack{n=1 \\ \text{odd } n}}^{\infty} n(n+1) \widetilde{\mathbf{B}}_1 \widetilde{\mathbf{B}}_2 + \sum_{\substack{n=0 \\ \text{even } n}}^{\infty} [\widetilde{d}_n \widetilde{\mathbf{B}}_1 + e_n \widetilde{\mathbf{B}}_1] [d_n \widetilde{\mathbf{B}}_2 + e_n \widetilde{\mathbf{B}}_2].$$

The conjugate of  $d_n$  is used on the right-hand side of (4.28) since  $d_0$  is strictly imaginary. Expanding the right side of (4.28) and combining terms, we obtain

$$(4.29) \quad \begin{aligned} \langle \mathbf{H}_1, \mathbf{H}_2 \rangle = & (|d_0|^2 \widetilde{\mathbf{B}}_{-1} + d_0 e_0 \widetilde{\mathbf{B}}_1) \widetilde{\mathbf{B}}_{-2} \\ & + \sum_{\substack{n=1 \\ \text{odd } n}}^{\infty} \{ \widetilde{d}_{n-1} e_{n-1} \widetilde{\mathbf{B}}_{n-2} \\ & + [n(n+1) + e_{n-1}^2 + d_{n+1}^2] \widetilde{\mathbf{B}}_n + d_{n+1} e_{n+1} \widetilde{\mathbf{B}}_{n+2} \} \widetilde{\mathbf{B}}_2 \end{aligned}$$

or, using (4.19),

$$(4.30) \quad \langle \mathbf{H}_1, \mathbf{H}_2 \rangle = \tilde{\sigma}_1 \left( -\widetilde{\mathbf{B}}_{-1} \widetilde{\mathbf{B}}_{-2} + \sum_{\substack{n=1 \\ \text{odd } n}}^{\infty} \widetilde{\mathbf{B}}_n \widetilde{\mathbf{B}}_n \right).$$

If  $\mathbf{H}_1 = \mathbf{H}_2$ , then from (4.27) and (4.30) it can be observed that  $\tilde{\sigma}_1$  as well as all eigenvalues of  $\mathbf{E}$  must be real. Furthermore,  $\mathbf{E}$  is similar to a real matrix under a diagonal similarity transform  $\text{diag}(i, 1, \dots, 1)$ . Therefore all components of the eigenvectors of  $\mathbf{E}$  are real with the exception of the first component, which is strictly imaginary. Hence (4.30) can be written

$$(4.31) \quad \langle \mathbf{H}_1, \mathbf{H}_2 \rangle = \tilde{\sigma}_1 \sum_{\substack{n=-1 \\ \text{odd } n}}^{\infty} \widetilde{\mathbf{B}}_n \widetilde{\mathbf{B}}_n.$$

Proceeding in a similar manner we can obtain

$$(4.32) \quad \langle \mathbf{H}_1, \mathbf{H}_2 \rangle = \tilde{\sigma}_2 \sum_{\substack{n=-1 \\ \text{odd } n}}^{\infty} \widetilde{B1}_n \widetilde{B2}_n$$

or

$$(4.33) \quad (\tilde{\sigma}_2 - \tilde{\sigma}_1) \sum_{\substack{n=-1 \\ \text{odd } n}}^{\infty} \widetilde{B1}_n \widetilde{B2}_n = 0,$$

which completes the proof that modes corresponding to distinct eigenvalues are orthogonal. The proof for the antisymmetric case is simpler than the symmetric case since all of the  $d_n$  are real and therefore  $\mathbf{F}$  is a real symmetric matrix with orthogonal eigenvectors.

**5. The limiting case  $\gamma \rightarrow \infty$  and the Haurwitz waves.** From (2.8) it can be seen that if the sphere is not rotating ( $\Omega = 0$ ) or if the equivalent height is infinite ( $h_0 = \infty$ ), then  $\gamma = \infty$ . Recall from (2.4) that the Hough vector functions define modes in the transformed variables  $\tilde{u}$ ,  $\tilde{v}$  and  $\tilde{h}$ . The purpose of this transform was to reduce the number of physical parameters in the shallow-water equations to the single dimensionless parameter  $\gamma$ . For the dependent variables  $u$ ,  $v$  and  $h$  the transform is not a function of  $\Omega$  and is therefore applicable to the limiting case  $\Omega = 0$ . However, in order to apply the transform, the limit of the Hough vector functions must be determined. In the first part of this section we will determine the limit of the Hough vector functions as  $\gamma$  tends to infinity.

The case in which  $h_0$  tends to infinity is somewhat different since the transform (2.4) is no longer applicable. This case will be treated later in this section by computing the limit of the modes in the untransformed variables  $u$ ,  $v$  and  $h$ . These limiting modes are called the Haurwitz modes [9]. We begin by first computing the limit of the Hough vector functions.

If we divide (3.24) and (3.26) by  $\gamma$  and then let  $\gamma$  go to infinity, we find that  $C_n^m = A_n^m = 0$ . Furthermore, (3.25) is satisfied if we set  $\sigma = -q_n^m$  and  $B_n^m = 1$ . Therefore the limit of the rotational mode as  $\gamma$  goes to infinity is

$$(5.1) \quad \mathbf{H}_{n,3}^m = y_{n,2}^m \quad \text{for } \sigma = -\frac{m}{n(n+1)}.$$

In order to obtain this limit we assumed that the frequencies  $\sigma$  were finite. In order to obtain the limit of the gravity waves we must assume that the limit of the corresponding frequencies is infinite. If (3.24)–(3.26) are divided by  $\gamma$ , and we define

$$(5.2) \quad \tau = \lim_{\Omega \rightarrow 0} \frac{\sigma}{\gamma},$$

then we obtain

$$(5.3) \quad \tau A_n^m = \sqrt{n(n+1)} C_n^m,$$

$$(5.4) \quad \tau B_n^m = 0,$$

$$(5.5) \quad \tau C_n^m = \sqrt{n(n+1)} A_n^m.$$

For nonzero  $\tau$ , (5.4) implies that  $B_n^m = 0$ . The remaining equations (5.3) and (5.5) have the solution  $A_n^m = 1/\sqrt{2}$ ,  $C_n^m = 1/\sqrt{2}$  corresponding to  $\tau = \sqrt{n(n+1)}$  and  $A_n^m = 1/\sqrt{2}$ ,  $C_n^m = -1/\sqrt{2}$  corresponding to  $\tau = -\sqrt{n(n+1)}$ . The limit of the gravity waves is therefore

$$(5.6) \quad \mathbf{H}_{n,1}^m = (iy_{n,1}^m - y_{n,3}^m)/\sqrt{2} \quad \text{for } \tau = \sqrt{n(n+1)},$$

$$(5.7) \quad \mathbf{H}_{n,2}^m = (iy_{n,1}^m + y_{n,3}^m)/\sqrt{2} \quad \text{for } \tau = -\sqrt{n(n+1)}.$$

Equations (5.1), (5.6) and (5.7) define the limit of the Hough vector functions as  $\gamma$  tends to infinity. It is of interest to note although the frequencies of the gravity waves tend to infinity in the transformed variable  $\tilde{t}$ , they are finite in the untransformed variable  $t$  as  $\Omega$  tends to zero.

We proceed now to compute the Haurwitz waves [9] as the limit of the modes in the untransformed variables  $u, v$  and  $h$ , as  $h_0$  goes to infinity. Define

$$(5.8) \quad \hat{A}_n^m = \sqrt{gh_0}A_n^m, \quad \hat{B}_n^m = \sqrt{gh_0}B_n^m, \quad \hat{C}_n^m = h_0C_n^m$$

as the coefficients in the expansion of the modes in the untransformed variables  $u, v$  and  $h$ . Then substituting into (3.24)-(3.26) we obtain

$$(5.9) \quad (\sigma + q_n^m)\hat{A}_n^m = \frac{g\sqrt{n(n+1)}}{2a\Omega} \hat{C}_n^m + p_n^m \hat{B}_{n-1}^m + p_{n+1}^m \hat{B}_{n+1}^m,$$

$$(5.10) \quad (\sigma + q_n^m)\hat{B}_n^m = p_n^m \hat{A}_{n-1}^m + p_{n+1}^m \hat{A}_{n+1}^m,$$

$$(5.11) \quad \frac{\sigma}{h_0} \hat{C}_n^m = \frac{\sqrt{n(n+1)}}{2a\Omega} \hat{A}_n^m.$$

From the last equation we see that the limit of  $\hat{A}_n^m$  is zero as  $h_0$  tends to infinity, which implies that the right-hand side of (5.10) is zero. Therefore, (5.10) has the solution  $\sigma = -q_n^m$  and  $\hat{B}_n^m = 1$ . From (5.9) we determine that

$$(5.12) \quad \hat{C}_{n-1}^m = -\frac{2a\Omega}{g\sqrt{n(n-1)}} p_n^m, \quad \hat{C}_{n+1}^m = -\frac{2a\Omega}{g\sqrt{(n+1)(n+2)}} p_{n+1}^m.$$

These coefficients define the Haurwitz waves

$$(5.13) \quad \mathbf{B}_n^m = y_{n,2}^m + \frac{2a\Omega}{g} \left( \frac{p_n^m}{\sqrt{n(n-1)}} y_{n-1,3}^m + \frac{p_{n+1}^m}{\sqrt{(n+1)(n+2)}} y_{n+1,3}^m \right)$$

with corresponding frequencies  $\sigma = -m/[n(n+1)]$ .

**6. A computational facility.** In this section we will describe a set of programs for computing the Hough vector functions and the computational techniques that are used in each of the programs. In addition to their application in the analysis of global meteorological data and in the prediction of global atmospheric motion, the Hough functions have been used to test the performance of computational methods that have been used to solve the global primitive equation model [2]. Since the Hough vector harmonics are the exact solution of the linearized shallow-water equations with respect to the basic state at rest, they can be used to verify the numerical solutions of the linearized part of the global shallow-water equations. The software that is described in this section can be used to assist in these applications.

The programs can compute the Hough vector functions for the limit case  $\gamma = \infty$  or  $\varepsilon = 0$  and also for the zonal modes corresponding to  $m = 0$ . There are four programs in the set: subroutine SIGMA, which computes the frequencies of the normal modes; subroutine ABCOEF, which computes the coefficients  $A_n^m, B_n^m$  and  $C_n^m$  in the expansion of the Hough vector harmonics in terms of the spherical vector harmonics; subroutine UVH, which uses the coefficients computed by ABCOEF in order to compute the Hough vector functions  $\Theta_n^m(\phi)$ ; subroutine UVHDER, in which a number of derived



quantities are computed, including vorticity, divergence, stream function and velocity potential. We begin with the calling sequence of subroutine SIGMA (Table 1) and a description of its parameters, followed by a discussion of the computational methods.

Subroutine SIGMA computes the frequencies as the eigenvalues of the matrices **A**, **B**, **C**, **D**, **E** or **F**, depending on the case. For computational purposes these matrices must be truncated to some finite order. For **A** and **B** the frequencies are determined in triplets corresponding to eastward gravity, westward gravity and rotational modes and therefore the order is selected as a multiple of 3, say  $3N$ . Furthermore, the value of  $N$  must be sufficiently large that the computed frequencies will be accurate approximations to those of the infinite matrix. In practice it was determined that this was the case if the matrix was truncated at a point where  $r_n = \sqrt{n(n+1)}/\varepsilon$  was at least as large as the other coefficients,  $p_n^m$  and  $q_n^m$ . For this reason  $N$  is selected as the maximum of 20, MAXL and  $\sqrt{\varepsilon}$ , where MAXL is the total number of meridional modes requested by the user.

The eigenvalues of the truncated matrices **A** and **B** are computed by first transforming the pentadiagonal matrix to a tridiagonal matrix using the EISPACK subroutine BANDR [8], and then computing the eigenvalues of the tridiagonal matrix using EISPACK subroutine IMTQL1. An eigenvalue is classified as westward gravity, rotational or eastward gravity mode depending on whether it is in the lowest, middle, or highest third, respectively, of the ordered set of eigenvalues.

The eigenvalues of **C**, **D** and **F** are computed using IMTQL1. The eigenvalues of the non-Hermitian matrix **E** are computed using a method based on determinants that was derived from a method for solving general tridiagonal systems [25].

The coefficients in the expansion of the Hough vector harmonics in terms of the spherical vector harmonics are computed in subroutine ABCOEF. See Table 2.

For  $m > 0$  the coefficients  $A$ ,  $B$ ,  $C$  are computed as the eigenvectors of **A** or **B** using EISPACK subroutine BANDV. For  $m = 0$  the coefficients are computed using a program written by Swarztrauber which computes the eigenvectors of a general tridiagonal matrix using inverse iteration. The program was written in such a way that the higher-order coefficients do not plateau at machine precision but rather converge asymptotically to zero. If the high-order coefficients are not small compared with the largest coefficients, then IERR is set to 3. A possible solution is to increase the size of MAXL, which can increase the number of coefficients in the series.

Subroutine UVH uses the coefficients that were computed by ABCOEF to compute the Hough vector functions. See Table 3. The fundamental computation in subroutine UVH is that of the associated Legendre functions  $P_n^m$ , which are computed using a package written at NCAR called ALFPAC. The spherical vector harmonics are then computed using (3.1), (3.3) and (3.4). The Hough vector functions are computed by summing the series (3.22) without the longitudinal dependence. The divergence, vorticity, velocity potential and stream function are computed from (3.33) through (3.36) in UVHDER which is like UVH but includes derived quantities. See Table 4.

**7. Summary.** In this paper we have performed a modal analysis of the linearized shallow-water equations using the spherical vector harmonics. This analysis differs from past approaches in which the vector dependent variables are replaced by scalar variables using stream function and velocity potential. Use of the vector harmonics can be generalized to other vector differential equations since it is not necessary to raise the order of the differential system. This analysis is made possible by the availability of new identities for the spherical vector harmonics that permit the replacement

TABLE 1

SUBROUTINE SIGMA (M,MAXL,IERR,EPS,EASTGS,WESTGS,ROTATS,W)

THIS PROGRAM COMPUTES THE FREQUENCIES OF THE LINEARIZED  
SHALLOW WATER EQUATIONS

INPUT PARAMETERS

M	ZONAL WAVENUMBER
MAXL	TOTAL NUMBER OF MERIDIONAL MODES. THE MODES ARE COMPUTED FOR L=0,...,MAXL-1
EPS	= $4 \cdot (A \cdot \Omega)^2 / (G \cdot H)$ WHERE
	A RADIUS OF THE EARTH
	OMEGA ANGULAR SPEED OF THE EARTH'S ROTATION
	G GRAVITY ACCELERATION
	HM EQUIVALENT HEIGHT
W	WORK STORAGE WITH AT LEAST $15 \cdot \text{MAX}(20, \text{MAXL}, \text{INT}(\text{SQRT}(\text{EPS})))$ LOCATIONS

\*\*\*\*\* IMPORTANT \*\*\*\*\*

NOTE THAT THE LENGTH OF THE WORK ARRAY W DEPENDS  
ON THE PARAMETER EPS

\*\*\*\*\*

OUTPUT PARAMETERS

IERR	= 0	NO ERROR
	= 1	MODES ARE NOT DISTINCT OR NOT ORDERED
	= 2	EISPACK SUBROUTINE INTQL1 WAS UNABLE TO DETERMINE ALL FREQUENCIES
	= 3	EPS IS LESS THAN ZERO

EASTGS	THE EASTWARD GRAVITY MODES FOR L=0,...,MAXL-1 STORED IN EASTGS(1) THROUGH EASTGS(MAXL)
--------	---

WESTGS	THE WESTWARD GRAVITY MODES FOR L=0,...,MAXL-1 STORED IN WESTGS(1) THROUGH WESTGS(MAXL)
--------	---

ROTATS	THE ROTATIONAL MODES FOR L=0,...,MAXL-1 STORED IN LOCATIONS ROTATS(1) THROUGH ROTATS(MAXL)
--------	---

THEREFORE THE ARRAYS EASTGS, WESTGS AND ROTATS  
MUST HAVE AT LEAST MAXL LOCATIONS

\*\*\*\*\*  
\*

SPECIAL ASYMPTOTIC CASES

FOR CERTAIN CASES IN WHICH THE FREQUENCIES ARE EITHER  
ZERO OR INFINITY, THE ASYMPTOTIC FORMS OF THE  
FREQUENCIES ARE COMPUTED.

1. IF EPS = 0 THEN THE FREQUENCIES OF THE GRAVITY WAVES  
ARE INFINITE AND BOTH EASTGS AND WESTGS ARE COMPUTED  
AS THE LIMIT OF  $\text{SQRT}(\text{EPS})$  TIMES THE FREQUENCY AS EPS  
GOES TO ZERO. THESE FINITE LIMITING QUANTITIES CAN  
BE USED TO COMPUTE THE NATURAL FREQUENCIES OF A  
NON-ROTATING SPHERE.

2. IF M=0 THEN THE FREQUENCIES OF THE ROTATIONAL WAVES  
ARE ZERO. IN THIS CASE SUBROUTINE SIGMA RETURNS SIGMA  
TILDE (SEE REFERENCE) WHICH IS USED BY SUBROUTINE ABCOEF  
WHEN COMPUTING THE MODES FOR M=0. THIS IS ALSO THE  
CASE FOR THE SINGLE MODE EASTGS(1) CORRESPONDING TO L=0.

\*\*\*\*\*

W	DOES NOT HAVE TO BE SAVED
---	---------------------------

TABLE 2

```

SUBROUTINE ABCOEF (M,MAXL,L,IEWR,IERR,SIG,EPS,BETA,A,B,C,W)
C
C INPUT PARAMETERS
C
C M      ZONAL WAVENUMBER
C
C MAXL   TOTAL NUMBER OF MERIDIONAL MODES. NOTE THAT
C         THE MODES ARE COMPUTED FOR L=0,...,MAXL-1
C
C L      MERIDIONAL MODE INDEX
C
C IEWR   =1 FOR EASTWARD GRAVITY WAVE
C         =2 FOR WESTWARD GRAVITY WAVE
C         =3 FOR ROTATIONAL WAVE
C
C IERR   =0 NO ERROR
C         =1 L IS GREATER THAN OR EQUAL TO MAXL
C         =2 ERROR IN EISPACK SUBROUTINE BANDV
C         =3 THE HIGH ORDER COEFFICIENTS ARE NOT SMALL
C           COMPARED TO THE LOW ORDER COEFFICIENTS
C
C SIG    DIMENSIONLESS FREQUENCY
C
C         IF IEWR=1 THEN SIG=EASTGS(L+1)
C         IF IEWR=2 THEN SIG=WESTGS(L+1)
C         IF IEWR=3 THEN SIG=ROTATS(L+1)
C
C WHERE THE ARRAYS EASTGS,WESTGS AND ROTATS ARE COMPUTED
C BY SUBROUTINE SIGMA.
C
C EPS    = 4.*(A*OMEGA)**2/(G*HM) WHERE
C
C         A      RADIUS OF THE SPHERE
C         OMEGA   ANGULAR SPEED OF ROTATION
C         G       GRAVITY ACCELERATION
C         HM      EQUIVALENT HEIGHT
C
C BETA   IS USUALLY ZERO UNLESS HAURWITZ WAVES ARE TO BE COMPUTED
C         IN WHICH CASE BETA
C
C         = 2.*A*OMEGA/G WHERE
C
C         A      RADIUS OF THE SPHERE
C         OMEGA   ANGULAR SPEED OF ROTATION
C         G       GRAVITY ACCELERATION
C
C W      A WORK ARRAY WITH 30*N LOCATIONS WHERE
C
C         N=MAXO(20,MAXL,INT(SQRT(EPS)))
C
C OUTPUT PARAMETERS
C
C A,B,C  ARRAYS WITH N LOCATIONS (SEE INPUT PARAMETER W ABOVE)
C         WHICH CONTAIN THE COEFFICIENTS IN THE EXPANSIONS OF
C         THE HOUGH VECTOR FUNCTION
C
C W      W DOES NOT HAVE TO BE SAVED
C

```

of the differential system with an algebraic system. The normal modes are then determined as the eigensolutions of infinite, banded, linear systems of equations.

New zonal rotational modes ( $m=0$ ) are computed as the limit of the Hough vector functions as the longitudinal wavenumber  $m$  tends to zero. Although the zonal modes are not unique, the set that is obtained in this manner is particularly interesting since it shares many properties with the nonzonal modes. The frequencies of these modes are zero and therefore correspond to steady solutions with respect to the rotating sphere. The asymptotic behavior of the frequencies, as  $m$  tends to zero, is obtained as a by-product of the method that is used to obtain the normal modes. The gravity modes with finite frequencies are also computed. All modes are determined from the eigensolutions of infinite tridiagonal linear systems of equations.

TABLE 3

```

SUBROUTINE UVH (M,MAXL,L,IEWR,EPS,NT,PHI,A,B,C,U,V,H,W)
C
C   GIVEN THE COEFFICIENT ARRAYS A,B AND C COMPUTED BY SUBROUTINE
C   ABCOEF, THEN THIS SUBROUTINE TABULATES THE HORIZONTAL VELOCITY
C   FIELDS U,V AND HEIGHT FIELD H AT THE LATITUDES SPECIFIED
C   IN THE ARRAY PHI
C
C   INPUT PARAMETERS
C
C   M       ZONAL WAVENUMBER
C
C   MAXL    TOTAL NUMBER OF MERIDIONAL MODES. NOTE THAT
C           THE MODES ARE COMPUTED FOR L=0,...,MAXL-1
C
C   L       MERIDIONAL MODE INDEX
C
C   IEWR    =1 FOR EASTWARD GRAVITY WAVE
C           =2 FOR WESTWARD GRAVITY WAVE
C           =3 FOR ROTATIONAL WAVE
C
C   EPS     =4.*(AE*OMEGA)**2/(G*HM) WHERE
C           AE   RADIUS OF THE SPHERE
C           OMEGA ANGULAR SPEED OF ROTATION
C           G     GRAVITY ACCELERATION
C           HM    EQUIVALENT HEIGHT
C
C   NT      NUMBER OF LATITUDES IN THE ARRAY PHI BELOW
C
C   PHI     AN ARRAY WHICH CONTAINS THE LATITUDES AT WHICH U,V
C           AND H ARE TABULATED (LATITUDES SHOULD BE SPECIFIED
C           IN RADIANS)
C
C   A,B,C   ARRAYS WITH N LOCATIONS (SEE INPUT PARAMETER W BELOW)
C           WHICH CONTAIN THE COEFFICIENTS COMPUTED BY SUBROUTINE
C           ABCOEF
C
C   W       A WORK ARRAY WITH 10*N+5*M LOCATIONS WHERE
C           N=MAXO(20,MAXL,INT(SQRT(EPS)))
C           NOTE
C           THIS ARRAY CAN BE THE SAME AS THE
C           ARRAY USED IN SUBROUTINE SIGMA
C
C   OUTPUT PARAMETERS
C
C   U       AN ARRAY OF LENGTH NT WHICH CONTAINS THE LONGITUDINAL
C           VELOCITY TABULATED AT THE LATITUDES SPECIFIED IN THE
C           ARRAY PHI
C
C   V       AN ARRAY OF LENGTH NT WHICH CONTAINS THE MERIDIONAL
C           VELOCITY TABULATED AT THE LATITUDES SPECIFIED IN THE
C           ARRAY PHI
C
C   H       AN ARRAY OF LENGTH NT WHICH CONTAINS THE HEIGHT FIELD
C           TABULATED AT THE LATITUDES SPECIFIED IN THE ARRAY PHI
C
C   W       DOES NOT HAVE TO BE SAVED
C
C
C

```

The components of the Hough vector functions are in transformed variables. They consist of the velocity components and the height of the free surface that have been transformed in such a way as to reduce the number of physical parameters in the shallow-water equations to a single dimensionless parameter. All of the modal solutions to the original untransformed equations cannot be obtained from the transformed system. The particular modes that correspond to an infinite equivalent height cannot be obtained as a limiting case of the transformed system. These modes are called the Haurwitz modes and are determined as the limit of solutions to the untransformed shallow-water equations as the equivalent height tends to infinity.

TABLE 4

SUBROUTINE UVHDER (M,MAXL,L,IEWR,EPS,NT,PHI,A,B,C,U,V,H,STRMFN, 1 VELPOT,DVRGNC,VRTCTY,GRDNTU,GRDNTV,GRDNTH,W)	
C	
C	
C	INPUT PARAMETERS
C	
C	M ZONAL WAVENUMBER
C	
C	MAXL TOTAL NUMBER OF MERIDIONAL MODES
C	
C	L MERIDIONAL MODE INDEX
C	
C	IEWR =1 FOR EASTWARD GRAVITY WAVE
C	=2 FOR WESTWARD GRAVITY WAVE
C	=3 FOR ROTATIONAL WAVE
C	
C	EPS =4.*(A*OMEGA)**2/(G*HM) WHERE
C	
C	A RADIUS OF SPHERE
C	OMEGA ANGULAR SPEED OF THE EARTHS ROTATION
C	G GRAVITY ACCELERATION
C	HM EQUIVALENT HEIGHT
C	
C	NT NUMBER OF LATITUDES IN THE ARRAY PHI BELOW
C	
C	PHI AN ARRAY WHICH CONTAINS THE LATITUDES AT WHICH U,V
C	AND H ARE TABULATED (LATITUDES SHOULD BE SPECIFIED
C	IN RADIANES)
C	
C	A,B,C ARRAYS WITH N LOCATIONS (SEE INPUT PARAMETER W BELOW)
C	WHICH CONTAIN THE COEFFICIENTS COMPUTED BY SUBROUTINE
C	ABCOEF
C	
C	W A WORK ARRAY WITH 27*N+32 LOCATIONS WHERE
C	
C	N=MAXO(20,MAXL,INT(SQRT(EPS)))
C	
C	NOTE
C	THIS ARRAY CAN BE THE SAME AS THE
C	ARRAY USED IN SUBROUTINE SIGMA
C	
C	OUTPUT PARAMETERS
C	
C	U AN ARRAY OF LENGTH NT WHICH CONTAINS THE LONGITUDNAL
C	VELOCITY TABULATED AT THE LATITUDES SPECIFIED IN THE
C	ARRAY PHI
C	
C	V AN ARRAY OF LENGTH NT WHICH CONTAINS THE MERIDONAL
C	VELOCITY TABULATED AT THE LATITUDES SPECIFIED IN THE
C	ARRAY PHI
C	
C	H AN ARRAY OF LENGTH NT WHICH CONTAINS THE HEIGHT FIELD
C	TABULATED AT THE LATITUDES SPECIFIED IN THE ARRAY PHI
C	
C	STRMFN AN ARRAY OF LENGTH NT WHICH CONTAINS THE STREAM FUNCTION
C	PSI TABULATED AT THE LATITUDES SPECIFIED IN THE ARRAY PHI.
C	
C	VELPOT AN ARRAY OF LENGTH NT WHICH CONTAINS THE VELOCITY POTENTIAL
C	TABULATED AT THE LATITUDES SPECIFIED IN THE ARRAY PHI.
C	
C	DVRGNC AN ARRAY OF LENGTH NT WHICH CONTAINS THE DIVERGENCE
C	TABULATED AT THE LATITUDES SPECIFIED IN THE ARRAY PHI.
C	
C	VRTCTY AN ARRAY OF LENGTH NT WHICH CONTAINS THE VORTICITY
C	TABULATED AT THE LATITUDES SPECIFIED IN THE ARRAY PHI.
C	
C	GRDNTU AN ARRAY OF LENGTH NT WHICH CONTAINS D(COS(PHI)*U)/DPHI
C	TABULATED AT THE LATITUDES SPECIFIED IN THE ARRAY PHI.
C	
C	GRDNTV AN ARRAY OF LENGTH NT WHICH CONTAINS D(COS(PHI)*V)/DPHI
C	TABULATED AT THE LATITUDES SPECIFIED IN THE ARRAY PHI.
C	
C	GRDNTH AN ARRAY OF LENGTH NT WHICH CONTAINS DH/DPHI TABULATED AT
C	THE LATITUDES SPECIFIED IN THE PHI ARRAY.
C	
C	W DOES NOT HAVE TO BE SAVED
C	
C	









TABLE 8

EIGENFREQUENCIES SIGMA FOR ZONAL WAVE NUMBER M = 3

L	EPSILON					EASTWARD GRAVITY					WESTWARD GRAVITY							
	10(-3)	10(-2)	10(-1)	10(0)	10(1)	10(2)	10(3)	10(4)	10(5)	10(-3)	10(-2)	10(-1)	10(0)	10(1)	10(2)	10(3)	10(4)	10(5)
0	109.4200576	34.5177385	3.3555953	0.8348487	1.0154930	0.3076864	0.0956286	0.0300754	0.0094943	5706828	-0.5706828	-1.2750161	-3.6068555	-11.0849784	-34.7677515	-104.6700589	-300.0174105	-94.8873491
1	141.3472919	44.6493111	4.4255259	14.0763923	4.4255259	1.4173666	0.5219739	0.1163748	0.0612117	5775613	-4.5775613	-12.5017977	-36.0689977	-114.2265997	-44.7091310	-141.4872919	-331.6768028	-104.9669544
2	173.1560652	54.7253644	5.4576433	17.2802935	5.4576433	1.7708969	0.6860957	0.1804375	0.0993715	636335	-5.5688097	-15.8822816	-48.5464896	-173.8041147	-54.8253768	-173.2560652	-304.9756712	-104.8461010
3	204.9042419	64.7743683	6.4743683	20.4675624	6.4743683	2.1020006	0.8195572	0.2281116	0.1269440	7092016	-6.5379204	-18.0027816	-53.6987090	-203.5306603	-64.8461010	-204.9756712	-328.3497374	-104.8761993
4	236.6172801	74.8091259	7.4839345	23.6462580	7.4839345	2.4234466	0.9348643	0.2675111	0.1495811	7881581	-7.4831581	-20.1024767	-57.96519	-236.8617278	-74.8627015	-236.6172801	-350.0174105	-104.8873491
5	268.3081305	84.8345300	8.4897961	26.8200352	8.4897961	2.7402827	1.0391189	0.2675111	0.1692233	8720162	-8.4897961	-22.1024767	-62.271430	-268.8617278	-84.8873491	-268.3081305	-373.0174105	-104.9086878
6	299.9840770	94.8540141	9.4935775	29.997612	9.4935775	3.0561618	1.1368391	0.2675111	0.1888020	9611748	-9.4935775	-24.1024767	-67.5271430	-299.8617278	-94.8873491	-299.9840770	-396.0174105	-104.9301189
7	331.6495300	104.8694215	10.4961147	33.1594788	10.4961147	3.3705472	1.2309943	0.2675111	0.2082847	1050344	-10.4961147	-26.1024767	-72.7851184	-331.6495300	-104.9086878	-331.6495300	-413.0174105	-104.9517959
8	363.3073163	114.8819049	11.4978695	36.3268114	11.4978695	3.6847918	1.3234613	0.2675111	0.2288479	1139524	-11.4978695	-28.1024767	-78.03174	-363.3073163	-114.9086878	-363.3073163	-436.0174105	-104.9730118
9	394.9593313	124.8922213	12.4991124	39.4931524	12.4991124	3.9988475	1.4153442	0.2675111	0.2492444	1228611	-12.4991124	-30.1024767	-83.28301	-394.9593313	-124.9301189	-394.9593313	-459.0174105	-104.9942159
10	426.6088931	134.9008878	13.5006650	42.6587611	13.5006650	4.271215	1.5094244	0.2675111	0.2704490	1317700	-13.5006650	-32.1024767	-88.53017	-426.6088931	-134.9517959	-426.6088931	-482.0174105	-105.0146321
11	458.2509489	144.9082697	14.5006650	45.8283152	14.5006650	4.5412427	1.5994244	0.2675111	0.2916536	1406789	-14.5006650	-34.1024767	-93.76530	-458.2509489	-144.9517959	-458.2509489	-505.0174105	-105.0359521
12	489.8921882	154.9146321	15.5011498	48.984338	15.5011498	4.814237	1.6921374	0.2675111	0.3128582	1495878	-15.5011498	-36.1024767	-99.00020	-489.8921882	-154.9517959	-489.8921882	-528.0174105	-105.0576733
13	521.5311399	164.9201719	16.5015106	52.1527255	16.5015106	5.08521	1.7831076	0.2675111	0.3340628	1584967	-16.5015106	-38.1024767	-104.23017	-521.5311399	-164.9517959	-521.5311399	-551.0174105	-105.0793945
14	553.1682030	174.9250385	17.5017799	55.3167396	17.5017799	5.3764056	1.8749831	0.2675111	0.3552674	1674056	-17.5017799	-40.1024767	-109.46348	-553.1682030	-174.9517959	-553.1682030	-574.0174105	-105.1011137
15	584.8036900	184.9293474	18.5019809	58.4805327	18.5019809	5.6580783	1.9731323	0.2675111	0.3764720	1763145	-18.5019809	-42.1024767	-114.693765	-584.8036900	-184.9517959	-584.8036900	-597.0174105	-105.1228359

The description of software for computing the Hough vector functions as well as the Haurwitz modes was also presented. The software package consists of four user-entry FORTRAN subroutines called SIGMA, ABCOEF, UVH and UVHDER. Subroutine SIGMA computes the frequencies of the normal modes as eigenvalues of the infinite banded systems. Subroutine ABCOEF computes the coefficients in the expansion of the mode in terms of the spherical vector harmonics. Subroutine UVH tabulates the components of the Hough vector function as functions of latitude and subroutine UVHDER tabulates certain derivatives of the components.

## REFERENCES

- [1] S. CHAPMAN AND R. S. LINDZEN, *Atmospheric Tides*, Gordon and Breach, New York, 1970.
- [2] W. C. CHAO AND M. A. GELLER, *Utilization of normal mode initial conditions for detecting errors in the dynamics part of primitive equation global models*, Monthly Weather Review, 110 (1982), pp. 304-306.
- [3] R. DALEY, *Normal mode initialization*, Rev. Geophys. Space Phys., 19 (1981), pp. 450-468.
- [4] I. A. DIKII, *The terrestrial atmosphere as an oscillating system*, Izv. Atmosph. Oceanic Phys., 1 (1965), pp. 275-286.
- [5] R. E. DICKINSON, *Propagators of atmospheric motions*, Ph.D. thesis, Massachusetts Institute of Technology, Cambridge, 1966.
- [6] T. W. FLATTERY, *Hough functions*, Tech. Rep. 21, Dept. Geophys. Sci., Univ. Chicago, Chicago, 1967.
- [7] ———, *Spectral models for global analysis and forecasting*, Proc. Sixth AWS Technical Exchange Conf., U.S. Naval Academy, Air Weather Service Tech. Rep. 242 (1970), pp. 42-53.
- [8] 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.
- [9] B. HAURWITZ, *The motion of atmospheric disturbances on the earth*, J. Marine Res., 3 (1940), pp. 254-267.
- [10] S. S. HOUGH, *On the application of harmonic analysis to the dynamical theory of the tides—Part II. On the general integration of Laplace's dynamical equations*, Phil. Trans. Roy. Soc. London, A191 (1898), pp. 139-185.
- [11] M. N. JONES, *Atmospheric oscillations—I*, Planet. Space Sci., 18 (1970), pp. 1393-1416.
- [12] A. KASAHARA, *Normal modes of ultralong waves in the atmosphere*, Monthly Weather Review, 104 (1976), pp. 669-690.
- [13] ———, *Numerical integration of the global barotropic primitive equations with Hough harmonic expansions*, J. Atmos. Sci., 34 (1977), pp. 687-701.
- [14] ———, *Further studies on a spectral model of the global barotropic primitive equations with Hough harmonic expansions*, J. Atmos. Sci., 35 (1978), pp. 2043-2051.
- [15] H. LAMB, *Hydrodynamics*, 6th edition, Dover, New York, 1932.
- [16] M. S. LONGUET-HIGGINS, *The eigenfunctions of Laplace's tidal equations over a sphere*, Phil. Trans. Roy. Soc. London, A262 (1968), pp. 511-601.
- [17] M. MARGULES, *Luftbewegungen in einer rotierenden Spharoidschale. Teil II*, Sitz.-Ber., Akad. Wiss. Wien, Math.-Naturwiss. I., Abt. IIa, 102 (1893), pp. 11-56. (English translation by B. Haurwitz is available from the National Center for Atmospheric Research, Boulder, Colorado, as technical note NCAR/TN-156+STR, entitled "Air motions in a rotating spheroidal shell".)
- [18] T. MATSUNO, *Quasi-geostrophic motions in the equatorial area*, J. Meteor. Soc. Japan, 44 (1966), pp. 25-43.
- [19] P. M. MORSE AND H. FESHBACH, *Methods of Theoretical Physics*, McGraw-Hill, New York, 1953.
- [20] H. E. MOSES, *The use of vector spherical harmonics in global meteorology and aeronomy*, J. Atmos. Sci., 31 (1974), pp. 1490-1499.
- [21] J. PEDLOSKY, *Geophysical Fluid Dynamics*, Springer-Verlag, New York, 1979.
- [22] G. W. PLATZMAN, *Two-dimensional free oscillation in natural basins*, J. Phys. Oceanogr., 2 (1972), pp. 117-138.
- [23] Y. SHIGEHISA, *Normal modes of the shallow water equations for zonal wavenumber zero*, J. Meteor. Soc. Japan, 61 (1983), pp. 479-494.
- [24] M. SIEBERT, *Atmospheric tides*, Advances in Geophysics, 7 (1961), pp. 105-187.
- [25] P. N. SWARZTRAUBER, *A parallel algorithm for solving general tridiagonal equations*, Math. Comp., 33 (1979), pp. 185-199.

- [26] P. N. SWARZTRAUBER, *On the spectral approximation of discrete scalar and vector functions on the sphere*, SIAM J. Numer. Anal., 16 (1979), pp. 934–949.
- [27] ———, *The approximation of vector functions and their derivatives on the sphere*, SIAM J. Numer. Anal., 18 (1981), pp. 934–949.
- [28] ———, *On the spectral analysis of vector differential equations on the sphere*, Proc. Fifth IMACS International Symposium on Computer Methods for Partial Differential Equations, Lehigh Univ., Bethlehem, PA, 1984.
- [29] G. I. TAYLOR, *The oscillations of the atmosphere*, Proc. Roy. Soc. London, A156 (1936), pp. 318–326.

# THE CONVERGENCE RATE OF MULTI-LEVEL ALGORITHMS APPLIED TO THE CONVECTION-DIFFUSION EQUATION\*

P. M. DE ZEEUW† AND E. J. VAN ASSELT†

**Abstract.** We consider the solution of the convection-diffusion equation in two dimensions by various multi-level algorithms (MLAs). We study the convergence rate of the MLAs and the stability of the coarse-grid operators, depending on the choice of artificial viscosity at the different levels. Four strategies are formulated and examined. A method to determine the convergence rate is described and applied to the MLAs, both in a problem with constant and in one with variable coefficients. As relaxation procedures the 7-point ILU and symmetric point Gauss-Seidel (SGS) methods are used.

**Key words.** artificial viscosity, convection-diffusion equation, multi-level algorithm, asymptotic stability, Galerkin approximation

## 1. Introduction. We consider the convection-diffusion equation

$$(1.1) \quad L_\epsilon u \equiv -\epsilon \Delta u + b_1(x, y) \frac{\partial u}{\partial x} + b_2(x, y) \frac{\partial u}{\partial y} = f(x, y)$$

for  $(x, y) \in \Omega \subset \mathbb{R}^2$ ,  $\epsilon > 0$ , with Dirichlet and Neumann boundary conditions on different parts of  $\delta\Omega$ .

When the diffusion coefficient  $\epsilon$  is small in comparison with the mesh-width  $h$ , the stability of discretizations of (1.1) by central differences (CD) or the finite element method (FEM) can be improved by augmenting  $\epsilon$  with an artificial viscosity of  $O(h)$ . This rather crude way of stabilizing the discrete problem may form part of more subtle iterative methods for solving (1.1) with small  $\epsilon$ , for instance the mixed defect correction process (cf. Hemker [4]) or the double discretization process (cf. Brandt [3]).

In § 2 we introduce four strategies for choosing the artificial viscosity on the coarse grids in the multi-level algorithm (MLA) (cf. Van Asselt [1]). In § 3 we describe the method which is used to determine the convergence behaviour of the multi-level algorithm for these strategies. In § 4 we compare the convergence rates as measured by the method described in § 3. Finally, some conclusions are formulated in § 5.

**2. Artificial viscosity, strategies, stability and asymptotic convergence rate.** In this section we derive all theoretical results for the constant coefficient case by local mode analysis neglecting the boundaries. We introduce various strategies for choosing the coarse-grid operators in the MLA. We give a motivation for the choice of these strategies, and analyze their stability (cf. Theorem 2.14, Corollary 2.18, Theorem 2.19, Corollary 2.24). Further we formulate some important properties of the different strategies (cf. Conjectures 2.25–2.27). In the case of FEM discretization we also consider the Galerkin coarse-grid approximation. In this paper we only consider the FEM based on a uniform triangulation of  $\Omega$  with right-angled triangles.

The trial and test space is spanned by the set of piecewise-linear “hat-functions”  $\phi_{ij}$  which take the value 1 at  $x_{ij}$  and 0 at all other vertices of triangles.

We consider the MLA (cf. Hemker [5]) with  $l+1$  levels:  $0, \dots, l$  and uniform square meshes on each level with meshwidths  $h_0$  and  $h_k = h_{k-1}/2$  for  $k = 1, \dots, l$ .

Let  $\{L_\epsilon^{k,l}\}_{k=0,\dots,l}$  be a sequence of discretizations of  $L_\epsilon$ . For the constant-coefficient equation we denote by  $\hat{L}_\epsilon(\omega)$ ,  $\omega \in \mathbb{R}^2$  the symbol (or characteristic form) of the

\* Received by the editors May 5, 1983, and in revised form January 3, 1984.

† Mathematical Centre, Kruislaan 413, 1098 SJ Amsterdam, the Netherlands.

continuous operator  $L_\epsilon$ . By  $\hat{L}_\epsilon^{k,l}(\omega)$ ,  $\omega \in T_k \equiv [-\pi/h_k, \pi/h_k]^2$ , we mean the symbol of the discrete operator  $L_\epsilon^{k,l}$ .

When a symbol is small the corresponding operator is unstable in the sense that small changes in the right-hand side cause great changes in the solution. Depending on the boundary conditions the continuous problem can be well posed. Therefore we allow the symbol of the discrete operator to be small only for those frequencies for which the symbol of the continuous operator is small. This idea is formalized in the following definitions.

DEFINITION 2.1. The  $\epsilon$ -asymptotic stability degree of  $L_\epsilon$  with respect to the mode  $e^{i\omega x}$  is the quantity  $\lim_{\epsilon \downarrow 0} |\hat{L}_\epsilon(\omega)|$ .

DEFINITION 2.2. The  $\delta$ -domain of  $L_\epsilon$  is the set of all  $\omega \in \mathbb{R}^2$  for which  $\lim_{\epsilon \downarrow 0} |\hat{L}_\epsilon(\omega)| > \delta > 0$ .

DEFINITION 2.3. The  $\epsilon$ -asymptotic stability degree of  $L_\epsilon^{k,l}$  with respect to the mode  $e^{i\omega x}$  is the quantity  $\lim_{\epsilon \downarrow 0} |L_\epsilon^{k,l}(\omega)|$ .

DEFINITION 2.4. The  $\delta$ -domain of  $L_\epsilon^{k,l}$  is the set of all  $\omega \in T_k$  for which  $\lim_{\epsilon \downarrow 0} |\hat{L}_\epsilon^{k,l}(\omega)| > \delta > 0$ .

DEFINITION 2.5. A strategy for coarse-grid operators is a set  $\{L_\epsilon^0, L_\epsilon^1, \dots, L_\epsilon^l, \dots\}$  with  $L_\epsilon^l \equiv \{L_\epsilon^{0,l}, \dots, L_\epsilon^{l,l}\}$ .

DEFINITION 2.6. Let  $S$  be a strategy for coarse-grid operators, then  $S$  is  $\epsilon$ -asymptotically stable with respect to  $L_\epsilon$  if for every  $\delta_0 > 0$  there exists a  $\delta_1 > 0$  such that for all  $0 \leq k \leq l$ , we have the  $\delta_1$ -domain of  $L_\epsilon^{k,l} \supset \delta_0$ -domain of  $L_\epsilon \cap T_k$ .

Remark 2.7. In order to avoid residual transfers in the MLA that are useless due to oscillating solutions, we require that a strategy is  $\epsilon$ -asymptotically stable with respect to  $L_\epsilon$ . Moreover we need a relaxation method for which the smoothing factors on all grids are less than 1. We then expect rapid convergence of the MLA.

Another approach would be to admit  $\epsilon$ -asymptotically unstable strategies and to require that the relaxation method is such that bad components in the residuals are sufficiently smoothed. This poses very strong demands upon the relaxation method. If a strategy is not  $\epsilon$ -asymptotically stable with respect to  $L_\epsilon$ , and the relaxation method can not sufficiently damp the oscillations we may expect divergence if the number of levels increases.

By  $L_{\epsilon+\beta_k^l, h_k}$  we denote a discretization of (1.1) with artificial viscosity  $\beta_k^l$  and meshwidth  $h_k$ , and for fixed  $h_0$  and  $\gamma > 0$  (independent of  $\epsilon, k$  and  $l$ ) we will consider the following four strategies for coarse-grid operators:

Strategy 1 ( $S_1$ ):

$$(2.8) \quad L_\epsilon^{k,l} = L_{\epsilon+\beta_k^l, h_k} \quad \text{and} \quad \beta_k^l = \gamma h_k, \quad k = 0, \dots, l.$$

Strategy 2 ( $S_2$ ):

$$(2.9) \quad L_\epsilon^{k,l} = L_{\epsilon+\beta_k^l, h_k}, \quad \beta_l^l = \gamma h_l, \quad \beta_k^l = \gamma h_{k+1}, \quad k = 0, \dots, l-1.$$

Strategy 3 ( $S_3$ ):

$$(2.10) \quad L_\epsilon^{k,l} = L_{\epsilon+\beta_k^l, h_k} \quad \text{and} \quad \beta_k^l = \gamma h_k, \quad k = 0, \dots, l.$$

Strategy 4 ( $S_4$ ):

$$(2.11) \quad L_\epsilon^{l,l} \equiv L_{\epsilon+\beta_l^l, h_l} \quad \text{with} \quad \beta_l^l = \gamma h_l, \quad L_\epsilon^{k,l} \equiv R_{k,k+1} L_\epsilon^{k+1,l} P_{k+1,k}, \quad k = l-1, \dots, 0.$$

( $R_{k,k+1}$  and  $P_{k+1,k}$  are the restriction and the prolongation which are consistent with the FEM used.)

*Remark 2.12.* The choice of  $L_\varepsilon^{k,l}$  according to  $S_4$  is called Galerkin coarse-grid approximation. If we consider a constant-coefficient problem and neglect the boundaries, then a coarse-grid operator constructed with the FEM according to  $S_1$ , is identical with the Galerkin coarse-grid approximation as in  $S_4$ . The molecule is given by

$$L_{\varepsilon+\beta_k^l, h_k} = \frac{\varepsilon + \beta_k^l}{h_k^2} \begin{bmatrix} 0 & -1 \\ -1 & 4 & -1 \\ & -1 & 0 \end{bmatrix} + \frac{b_1}{6h_k} \begin{bmatrix} -1 & 1 \\ -2 & 0 & 2 \\ & -1 & 1 \end{bmatrix} + \frac{b_2}{6h_k} \begin{bmatrix} 1 & 2 \\ -1 & 0 & 1 \\ & -2 & -1 \end{bmatrix}.$$

*Remark 2.13.* It follows from (2.8)–(2.10) that

$$\text{for } S_1: \lim_{l \rightarrow \infty} \beta_0^l / h_k = \lim_{l \rightarrow \infty} \gamma / 2^l = 0,$$

$$\text{for } S_2: \beta_k^l / h_k \geq \gamma / 2 \text{ uniformly for all } k, l,$$

$$\text{for } S_3: \beta_k^l / h_k = \gamma \text{ uniformly for all } k, l.$$

In Theorem 2.14, Corollary 2.18 and Corollary 2.24 we will prove that  $S_1$  and  $S_4$  are not  $\varepsilon$ -asymptotically stable and  $S_2$  and  $S_3$  are. Further we will point out that the convergence rate of the MLA with  $S_2$  is better than with  $S_3$ .

**THEOREM 2.14.** *Consider the CD- or FEM-discretizations of (1.1) with artificial viscosity  $\beta_k^l$  and constant coefficients; then  $S_1$  is not  $\varepsilon$ -asymptotically stable with respect to  $L_\varepsilon$ .*

*Proof.* We give the proof only for the CD-discretizations; the proof for the FEM-discretizations is similar. The CD-discretization of (1.1) with artificial viscosity  $\beta_k^l$  and constant coefficients  $b_1$  and  $b_2$ ,  $b_1^2 + b_2^2 = 1$ , reads

$$\begin{aligned} L_{\varepsilon+\beta_k^l, h_k} u \equiv & \left( -\frac{\varepsilon + \beta_k^l}{h_k^2} - \frac{b_2}{2h_k} \right) u_{i,j-1}^{h_k} + \left( -\frac{\varepsilon + \beta_k^l}{h_k^2} + \frac{b_2}{2h_k} \right) u_{i,j+1}^{h_k} \\ & + \left( -\frac{\varepsilon + \beta_k^l}{h_k^2} - \frac{b_1}{2h_k} \right) u_{i-1,j}^{h_k} + \left( -\frac{\varepsilon + \beta_k^l}{h_k^2} + \frac{b_1}{2h_k} \right) u_{i+1,j}^{h_k} \\ & + 4 \left( \frac{\varepsilon + \beta_k^l}{h_k^2} \right) u_{i,j}^{h_k} = f_{i,j}^{h_k}. \end{aligned} \tag{2.15}$$

Its characteristic form reads

$$\hat{L}_{\varepsilon+\beta_k^l, h_k}(\omega) = -\frac{2(\varepsilon + \beta_k^l)(\cos \omega_1 h_k + \cos \omega_2 h_k - 2)}{h_k^2} + \frac{i(b_1 \sin \omega_1 h_k + b_2 \sin \omega_2 h_k)}{h_k}. \tag{2.16}$$

The characteristic form of  $L_\varepsilon$  reads

$$\hat{L}_\varepsilon(\omega) = \varepsilon(\omega_1^2 + \omega_2^2) + i(b_1 \omega_1 + b_2 \omega_2), \tag{2.17}$$

hence the  $\delta_0$ -domain of  $L_\varepsilon$  is the set of all  $\omega \in \mathbb{R}^2$  for which  $|b_1 \omega_1 + b_2 \omega_2| > \delta_0 > 0$ . We have to show that a  $\delta_0 > 0$  exists such that for all  $\delta_1 > 0$  there exist  $k, l \in \mathbb{Z}$ ,  $0 \leq k \leq l$ , such that for an  $\tilde{\omega} \in \mathbb{R}^2$  with  $\tilde{\omega} \in (\delta_0\text{-domain of } L_\varepsilon) \cap T_k$  we have  $\tilde{\omega} \notin \delta_1$ -domain of  $L_{\varepsilon+\beta_k^l, h_k}$ . For that purpose we proceed as follows. Take  $\delta_0 = 0.1\pi/h_0$  and let  $\delta_1 > 0$  be arbitrary. Take  $k = 0$  and  $l > \log_2(4\gamma/h_0\delta_1)$ ; then for either  $\tilde{\omega} = (\pi/h_0, 0) \in T_0$  or  $\tilde{\omega} = (0, \pi/h_0) \in T_0$  both  $|b_1 \tilde{\omega}_1 + b_2 \tilde{\omega}_2| > \delta_0$  and  $\lim_{\varepsilon \downarrow 0} |\hat{L}_{\varepsilon+\beta_0^l, h_0}(\tilde{\omega})| = 4\gamma/(h_0 2^l) < \delta_1$  hold. Hence  $S_1$  is not  $\varepsilon$ -asymptotically stable with respect to  $L_\varepsilon$ .

This leads us to

**COROLLARY 2.18.** *Consider  $L_\varepsilon$  with constant coefficients  $b_1$  and  $b_2$ ; then  $S_4$  is not  $\varepsilon$ -asymptotically stable with respect to  $L_\varepsilon$ .*

*Proof.* The proof follows immediately from (2.12) and (2.14).

**THEOREM 2.19.** *Consider the CD-discretizations of (1.1) with artificial viscosity  $\beta_k^l$  and constant coefficients. Let  $S$  be a strategy with  $\beta_k^l/h_k \geq C > 0$  uniformly for all  $k, l (k \leq l) \in \mathbb{Z}$ ; then  $S$  is  $\varepsilon$ -asymptotically stable.*

*Proof.* Again we use (2.15)–(2.17). We have to prove:

$$\forall \delta_0 > 0 \exists \delta_1 > 0 \forall k, l, 0 \leq k \leq l$$

$$\Rightarrow \delta_0\text{-domain of } L_\varepsilon \cap T_k \subset \delta_1\text{-domain of } L_{\varepsilon+\beta_k^l, h_k}.$$

Take  $\delta_1 = \min(\frac{1}{2}, 2C/5)\delta_0$ . In the case  $\delta_0 > 2^{1/2}\pi/h_k$  the inclusion is trivially satisfied because  $\delta_0$ -domain of  $L_\varepsilon \cap T_k = \emptyset$ . If  $0 < \delta_0 \leq 2^{1/2}\pi/h_k$  then  $\omega \in \delta_0$ -domain of  $L_\varepsilon \cap T_k$  implies

$$\delta_0 h_k < |b_1 \omega_1 h_k + b_2 \omega_2 h_k|.$$

The normalization  $b_1^2 + b_2^2 = 1$  and the inequality  $|\sin x - x| \leq |x^3|/4$  for all  $x \in \mathbb{R}$  yield

$$(2.20) \quad \delta_0 h_k < |b_1 \sin \omega_1 h_k + b_2 \sin \omega_2 h_k| + \frac{|\omega_1 h_k|^3}{4} + \frac{|\omega_2 h_k|^3}{4}.$$

We distinguish the two complementary cases:

- (i)  $|\omega_1 h_k|^3 \leq \delta_0 h_k$  and  $|\omega_2 h_k|^3 \leq \delta_0 h_k$ ;
- (ii)  $|\omega_1 h_k|^3 > \delta_0 h_k$  or  $|\omega_2 h_k|^3 > \delta_0 h_k$ .

Because of (2.16) and (2.20) case (i) implies:

$$(2.21) \quad \lim_{\varepsilon \downarrow 0} |\hat{L}_{\varepsilon+\beta_k^l, h_k}(\omega)| \geq \frac{|b_1 \sin \omega_1 h_k + b_2 \sin \omega_2 h_k|}{h_k} > \frac{\delta_0}{2} \geq \delta_1.$$

To complete the proof we now consider case (ii). It follows from (2.16) and  $\beta_k^l/h_k \geq C$  that

$$(2.22) \quad \lim_{\varepsilon \downarrow 0} |\hat{L}_{\varepsilon+\beta_k^l, h_k}(\omega)| \geq \frac{2C(1 - \cos \omega_1 h_k + 1 - \cos \omega_2 h_k)}{h_k},$$

and from (ii) and  $0 < \delta_0 h_k \leq 2^{1/2}\pi$  it follows that the right-hand side of (2.22) is greater than or equal to

$$\frac{2C\delta_0(1 - \cos((\delta_0 h_k)^{1/3}))}{\delta_0 h_k},$$

hence

$$(2.23) \quad \lim_{\varepsilon \downarrow 0} |\hat{L}_{\varepsilon+\beta_k^l, h_k}(\omega)| > \frac{2C\delta_0}{5} \geq \delta_1 > 0.$$

Both (2.21) and (2.23) hold uniformly for all  $k, l$  so  $S$  is  $\varepsilon$ -asymptotically stable with respect to  $L_\varepsilon$ .

Note that the condition of Theorem 2.19 is satisfied by taking on coarser grids the artificial viscosity proportional to the current meshwidth.

**COROLLARY 2.24.** *Consider the CD-discretizations of (1.1) with artificial viscosity  $\beta_k^l$  and constant coefficients; then  $S_2$  and  $S_3$  are  $\varepsilon$ -asymptotically stable with respect to  $L_\varepsilon$ .*

*Proof.* The proof follows immediately from Remark 2.13 and Theorem 2.19.

It is obvious that the  $\varepsilon$ -asymptotic stability degree of the individual grid-operators belonging to  $S_2$  is larger than in the case of  $S_1$ . Moreover for decreasing  $\gamma$  the smoothing factors for  $S_1$  become worse (cf. Table 2). We formulate this in the following

*Conjecture 2.25.* For a *fixed* number of levels the set of  $\gamma$ -values for which the MLA with  $S_2$  converges, is larger than that for which the MLA with  $S_1$  converges.

In case of a two-level algorithm (TLA),  $l = 1$ , and a constant-coefficient problem, a two-level analysis shows that the asymptotic rate of convergence for  $S_1$  or  $S_2$ , for which the artificial viscosity is equal on both levels is better than for  $S_3$ , where the artificial viscosity corresponds to the meshwidth. (cf. Van Asselt [1]). Therefore in  $S_1$  we take an equal artificial viscosity on all levels. For this strategy, however, stability problems may occur on coarser grids (cf. Theorem 2.14).  $S_3$  is  $\varepsilon$ -asymptotically stable (cf. Corollary 2.24), but the two-level analysis indicates that the convergence rate is slower.  $S_2$  is an intermediate strategy where on levels  $l$  and  $l - 1$  the artificial viscosity is the same, and it is also  $\varepsilon$ -asymptotically stable (cf. Corollary 2.24). These arguments lead to the following

*Conjecture 2.26.*  $S_2$  combines the rapid convergence rate of  $S_1$  with the stability of  $S_3$ .

At level  $l$  the discrete operators  $L_{\varepsilon+\beta^l, h_l}$  using  $S_1, S_2, S_3$  are equal.

At level  $l - 1$  the discrete operators  $L_{\varepsilon+\beta^{l-1}, h_{l-1}}$  using  $S_1, S_2$  are equal ( $S_3$  is not), and the relative order of consistency of the  $S_1$  and  $S_2$  operators on level  $l$  and  $l - 1$  is the same and higher than that of  $S_3$ . Furthermore, consider the part of  $T_l$  where the smoothing effect of a relaxation method applied to  $S_2$  and  $S_3$  is the same as in the case of  $S_1$  in terms of local mode analysis. For  $S_2$  this part is larger than for  $S_3$  (cf. Fig. 1). For  $S_4$  the same arguments hold as for  $S_1$  (cf. Remark 2.12). This leads us to formulate the following

*Conjecture 2.27.* For a *finite* number of levels and  $\gamma$  sufficiently large the difference between the asymptotic rate of convergence of the MLAs using  $S_1$  or  $S_4$  and  $S_2$  is smaller than that between  $S_3$  and  $S_2$ . The properties stated in Theorem 2.14, Corollary 2.18, Corollary 2.24 and Conjectures 2.25–2.27 will be confirmed by numerical experiments in § 4.

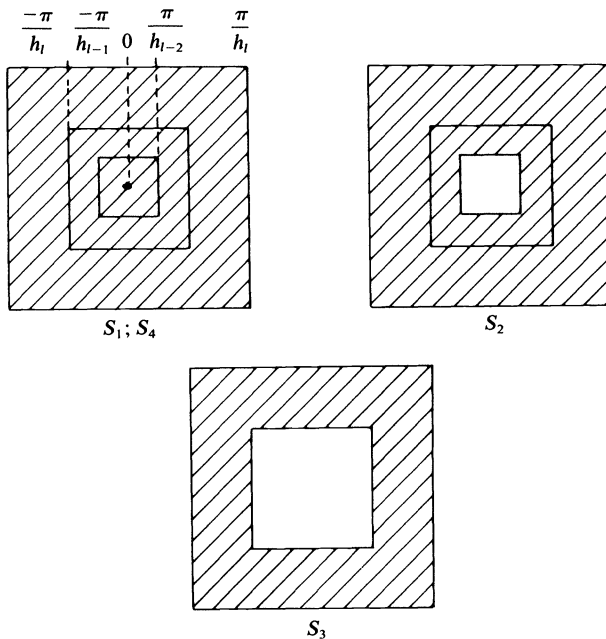


FIG. 1. Parts of  $T_l$  where for  $S_2$  and  $S_3$  the smoothing effect is the same as for  $S_1$  and  $S_4$ .



**3. Numerical approximation of the convergence rate.** In this section we give a description of the method used to determine the asymptotic rate of convergence of the MLA. Let

$$(3.1) \quad A_h v_h = f_h \text{ be a discretization of (1.1).}$$

The MLA used to solve (3.1) can be described as a defect correction process (cf. Hemker [5]):

$$(3.2) \quad \begin{aligned} &v_h^0 \text{ given start approximation,} \\ &v_h^{i+1} = M_h v_h^i + B_h^{-1} f_h, \quad i = 0, 1, \dots \end{aligned}$$

with amplification matrix  $M_h = I_h - B_h^{-1} A_h$ .  $I_h$  is the identity matrix, and  $B_h^{-1}$  is an approximate inverse of  $A_h$ , determined by coarse-grid and smoothing operators, prolongation and restriction. We suppose  $A_h$  and  $B_h$  to be nonsingular. For the error  $e_h^i = v_h - v_h^i$ ,  $i = 0, 1, \dots$  the following relation holds:

$$e_h^{i+1} = M_h e_h^i.$$

The convergence behavior of the MLA is determined by the spectral radius of  $M_h$ . This motivates the following:

**DEFINITION 3.3.** *The asymptotic rate of convergence* of the MLA (3.2) is  $-\log_{10} \rho(M_h)$  where  $\rho(M_h) \equiv \max_j |\lambda_j|$  is the spectral radius of  $M_h$ ;  $\lambda_j$  are the eigenvalues of  $M_h$ .

**THEOREM 3.4.**

$$\sup_{x \neq 0} \lim_{k \rightarrow \infty} \left( \frac{\|M_h^k x\|}{\|x\|} \right)^{1/k} = \rho(M_h),$$

with  $\|\cdot\|$  an arbitrary norm.

*Proof.* See Stoer and Bulirsch [7, (8.2.4)], Varga [8, Thm. (3.2)]. Because of Theorem 3.4 we can compute an approximation  $\rho_{m,k}(M_h, e_h^0)$  of  $\rho(M_h)$  defined by

$$(3.5) \quad \rho_{m,k}(M_h, e_h^0) \equiv \left( \frac{\|M_h^{m+k} e_h^0\|_2}{\|M_h^m e_h^0\|_2} \right)^{1/k},$$

where  $\|\cdot\|_2$  is the Euclidean norm. Note that

$$(3.6) \quad \sup_{e_h^0 \neq 0} \lim_{m,k \rightarrow \infty} \rho_{m,k}(M_h, e_h^0) = \rho(M_h).$$

In numerical computations  $v_h^j$ ,  $j = m, \dots, m+k$  are obtained by the iterative method under consideration. When for increasing  $m$  and  $k$ ,  $\|e_h^j\|_2$  reaches values near the square root of the machine accuracy, we replace  $e_h^j$  by  $e_{h,\eta}^j$ :

$$(3.7) \quad e_{h,\eta}^{(j)} \equiv \eta e_h^j (\eta \gg 1),$$

and replace  $v_h^j$  by  $v_{h,\eta}^j$ :

$$(3.8) \quad v_{h,\eta}^j \equiv v_h + e_{h,\eta}^j.$$

Thus

$$\frac{\|e_{h,\eta}^{j+1}\|_2}{\|e_{h,\eta}^j\|_2} = \frac{\|e_h^{j+1}\|_2}{\|e_h^j\|_2},$$

and as

$$(3.9) \quad \rho_{m,k}(M_h, e_h^0) = \left( \prod_{j=m}^{m+k-1} \frac{\|e_h^{j+1}\|_2}{\|e_h^j\|_2} \right)^{1/k},$$

in this way values of  $\rho_{m,k}(M_h, e_h^0)$  can be computed for large  $m$  and  $k$ . By this method ultimately the eigenfunctions of  $M_h$  corresponding to nondominant eigenvalues will decrease exponentially relative to the dominant eigenfunctions. Note that for small  $m$  and  $k$ ,  $\rho_{m,k}$  depends strongly on  $f_h$  while  $\rho$  does not. There are more refined methods to determine the spectral radius. (cf. Wilkinson [11]). However for our purpose the method described is sufficiently accurate.

**4. Numerical results.** In this section we give the results of numerical experiments to compare the strategies  $S_1, S_2, S_3$  and  $S_4$  and to verify the properties stated in Theorem 2.14 Corollaries 2.18 and 2.24 and Conjectures 2.25–2.27. We take three test problems. Test problem 1 with constant coefficients closely resembles the problem analysed by two-level analysis in Van Asselt [1]. Test problem 2 has variable coefficients. Although a strict application of Fourier analysis arguments does not hold for these variable coefficient problems, the experiments for the latter test problem show that globally the same properties hold as for the constant-coefficient case. For the second problem we also show to what extent the strategies  $S_1, \dots, S_4$  are better than relaxation alone (i.e., without coarse-grid correction). Test problem 3 differs from Test problem 1 by discretization (FEM), relaxation (ILU) and number of levels.

*Test problem 1.* We consider the following convection-diffusion equation (see Fig. 2)

$$(4.1) \quad \begin{aligned} & -(\varepsilon + \gamma h)\Delta u + \frac{\partial}{\partial y} u = 0 \quad \text{on } \Omega = [0, 1] \times [-1, 1], \\ & \varepsilon = 10^{-6}, \quad h = \frac{1}{16}. \end{aligned}$$

The boundary conditions are:

$$(4.2) \quad \begin{aligned} u|_{\delta_1\Omega} &= \begin{cases} 1, & 0 \leq x < \frac{1}{2} - 10^{-6}, \\ -10^6(x - \frac{1}{2}), & \frac{1}{2} - 10^{-6} \leq x \leq \frac{1}{2} + 10^{-6}, \\ -1, & \frac{1}{2} + 10^{-6} < x \leq 1, \end{cases} \\ \frac{\partial u}{\partial n} \Big|_{\delta_2\Omega} &= \frac{\partial u}{\partial n} \Big|_{\delta_3\Omega} = \frac{\partial u}{\partial n} \Big|_{\delta_4\Omega} = 0, \end{aligned}$$

with  $\delta_1\Omega, \dots, \delta_4\Omega$  in Fig. 2.

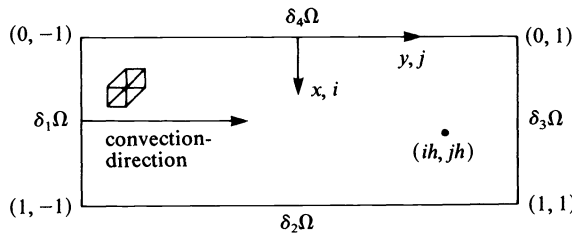


FIG. 2. The domain  $\Omega$ .

Equation (4.1) is discretized by CD on levels  $k=0, \dots, l=3$  with meshsize  $h_k = 1/2^{k+1}$ . The boundary conditions are not substituted. The Dirichlet boundary

conditions are implemented with a large number on the main diagonal to avoid unwanted coarse-grid corrections at the boundary. The Neumann boundary conditions are discretized as follows:

$$\begin{aligned} \delta_2\Omega: u(1, y) - u(1 - h_k, y) &= 0, & -1 < y \leq 1, \\ \delta_3\Omega: u(x, 1) - u(x, 1 - h_k) &= 0, & 0 < x < 1, \\ \delta_4\Omega: u(0, y) - u(h_k, y) &= 0, & -1 < y \leq 1, \quad k = 0, \dots, l = 3. \end{aligned}$$

For various values of  $\gamma$  the discretized equation is solved with the  $W$ -cycle MLA (i.e., the application of 2 multi-level-iteration steps to approximate the solution of the coarse-grid equation).

We perform one pre- and one post-relaxation step consisting of symmetric point Gauss-Seidel relaxation (SGS) in the  $y$ -direction. We use 7-point prolongation and 7-point restriction (cf. Hemker [6], Wesseling [9]). On the coarsest level we solve exactly. A random initial approximation of the solution is used. The values for  $m$  and  $k$  in (3.9) are 30 and 10 respectively.

*Test problem 2.* We consider the following convection-diffusion equation (see Fig. 3)

$$\begin{aligned} (4.3) \quad & -(\varepsilon + \gamma h)\Delta u + b_1 \frac{\partial}{\partial x} u + b_2 \frac{\partial}{\partial y} u = 0 \quad \text{on } \Omega = [0, 1] \times [-1, 1], \\ & \varepsilon = 10^{-6}, \quad h = \frac{1}{16}, \quad b_1 = y(1 - x^2), \quad b_2 = -x(1 - y^2). \end{aligned}$$

The boundary conditions are

$$(4.4) \quad \begin{aligned} & u|_{\delta_1\Omega} = 1 + \tanh(10 + 20x), \quad -1 \leq x \leq 0, \\ & \frac{\partial u}{\partial n} \Big|_{\delta_2\Omega} = \frac{\partial u}{\partial n} \Big|_{\delta_3\Omega} = \frac{\partial u}{\partial n} \Big|_{\delta_4\Omega} = \frac{\partial u}{\partial n} \Big|_{\delta_5\Omega} = 0. \end{aligned}$$

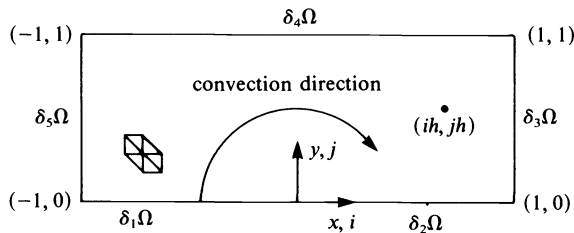


FIG. 3. The domain  $\Omega$ .

Equations (4.3) and (4.4) are discretized by the FEM on levels  $k = 0, \dots, l = 4$  with mesh-size  $h_k = (1/2)^k$ . The boundary conditions are not substituted and the Dirichlet boundary conditions are implemented with a large number on the main diagonal. For different values of  $\gamma$ , and  $S_1$ - $S_4$  the discretized equation is solved with the  $W$ -cycle MLA. We perform one pre- and one post-relaxation step by means of 7-point ILU relaxation, (cf. Wesseling and Sonneveld [10]). The ILU-decomposition is ordered lexicographically (cf. Fig. 3). On the coarsest level we solve exactly. Again we use 7-point prolongation and 7-point restriction (that are consistent with the FEM discretization), and a random initial approximation. In (3.9)  $m$  and  $k$  are again 30 and 10.

*Test problem 3.* For  $l = 4, 5, 6$ , we consider (4.1) with different  $h$ , and (4.2) discretized by the FEM on levels  $k = 0, \dots, l$ , with mesh size  $h_k = (1/2)^{k+1}$ ,  $\gamma = 1/2$ . The boundary

conditions are not substituted and the Dirichlet boundary conditions are implemented with a large number on the main diagonal.

The discretized equation is solved with the  $W$ -cycle MLA. We perform one pre- and post-relaxation step by means of 7-point-ILU relaxation (on the coarsest level we do not solve directly, but perform 2 relaxation sweeps). The ILU-decomposition is ordered lexicographically (cf. Fig. 3). We use 7-point prolongation and 7-point restriction. A random initial approximation of the solution is used. The values for  $m$  and  $k$  in (3.9) are 20 and 10 respectively.

Figures 4 and 5 show the properties in Conjectures (2.25)–(2.27) for Test problems 1 and 2, respectively. Figure 5 also shows that all strategies  $S_1$ – $S_4$  are better than relaxations without coarse-grid corrections. In Table 1 for  $S_1$ ,  $S_2$  and  $S_3$  the smoothing factors of SGS are given at different levels and for different  $\gamma$ . We notice that for  $\log_2 \gamma > 0$  the big difference in the asymptotic rate of convergence of  $S_2$  and  $S_3$  (cf. Fig. 4) is mainly caused by the order of consistency and to a small extent by the relaxation method, because the smoothing factors are almost the same.

In order to demonstrate Theorem 2.14, Corollaries 2.18 and 2.24 in connection with Remark 2.7 we take Test problem 3. Table 2 shows the convergence rates as measured (cf. Definition 3.3). Note that  $S_1$  and  $S_4$  show similar stability and convergence behavior (cf. Remark 2.12).

*Remark 4.5.* With respect to Remark 2.7 we notice that in many cases a decreasing stability coincides with a worsening smoothing factor (cf. Table 1).

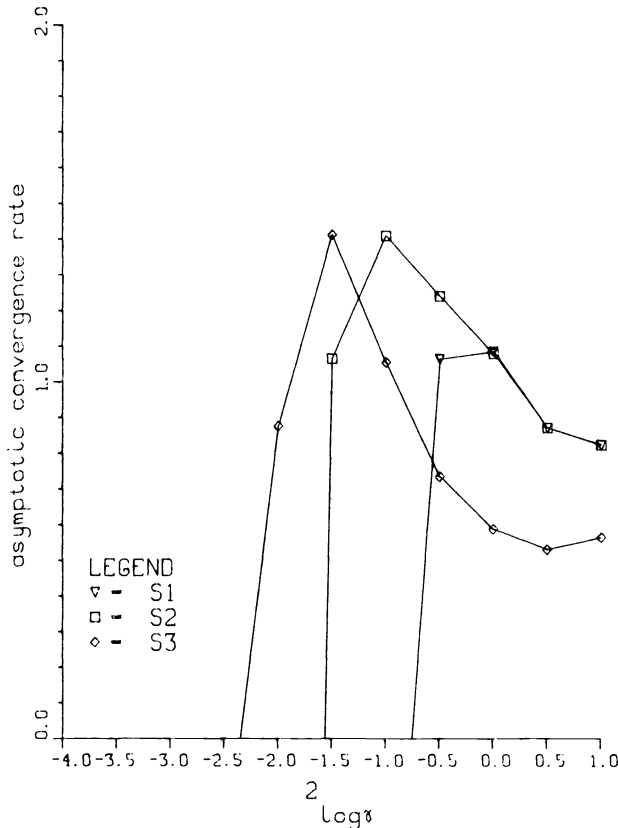


FIG. 4. Asymptotic convergence rates for Test problem 1. Only the part of the figure with positive asymptotic convergence rate is drawn. ( $l=3, h_l = \frac{1}{16}$ ).

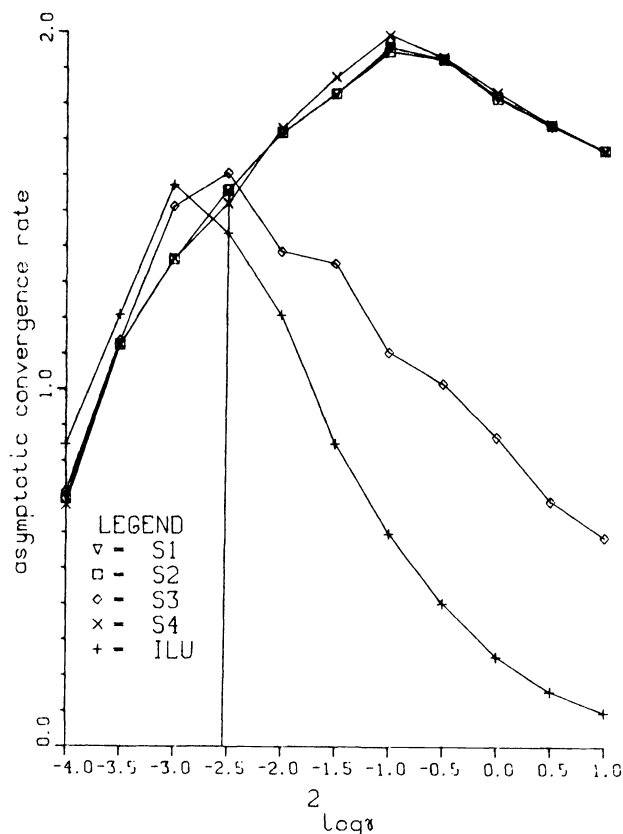


FIG. 5. Asymptotic convergence rates for Test problem 2. The graph depicted by "+" represents two ILU relaxation sweeps in one iteration step without coarse-grid correction. Only the part of the figure with positive asymptotic convergence rate is drawn. ( $l=4$ ,  $h_1 = \frac{1}{16}$ ).

**5. Conclusions.** In order to solve the convection-diffusion equation in two dimensions by a multi-level algorithm (MLA), we consider 4 strategies for coarse-grid operators:

- $S_1$ : on each coarse grid the same artificial viscosity as on the finest grid;
- $S_3$ : on each coarse grid the artificial viscosity corresponding to the mesh width;
- $S_2$ : an intermediate choice, with the same artificial viscosity on the two finest grids;
- $S_4$ : Galerkin approximation for the coarse-grid operators.

For  $S_1$  and  $S_4$  the artificial viscosity may become too small on coarse grids, and hence stability problems and bad smoothing-factors may occur.  $S_1$  and  $S_4$  are not  $\varepsilon$ -asymptotically stable,  $S_2$  and  $S_3$  are. (cf. Definition 2.6, Theorem 2.14, Corollaries 2.18, 2.24 and Table 1).

If the finest-grid artificial viscosity is sufficiently large, the asymptotic rate of convergence of the MLA according to  $S_2$  is far better than that of  $S_3$  (cf. Conjecture 2.26 and Figs. 4, 5).

**Acknowledgments.** The authors would like to thank Dr. P. W. Hemker (Mathematisch Centrum, Amsterdam) and Prof. Dr. Ir. P. Wesseling (Department of Mathematics, Delft University) for their constructive comments and careful reading of the manuscript.

TABLE 1  
Smoothing-factors for one SGS sweep, Test problem 1, different  $\gamma$ , levels and strategies  
(local mode analysis, cf. Brandt [2]).

$k \backslash S$	$S_1$	$S_2$	$S_3$
3	0.36	0.36	0.36
2	4.84	4.84	0.36
1	186	4.84	0.36

$\log_2 \gamma = -1.5$

$k \backslash S$	$S_1$	$S_2$	$S_3$
3	0.24	0.24	0.24
2	0.80	0.80	0.24
1	15625	0.80	0.24

$\log_2 \gamma = -1.0$

$k \backslash S$	$S_1$	$S_2$	$S_3$
3	0.23	0.23	0.23
2	0.36	0.36	0.23
1	4.84	0.36	0.23

$\log_2 \gamma = -0.5$

$k \backslash S$	$S_1$	$S_2$	$S_3$
3	0.24	0.24	0.24
2	0.24	0.24	0.24
1	0.80	0.24	0.24

$\log_2 \gamma = 0.0$

$k \backslash S$	$S_1$	$S_2$	$S_3$
3	0.24	0.24	0.24
2	0.23	0.23	0.24
1	0.36	0.23	0.24

$\log_2 \gamma = 0.5$

$k \backslash S$	$S_1$	$S_2$	$S_3$
3	0.25	0.25	0.25
2	0.24	0.24	0.25
1	0.24	0.24	0.25

$\log_2 \gamma = 1.0$

TABLE 2  
Convergence rates for Test problem 3,  $S_1$ - $S_4$ , and increasing  $l$ .

level $l$	$h_l$	strategy			
		$S_1$	$S_2$	$S_3$	$S_4$
4	1/32	2.01	1.78	1.61	2.01
5	1/64	$\ll 0$	1.70	1.33	$\ll 0$
6	1/128	$\ll 0$	1.17	0.87	$\ll 0$

## REFERENCES

- [1] E. J. VAN ASSELT, *The multi-grid method and artificial viscosity*, Multigrid Methods, Proc. Conference held at Köln-Porz, November 23-27, 1981, W. Hackbusch and U. Trottenberg, eds., Lecture Notes in Mathematics 960, Springer-Verlag, Berlin, 1982, pp. 313-327.
- [2] A. BRANDT, *Multi-level adaptive solutions to boundary-value problems*, Math. Comp., 31 (1979), pp. 333-390.
- [3] ———, *Guide to multigrid development*, Multigrid Methods Proc. Conference held at Köln-Porz, November 23-27, 1981, W. Hackbusch and U. Trottenberg, eds., Lecture Notes in Mathematics 960, Springer-Verlag, Berlin, 1982, pp. 220-312.
- [4] P. W. HEMKER, *Mixed defect correction iteration for the accurate solution of the convection diffusion equation*, Multigrid Methods. Proc. Conference held at Köln-Porz, November 23-27, 1981, W. Hackbusch and U. Trottenberg, eds., Lecture Notes in Mathematics 960, Springer-Verlag, Berlin, 1982, pp. 485-502.

- [5] P. W. HEMKER, *Introduction to multigrid methods*, Nieuw Archief voor Wiskunde (3), XXIX (1981), pp. 71-101.
- [6] ———, *Fourier analysis of gridfunctions, prolongations and restrictions*, Rep. NW 93/80, Mathematical Centre, Amsterdam, 1980.
- [7] J. STOER AND R. BULIRSCH, *Einführung in die Numerische Mathematik II*, Springer-Verlag, Berlin, 1973.
- [8] R. S. VARGA, *Matrix Iterative Analysis*, Prentice-Hall, Englewood Cliffs, NJ, 1962.
- [9] P. WESSELING, *Theoretical and practical aspects of a multigrid method*, this Journal, 3 (1982), pp. 387-407.
- [10] P. WESSELING AND SONNEVELD P. *Numerical experiments with a multiple grid and a preconditioned Lanczos type method*, in Approximation methods for Navier-Stokes problems, Proc, Paderborn 1979, R. Rautmann, ed., Lecture Notes in Mathematics 771, Springer-Verlag, Berlin, 1980, pp. 543-562.
- [11] J. H. WILKINSON, *The Algebraic Eigenvalue Problem*, Clarendon Press, Oxford, 1965.

## EVALUATION OF THE NONCENTRAL $F$ -DISTRIBUTION BY NUMERICAL CONTOUR INTEGRATION\*

CARL W. HELSTROM† AND JAMES A. RITCEY†

**Abstract.** The noncentral  $F$ -distribution, which yields the power function of tests of linear hypotheses as in the analysis of variance and of covariance, is computed by numerical integration of a contour integral in the complex plane. Determining the percentage points of this distribution is facilitated by approximating the integral by Laplace's method during the initial stages of a search by the secant method.

**Key words.**  $F$ -distribution, beta distribution, contour integration

**AMS (MOS) subject classifications.** 65U05, 65R10

**1. The noncentral  $F$ -distribution.** The noncentral  $F$ -distribution originates in a common hypothesis-testing problem. Under the null hypothesis  $H_0$ ,  $f_1$  independent normal random variables  $x_1, x_2, \dots, x_{f_1}$  have known means, which we can take as zero; under the alternative  $H_1$  their means  $m_1, m_2, \dots, m_{f_1}$  differ from zero. The unknown common variance  $\sigma^2$  of these variables is estimated from observations of  $f_2$  normal random variables  $y_1, y_2, \dots, y_{f_2}$  of mean zero and equal variance  $\sigma^2$ ; the  $y$ 's are independent among themselves and of the  $x$ 's. The alternative hypothesis  $H_1$  is chosen when  $X > 0$ , where

$$(1.1) \quad X = \frac{1}{2} \sum_{k=1}^{f_1} x_k^2 - \frac{1}{2} \omega \sum_{m=1}^{f_2} y_m^2,$$

the factor  $\omega$  having been set to yield a preassigned size  $\alpha$  of the hypothesis test, or false-alarm probability,

$$\alpha = \Pr(X > 0 | H_0).$$

To be calculated are the value of  $\omega$  and the power  $\beta$  of the test, which is the probability of rejecting the null hypothesis  $H_0$  when the alternative  $H_1$  is true,

$$(1.2) \quad \beta = q(\lambda, \omega; f_1, f_2) = \Pr(X > 0 | H_1);$$

$\beta$  is a function of the mean values  $m_k$ ,  $1 \leq k \leq f_1$ , but happens to depend on them only through the noncentrality parameter

$$(1.3) \quad \lambda = \frac{1}{2} \sum_{k=1}^{f_1} \frac{m_k^2}{\sigma^2}.$$

This problem arises in the analysis of variance and of covariance and more generally in tests of linear hypotheses about observed data [1]-[4]. It also arises in radar signal detection, in which the presence or absence of an echo signal from a target at a particular range is decided on the basis of the sum of the filtered and quadratically rectified inputs to the radar receiver during time intervals following  $a = \frac{1}{2} f_1$  transmitted pulses. This sum is compared with  $\omega$  times an estimate of the noise variance based on similarly filtered, rectified, and summed receiver inputs during a total of  $b = \frac{1}{2} f_2$  intervals

---

\* Received by the editors July 11, 1983. This research was sponsored by the U.S. Air Force Office of Scientific Research, Air Force Systems Command, under grant AFOSR-82-0343.

† Department of Electrical Engineering and Computer Sciences, University of California, San Diego, La Jolla, California, 92093.



known or assumed to contain no signals [5]–[9]. In the application to radar  $f_1 = 2a$  and  $f_2 = 2b$  are even numbers, and  $b$  is usually an integral multiple of  $a$ .

In statistics the quantity  $f_2\omega/f_1$  is customarily denoted by  $F$ , and the distribution of probabilities  $q$  for  $\lambda \neq 0$  is called the “noncentral  $F$ -distribution.” To keep our expressions simple we deal with  $\omega$  instead, and we set  $a = \frac{1}{2}f_1$ ,  $b = \frac{1}{2}f_2$ . Under the null hypothesis ( $\lambda = 0$ ) the related statistic  $x = \omega/(\omega + 1)$  has the beta-distribution [10]

$$(1.4) \quad \begin{aligned} f_\beta(x) &= [B(a, b)]^{-1} x^{a-1} (1-x)^{b-1}, \\ B(a, b) &= \frac{\Gamma(a)\Gamma(b)}{\Gamma(a+b)}, \quad 0 < x < 1. \end{aligned}$$

Tables of the percentage points of  $F$  and  $x$  for  $\lambda = 0$  exist [10], [11], and Pearson [12] has tabulated the associated cumulative probability.

The bilateral  $t$ -distribution with  $f$  degrees of freedom is a special case:  $f_1 = 1$ ,  $f_2 = f$ ,  $\omega = t_0^2/f$ , where  $t_0$  is the decision level for the bilateral  $t$ -test with size

$$\alpha = \Pr(|t| > t_0 | H_0) = q(0, t_0^2/f; 1, f).$$

Here  $t = \mu/s$  with

$$s^2 = f^{-1} \sum_{k=1}^f x_k^2,$$

where  $x_1, x_2, \dots, x_f$  and  $\mu$  are independent normal random variables with equal variances  $\sigma^2$ . The  $f$  variables  $x_1, x_2, \dots, x_f$  always have mean zero. Under the null hypothesis  $\mu$  has mean zero; under the alternative it has mean  $\bar{\mu}$ . The power of the test is then

$$\beta = \Pr(|t| > t_0 | H_1) = q(\lambda, t_0^2/f; 1, f)$$

and the noncentrality parameter is  $\lambda = \bar{\mu}^2/2\sigma^2$ .

Computation of the power function  $q(\lambda, \omega; 2a, 2b)$  has been treated by a number of writers [2], [4], [5], [8], [13]–[15], but only sparse tables exist [2], [4], [16]. When  $b = \frac{1}{2}f_2$  is an integer, finite algorithms requiring the summation of  $b$  terms whose values can be computed recursively are available [2], [13]–[15]. A general series representation has been given by Robertson [8],

$$(1.5) \quad \begin{aligned} 1 - q(\lambda, \omega; 2a, 2b) &= e^{-\lambda} x^a (1-x)^b \sum_{n=0}^{\infty} \frac{\Gamma(a+b+n)}{\Gamma(b)\Gamma(a+1+n)} x^n \sum_{i=0}^n \frac{\lambda^i}{i!}, \\ x &= \frac{\omega}{\omega + 1}, \end{aligned}$$

which holds for nonintegral values of  $a$  and  $b$  as well. The terms of this series can be computed recursively,

$$1 - q(\lambda, \omega; 2a, 2b) = \sum_{n=0}^{\infty} f_n h_n,$$

where

$$(1.6) \quad f_0 = \frac{x^a(1-x)^b}{aB(a, b)}, \quad f_n = \frac{a+b+n-1}{a+n} x f_{n-1},$$

$$(1.7) \quad h_0 = d_0 = e^{-\lambda}, \quad h_n = h_{n-1} + d_n, \quad d_n = \frac{\lambda}{n} d_{n-1}.$$

By rearranging the order of summation we obtain an alternative algorithm

$$(1.8) \quad q(\lambda, \omega; 2a, 2b) = \alpha + \sum_{n=1}^{\infty} d_n g_n$$

with  $d_n$  computed as in (1.7) and

$$(1.9) \quad g_1 = f_0, \quad g_{n+1} = g_n + f_n,$$

the  $f_n$ 's being computed as in (1.6). The value of the size  $\alpha$  needed for (1.8) can be determined from [8]

$$(1.10) \quad \begin{aligned} \alpha &= q(0, \omega; 2a, 2b) = 1 - q(0, \omega^{-1}; 2b, 2a) \\ &= x^a (1-x)^b \sum_{n=0}^{\infty} \frac{\Gamma(a+b+n)}{\Gamma(a)\Gamma(b+1+n)} (1-x)^n, \quad x = \frac{\omega}{\omega+1}. \end{aligned}$$

For the sake of accuracy one should compute whichever probability  $q$  or  $1-q$  is the smaller.

Tang's algorithm [2], which evaluates the probability  $1-\beta$  for integral values of  $b$ , sums  $b$  terms that are computed recursively; and when  $b \gg 1$ , round-off error may introduce a large relative error into the value of  $\beta$  when  $\beta$  is near 0. Both Tang's algorithm and Robertson's [8] may suffer from underflow or overflow when the noncentrality parameter  $\lambda$  is large because of the presence of the factor  $e^{-\lambda}$ . For  $f_1 = f_2 = 7$ , for instance, and  $\alpha = 10^{-4}$  ( $\omega = 30.477$ ), the value of  $\lambda$  for  $\beta = 0.999$  equals 376.72; in cases such as this, special arrangements to handle  $e^{-\lambda}$  and the partial exponential series  $\sum_i \lambda^i / i!$  in (1.5) are necessary. Robertson [8] pointed out that for the sake of accuracy one ought to compute the partial exponential series in (1.5) by starting with its largest term and summing in both directions when  $n \gg \lambda \gg 1$ ; this precludes using the simple recursions in (1.6) and (1.7).

Because of these difficulties, we have investigated a method of computing the central and noncentral  $F$ -distributions by numerical contour integration in the complex plane; it is described in § 2. Good accuracy over a broad range of parameters can be attained with relatively few steps of numerical integration, and a bound can be set on the truncation error that permits a simple rule for stopping the numerical integration. The method resembles that used by Lugannani and Rice for computing the density function of the ratio of quadratic forms in normal variates [17], of which the noncentral  $F$ -distribution can be viewed as a special case. Their concern was focussed, however, on quadratic forms built on nondiagonal and indefinite matrices, and they did not treat computation of the cumulative distribution in any detail. In the final section we show how the search for the value of the noncentrality parameter  $\lambda$  yielding a prescribed power  $\beta$ , or for the value of  $\omega$  yielding a prescribed size  $\alpha$ , can be facilitated by beginning with a Laplace approximation to the contour integral.

**2. The contour integration.** The power  $\beta$  of the hypothesis test is equal to  $\Pr(X/\sigma^2 > 0)$ , and we calculate it from a contour integral for the inverse Laplace transform of the moment-generating function (m.g.f.) of  $\xi = X/\sigma^2$ . This m.g.f. is easily derived from those of the random variables  $\frac{1}{2}x_k^2$  and  $\frac{1}{2}y_m^2$  appearing in (1.1),

$$(2.1) \quad \begin{aligned} E[\exp(-\frac{1}{2}zx_k^2)] &= (1 + \sigma^2 z)^{-1/2} \exp\left(-\frac{m_k^2 z}{2(1 + \sigma^2 z)}\right), \\ E[\exp(-\frac{1}{2}zy_m^2)] &= (1 + \sigma^2 z)^{-1/2}, \end{aligned}$$

and because these random variables are independent,

$$(2.2) \quad h(z) = E(e^{-z\xi}) = (1+z)^{-a}(1-\omega z)^{-b} \exp\left(-\frac{\lambda z}{1+z}\right),$$

with the noncentrality parameter  $\lambda$  given as in (1.3). The probability density function of  $\xi = X/\sigma^2$  is then

$$(2.3) \quad f_\xi(\xi) = \int_{c-i\infty}^{c+i\infty} h(z) e^{z\xi} \frac{dz}{2\pi i}$$

by the inversion formula for Laplace transforms; the contour of integration runs parallel to the imaginary axis and crosses the real axis between the singularities of  $h(z)$  at  $-1$  and  $\omega^{-1}$ . The power  $\beta$  of the hypothesis test is then

$$(2.4) \quad \begin{aligned} \beta &= q(\lambda, \omega; 2a, 2b) = \int_0^\infty f_\xi(\xi) d\xi \\ &= - \int_{c-i\infty}^{c+i\infty} z^{-1}(1+z)^{-a}(1-\omega z)^{-b} \exp\left(-\frac{\lambda z}{1+z}\right) \frac{dz}{2\pi i}, \quad -1 < c < 0. \end{aligned}$$

We take  $c = \text{Re } z < 0$  in order to integrate  $e^{z\xi}$  in (2.3) over  $0 < \xi < \infty$ . One uses (2.4) for values of  $q < \frac{1}{2}$ . For  $q > \frac{1}{2}$  one calculates instead the complementary probability

$$(2.5) \quad \begin{aligned} 1 - q(\lambda, \omega; 2a, 2b) &= \int_{-\infty}^0 f_\xi(\xi) d\xi \\ &= \int_{c-i\infty}^{c+i\infty} z^{-1}(1+z)^{-a}(1-\omega z)^{-b} \exp\left(-\frac{\lambda z}{1+z}\right) \frac{dz}{2\pi i}, \quad 0 < c < \omega^{-1}. \end{aligned}$$

The magnitude of the integrand decreases along the contour of integration most rapidly in each direction from the real axis if the contour passes through a saddlepoint of the integrand. Saddlepoints exists for (2.4) and (2.5) at  $z = u'_0$  and  $z = u''_0$ , respectively,  $-1 < u'_0 < 0$ ,  $0 < u''_0 < \omega^{-1}$ , which lie at minima of the ‘‘phase’’

$$(2.6) \quad \Phi(z) = \ln [(\pm z)^{-1} h(z)] = -a \ln(1+z) - b \ln(1-\omega z) - \ln(\pm z) - \frac{\lambda z}{1+z}$$

of the integrand as  $z$  moves along the real axis from  $-1$  to  $\omega^{-1}$ . Here  $(\pm z) = -z$  when (2.4) is being integrated and the saddlepoint is at  $z = u'_0 < 0$ ;  $(\pm z) = z$  when (2.5) is being integrated and the saddlepoint is at  $z = u''_0 > 0$ . These saddlepoints are the roots of

$$(2.7) \quad \begin{aligned} \Phi'(u) &= -\frac{a}{1+u} + \frac{\omega b}{1-\omega u} - \frac{1}{u} - \lambda(1+u)^{-2} = 0, \\ u &= u'_0, \quad -1 < u'_0 < 0, \quad u = u''_0, \quad 0 < u''_0 < \omega^{-1}. \end{aligned}$$

They can be expeditiously calculated by Newton’s method, a trial value of  $u_0$  being replaced by

$$(2.8) \quad u_0 \leftarrow u_0 - \frac{\Phi'(u_0)}{\Phi''(u_0)},$$

where  $\Phi''(u)$  is the second derivative of the phase in (2.6).

An alternative method for locating the saddlepoint sets  $v = \text{Im } z$  equal to some small positive number  $\varepsilon$  and searches for the value of  $u = \text{Re } z$  at which  $\text{Im } \Phi(u + i\varepsilon) =$

$f(u) = 0$ . The secant method can be used, a trial value  $u_0$  being replaced by

$$(2.9) \quad u_0 \leftarrow u_0 - \frac{f(u_0)\Delta u}{f(u_0 + \Delta u) - f(u_0)}$$

with  $\Delta u$  an increment of the order of 0.1 of the width of the interval on the  $\text{Re } z$ -axis where the saddlepoint lies (1 on the left,  $\omega^{-1}$  on the right). We found  $\varepsilon = 10^{-4}$  a suitable value. The position  $u_0$  of the saddlepoint is not needed to high accuracy. This method utilizes the same routine as is needed to compute the values of the integrand  $\exp \Phi(z)$  along the path of integration and thereby economizes on memory occupancy, a consideration when the computations are being carried out on a hand-held programmable calculator.

Evaluation of cumulative probabilities by numerical integration along a contour in the complex plane was demonstrated by Rice [18], who used contours passing through the saddlepoint not of  $z^{-1}h(z)$ , but of  $h(z)$ , and limited the applicability of the method to the far tails of the probability density function; see [19]. As in [18], we carried out the numerical integration by the trapezoidal rule, whose advantage for numerical integration of infinite integrals of analytic functions has been described by Rice [20]. As in [18] the initial step size was taken as

$$\Delta v = [\Phi''(u_0)]^{-1/2},$$

and the step size was halved until the result stabilized to the number of significant figures desired. When the alternative method (2.9) for locating the saddlepoint is used, the value of the second derivative  $\Phi''(u_0)$  can be estimated with sufficient accuracy from

$$(2.10) \quad \Phi''(u_0) \cong 2 \text{Re} [\Phi(u_0) - \Phi(u_0 + iv)]/v^2$$

by taking  $v$  of the order of the interval  $\Delta v$  anticipated.

Because of the symmetry of the integrand and of the path of integration, the integral to be evaluated numerically can be expressed as

$$(2.11) \quad q(\lambda, \omega; 2a, 2b) = -\pi^{-1} \text{Re} \int_0^\infty z^{-1}(1+z)^{-a}(1-\omega z)^{-b} \exp\left(-\frac{\lambda z}{1+z}\right) dv \\ + \begin{cases} 1, & u_0 > 0, \\ 0, & u_0 < 0, \end{cases} \quad z = u_0 + iv.$$

It is shown in the appendix that when the integration is stopped at a point  $z_1 = u_0 + iv_1$ , the resulting truncation error is bounded by

$$(2.12) \quad \mathcal{E} \leq \frac{|1+z_1||1-\omega z_1|}{2\omega v_1[(a-1)(b-1)]^{1/2}}|I|,$$

where  $|I|$  is the absolute value of the integrand at the point  $z_1$ . Because the factor multiplying  $|I|$  is usually rather less than 1 for  $a \gg 1$ ,  $b \gg 1$ , the simple rule that the numerical integration be halted when the magnitude  $|I|$  of the integrand itself falls below a prescribed fraction of the accumulated sum times  $\Delta v$  is adequate in most instances. A program utilizing this rule was written for a hand-held calculator.

Table 1 lists some results of the numerical contour integration for  $f_1 = 15$ ,  $\alpha = 0.01$ , and  $f_2 = 15$  and 45. The integration was stopped when the ratio of the absolute value  $|I|$  of the integrand to  $\Delta v$  times the accumulated sum fell below  $10^{-7}$ . The column marked "bound factor" lists the values of the factor multiplying  $|I|$  in (2.12) and indicates that the relative truncation error is of the order of  $10^{-8}$  or less. The column labeled "series" gives the results of summing (1.8) or (1.5) by the simple recursion

TABLE 1  
 $1 - \beta$ : Noncentral  $F$ -distribution  $f_1 = 15, \alpha = 0.01$ .

$f_2 = 15, \omega = F = 3.52219$					
$\lambda$	Numerical integration	Number of steps	Bound factor	Series (1.8) or (1.5)	Number of terms
20	0.442289617	14	0.104		
	0.442226950	28	0.104	0.442226930	123
	0.442226946	55	0.103		
40	6.14287653 (-2)	12	0.101		
	6.14278071 (-2)	23	0.100	6.14278044 (-2)	132
	6.14278071 (-2)	46	0.100		
60	4.42491611 (-3)	12	0.0999		
	4.42480089 (-3)	23	0.0996	4.42480072 (-3)	145
	4.42480089 (-3)	45	0.0994		
80	2.16901762 (-4)	12	0.0993		
	2.16894276 (-4)	24	0.0993	2.16894267 (-3)	159
	2.16894276 (-4)	47	0.0992		
$f_2 = 45, \omega = 0.82139, F = 2.46418$					
10	0.572418264	11	0.0938		
	0.572486848	22	0.0938	0.572486853	34
	0.572486853	43	0.0938		
20	0.122785853	9	0.0938		
	0.122781876	17	0.0939	0.122781875	62
	0.122781876	33	0.0941		
30	1.27422435 (-2)	8	0.0943		
	1.27422097 (-2)	16	0.0943	1.27422096 (-2)	66
	1.27422097 (-2)	31	0.0946		
40	8.06069499 (-4)	8	0.0945		
	8.06069144 (-4)	15	0.0954	8.06069135 (-4)	71
	8.06069143 (-4)	30	0.0954		

method represented by (1.6), (1.7), and (1.9); summation was stopped when the ratio of the last included term to the accumulated sum fell below  $10^{-8}$ . Because all the terms are positive, the relative error of the series must be at least of this size. The number of steps of numerical integration and the number of terms in the series are also listed.

**3. The inverse  $F$ -distribution.** When searching for the value of  $\omega = f_1 F / f_2$  yielding a prescribed size or false-alarm probability  $\alpha$ , or for the value of the noncentrality parameter  $\lambda$  yielding a preassigned power  $\beta = q(\lambda, \omega; f_1, f_2)$ , it is most efficient to begin with a crude, but quickly calculated approximation to  $q(\lambda, \omega; f_1, f_2)$  or its complement. Such an approximation is obtained by applying Laplace's method [21], [22] to the integrals in (2.4) or (2.5), as the case may be,

$$(3.1) \quad \left. \begin{aligned} -1 < u_0 < 0: q(\lambda, \omega; 2a, 2b) \\ 0 < u_0 < \omega^{-1}: 1 - q(\lambda, \omega; 2a, 2b) \end{aligned} \right\} \cong [2\pi\Phi''(u_0)]^{-1/2} \exp \Phi(u_0),$$

in which the saddlepoint  $u_0 = u'_0$  or  $u''_0$  is calculated by Newton's method as in (2.7)–(2.8). The second derivative  $\Phi''(u_0)$  needed in (3.1) is generated in the course of this computation, and the phase  $\Phi(u_0)$  is obtained from (2.6). Alternatively  $\Phi''(u_0)$  can

be calculated as in (2.10). One can then apply the secant method to (3.1) until the value of  $\omega$  or  $\lambda$  stops changing by more than, say, one percent, whereupon one switches to the computation of values of  $q(\lambda, \omega; 2a, 2b)$  or its complement by numerical integration of (2.11), continuing until a sufficiently precise value of  $\omega$  or  $\lambda$  is reached.

The starting value of  $\lambda$  in this procedure can be obtained from the normal approximation to the statistic  $\xi = X/\sigma^2$ , whose mean and variance are obtained by expanding  $\ln h(z)$  in the neighborhood of  $z = 0$ ,

$$(3.2) \quad E(\xi) = a - \omega b + \lambda,$$

$$(3.3) \quad \text{Var } \xi = a + \omega^2 b + 2\lambda,$$

so that

$$(3.4) \quad q(\lambda, \omega; 2a, 2b) \cong Q(\eta) = (2\pi)^{-1/2} \int_{\eta}^{\infty} e^{-t^2/2} dt,$$

$$(3.5) \quad \eta = (a + \omega^2 b + 2\lambda)^{-1/2}(\omega b - a - \lambda).$$

Thus with  $\eta$  the normal variate defined in (3.4), it is only necessary to solve the quadratic equation resulting from (3.5) for  $\lambda$ . Table 2 shows values of  $\lambda$  obtained by successively applying the normal and saddlepoint approximations, along with the final values of  $\lambda$  in typical cases. At most three iterations of the saddlepoint approximation and two of the numerical integration provided values of  $\lambda$  yielding the preassigned values of the power  $\beta$  to six significant figures.

TABLE 2  
Percentage points, noncentral F-distribution.

$f_1 = 15, f_2 = 15, \alpha = 10^{-4}, \omega = F = 8.22898$			
Power $q$	Normal Approx.	Saddlepoint Approx.	Numerical Integration
0.1	23.8095	26.1428	25.8389
0.5	54.2174	53.7850	52.1010
0.9	87.9100	87.6929	88.6118
0.99	117.9838	126.3875	126.6475
0.999	141.5369	159.3143	159.4327
$f_1 = 15, f_2 = 31, \alpha = 10^{-4}, \omega = 2.35053, F = 4.85777$			
0.1	14.7421	15.3036	15.1008
0.5	28.9333	30.0508	28.7027
0.9	46.4093	45.9502	46.3452
0.99	63.4400	63.6918	63.8256
0.999	77.6389	78.3439	78.4184

To obtain a starting value for  $\omega$  for fixed size  $\alpha$  one can similarly try solving the normal approximation

$$\omega b - a = \eta_0(a + \omega^2 b)^{1/2}$$

with the normal variate  $\eta_0$  such that  $Q(\eta_0) = \alpha$ , but no real solution exists when  $b > \eta_0^2$ . We found that the initial value of  $\omega$  can be adequately taken as

$$\omega = (a + a^{1/2} \eta_0) / b$$

when this is less than or of the order of 1. When this turns out to be rather larger than 1, the approximation

$$\omega = [bB(a, b)\alpha]^{-1/b} - 1,$$

resulting from taking only the first term of the series in (1.10), can be used instead.

**Appendix.** The integral in (2.11) is to be evaluated by numerical integration along a vertical contour on which  $z = u_0 + iv$ . The error incurred by cutting off the integration at the point  $z_1 = u_0 + iv_1$  has absolute value

$$(A.1) \quad \mathcal{E} = \pi^{-1} \left| \int_{v_1}^{\infty} z^{-1}(1+z)^{-a}(1-\omega z)^{-b} \exp\left(-\frac{\lambda z}{1+z}\right) dv \right|,$$

and to this we seek an upper bound.

Applying the Schwarz inequality, we write

$$(A.2) \quad \begin{aligned} \mathcal{E}^2 &\leq \pi^{-2} \int_{v_1}^{\infty} \left| z^{-1/2}(1+z)^{-a} \exp\left(-\frac{\lambda z}{2(1+z)}\right) \right|^2 dv \\ &\quad \times \int_{v_1}^{\infty} \left| z^{-1/2}(1-\omega z)^{-b} \exp\left(-\frac{\lambda z}{2(1+z)}\right) \right|^2 dv. \end{aligned}$$

On the contour  $z = u_0 + iv$ , for  $v \geq v_1$ ,

$$(A.3) \quad \begin{aligned} \left| \exp\left(-\frac{\lambda z}{2(1+z)}\right) \right|^2 &= \exp\left[-\lambda + \operatorname{Re}\left(\frac{\lambda}{1+z}\right)\right] \\ &= \exp\left[-\lambda + \frac{\lambda(1+u_0)}{(1+u_0)^2 + v^2}\right] \\ &\leq \exp\left[-\lambda + \frac{\lambda(1+u_0)}{(1+u_0)^2 + v_1^2}\right] = \left| \exp\left(-\frac{\lambda z_1}{1+z_1}\right) \right| \end{aligned}$$

provided that  $u_0 > -1$ , as is the case when  $u_0$  is a saddlepoint. Thus the second factor in (A.2) is bounded as

$$(A.4) \quad \begin{aligned} &\int_{v_1}^{\infty} |z|^{-1} |1-\omega z|^{-2b} \exp\left[-\operatorname{Re}\left(\frac{\lambda z}{1+z}\right)\right] dv \\ &\leq \left| \exp\left(-\frac{\lambda z_1}{1+z_1}\right) \right| \int_{v_1}^{\infty} |z|^{-1} |1-\omega z|^{-2b} dv. \end{aligned}$$

Calling the integral in (A.4)  $J_b$ , we find

$$(A.5) \quad \begin{aligned} J_b &= \int_{v_1}^{\infty} |z|^{-1} |1-\omega z|^{-2b} dv \\ &= \int_{v_1}^{\infty} (u_0^2 + v^2)^{-1/2} [(1-\omega u_0)^2 + \omega^2 v^2]^{-b} dv \\ &\leq v_1^{-1} \int_{v_1}^{\infty} (u_0^2 + v^2)^{-1/2} [1-2\omega u_0 + \omega^2(u_0^2 + v^2)]^{-b} v dv. \end{aligned}$$

Putting  $y^2 = u_0^2 + v^2$ ,  $y dy = v dv$ , we get

$$\begin{aligned}
 J_b &\leq v_1^{-1} \int_{|z_1|}^{\infty} (1 - 2\omega u_0 + \omega^2 y^2)^{-b} dy \\
 (A.6) \quad &\leq v_1^{-1} |z_1|^{-1} \int_{|z_1|}^{\infty} (1 - 2\omega u_0 + \omega^2 y^2)^{-b} y dy \\
 &= [2\omega^2(b-1)v_1|z_1||1 - \omega z_1|^{2(b-1)}]^{-1}, \quad b > 1.
 \end{aligned}$$

The other integral in (A.2) is similarly handled, except that  $\omega = -1$ . Hence the squared error is bounded by

$$\mathcal{E}^2 \leq \frac{|\exp(-\lambda z_1/(1+z_1))|^2}{4\pi^2 \omega^2 (a-1)(b-1)v_1^2 |z_1|^2 |1+z_1|^{2(a-1)} |1-\omega z_1|^{2(b-1)}}, \quad a > 1, \quad b > 1$$

and

$$\begin{aligned}
 (A.7) \quad \mathcal{E} &\leq \frac{|\exp(-\lambda z_1/(1+z_1))|}{2\pi \omega [(a-1)(b-1)]^{1/2} v_1 |z_1| |1+z_1|^{a-1} |1-\omega z_1|^{b-1}} \\
 &= \frac{|1+z_1||1-\omega z_1|}{2\omega v_1 [(a-1)(b-1)]^{1/2}} |I|,
 \end{aligned}$$

where

$$(A.8) \quad |I| = \pi^{-1} \left| z_1^{-1} (1+z_1)^{-a} (1-\omega z_1)^{-b} \exp\left(-\frac{\lambda z_1}{1+z_1}\right) \right|$$

is the absolute value of the integrand at the cutoff point. In this analysis it is unnecessary for  $u_0$  to be the saddlepoint of the integrand, provided that for  $\lambda > 0$ ,  $u_0 > -1$ ; for  $\lambda = 0$  there is no such condition.

#### REFERENCES

- [1] S. KOŁODZIEJCZYK, *On an important class of statistical hypotheses*, *Biometrika*, 27 (1935), pp. 161-190.
- [2] P. C. TANG, *The power function of the analysis of variance tests with tables and illustrations of their use*, *Statistical Research Memoirs*, 2 (1938), pp. 126-149+8 tables.
- [3] A. WALD, *On the power function of the analysis of variance test*, *Ann. Math. Statist.*, 13 (1942), pp. 434-439.
- [4] P. B. PATNAIK, *The non-central  $\chi^2$  and F-distributions and their applications*, *Biometrika*, 36 (1949), pp. 202-232.
- [5] R. L. MITCHELL AND J. F. WALKER, *Recursive methods for computing detection probabilities*, *Trans. IEEE, AES-7* (1971), pp. 671-676.
- [6] A. E. GIBSON, *Adaptive detection probabilities for fluctuating target models in nonhomogeneous Gaussian noise*, *Trans. IEEE, AES-9* (1973), pp. 113-115.
- [7] G. M. DILLARD, *Mean level detection of nonfluctuating signals*, *Trans. IEEE, AES-10* (1974), pp. 795-799.
- [8] G. H. ROBERTSON, *Computation of the noncentral F distribution (CFAR) detection*, *Trans. IEEE, AES-12* (1976), pp. 568-571.
- [9] R. NITZBERG, *Analysis of the arithmetic mean CFAR normalizer for fluctuating targets*, *Trans. IEEE, AES-14* (1978), pp. 44-47.
- [10] M. MERRINGTON AND C. M. THOMPSON, *Tables of percentage points of the inverted beta (F) distribution*, *Biometrika*, 33 (1943), pp. 73-88.
- [11] E. S. PEARSON AND H. C. HARTLEY, *Biometrika Tables for Statisticians*, vol. 1, Cambridge Univ. Press, Cambridge, 1962. See Table 16, pp. 142-155, and Table 18, pp. 157-163.
- [12] K. PEARSON, *Tables of the Incomplete Beta Function*, 2nd ed., Cambridge Univ. Press, Cambridge, 1968.
- [13] J. WISHART, *A note on the distribution of the correlation ratio*, *Biometrika*, 24 (1932), pp. 441-456.
- [14] W. L. NICHOLSON, *A computing formula for the power of the analysis of variance test*, *Ann. Math. Statist.*, 25 (1954), pp. 607-610.



- [15] J. L. HODGES, JR., *On the noncentral beta-distribution*, Ann. Math. Statist., 26 (1955), pp. 648–653.
- [16] E. LEHMER, *Inverse tables of probabilities of errors of the second kind*, Ann. Math. Statist., 15 (1944), pp. 388–398.
- [17] R. LUGANNANI AND S. O. RICE, *Distribution of the ratio of quadratic forms in normal variates—numerical methods*, this Journal, 5 (1984), pp. 476–488.
- [18] S. O. RICE, *Distribution of quadratic forms in normal random variables—evaluation by numerical integration*, this Journal, 1 (1981), pp. 438–448.
- [19] C. W. HELSTROM, *Comment: Distribution of quadratic forms in normal random variables—evaluation by numerical integration*, this Journal, 3 (1983), pp. 353–356.
- [20] S. O. RICE, *Efficient evaluation of integrals of analytic functions by the trapezoidal rule*, Bell System Tech. J., 52 (1973), pp. 707–722.
- [21] F. W. J. OLVER, *Asymptotics and Special Functions*, Academic Press, New York, 1974. See pp. 80–82.
- [22] C. W. HELSTROM, *Approximate evaluation of detection probabilities in radar and optical communications*, Trans. IEEE, AES-14 (1978), pp. 630–640.

## A TWO-DIMENSIONAL MESH REFINEMENT METHOD FOR PROBLEMS WITH ONE-DIMENSIONAL SINGULARITIES\*

DAVID L. BROWN† AND LUIS GUILLERMO M. REYNA‡

**Abstract.** This paper introduces a method for resolving internal layers that can occur in the solutions of time-dependent differential equations in two space dimensions. Singular features in these solutions that are essentially one-dimensional in nature but are not oriented with the computational mesh are resolved using one-dimensional mesh refinement techniques with a procedure that is similar to an ADI method. A careful interpolation procedure assures that the resolution obtained in each ADI step is not lost in the succeeding ADI step.

**Key words.** two-dimensional mesh refinement, internal layers, shock resolution, alternating direction implicit method

**1. Introduction.** In this paper we discuss a numerical method developed to resolve internal layers which can occur in the solutions of time-dependent problems. As an example, the two-dimensional Burgers' equation exhibits behavior of the type we wish to discuss: We consider

$$(1.1) \quad U_t + \left(\frac{1}{2}U^2\right)_x + \left(\frac{1}{2}U^2\right)_y = \varepsilon(U_{xx} + U_{yy})$$

on  $-\infty \leq x, y \leq \infty$ ,  $t \geq 0$  with initial values  $U(x, y, 0)$  given. Here  $U = U(x, y, t)$  is the dependent variable,  $0 < \varepsilon \ll 1$  is a small parameter and subscripts denote partial differentiation. Depending on the initial conditions given, the solutions of (1.1) can exhibit either boundary layer or internal layer behavior, i.e., regions of rapid change in the solution can occur in locally one-dimensional regions of width  $O(\varepsilon)$ , while the solution everywhere else varies on a length scale which is  $O(1)$ . The internal layers which occur in the solutions of (1.1) are typically referred to as *viscous shock profiles* with  $\varepsilon$  being the viscosity coefficient.

In some problems, such as compressible fluid flow, it is often the case that the accurate resolution of the viscous shock profiles is unimportant; only the size of the jump at the shock (and hence its speed) has any significant effect on the smooth part of the solution. Methods that give an accurate representation of the solutions to these types of problems without resolving the details of the shock structure can be found in, for example, Engquist and Osher [4], Osher and Solomon [12], Harten and Lax [6], and Chorin [3]. The purpose of this paper, however, is to present a method which can be applied to problems in which the detailed structure of the internal transition layers is important. Such problems arise, for example, in the study of chemically reacting fluids, where the details of the transitions can influence the speed of propagation of the reaction fronts. Methods for one-dimensional problems of this type have been considered by Brown [1], and Miller, Doss and Miller [11]. Brown uses adaptively determined local nonuniform moving mesh segments to resolve the moving features with rapid variation. This moving mesh is embedded within a much coarser mesh with meshwidths that are appropriate for accurately representing the smooth parts of the

---

\* Received by the editors July 14, 1983, and in revised form January 23, 1984. This research was partially supported by the Office of Naval Research under contract N00014-80-CO076. Computer time was provided by the Stanford Exploration Project (Stanford University Department of Geophysics), the National Center for Atmospheric Research, and on the California Institute of Technology Applied Mathematics Department Fluid Dynamics VAX.

† Department of Applied Mathematics, California Institute of Technology, Pasadena, California 91125.

‡ Courant Institute of Mathematical Sciences, New York University, New York, New York 10012.

solution. It is essentially this method that will be extended in this paper to the two space-dimensional case. We will restrict our consideration to problems whose solutions exhibit rapid transitions that are essentially one-dimensional in nature, such as viscous shock profiles. For brevity, we will often refer to such a feature as a "shock", but with the understanding that for the reasons discussed above, we are always interested in its viscous profile.

The method for two-dimensional mesh refinement that we will discuss in the following sections was originally suggested to us by H. O. Kreiss [10]. It uses the one-dimensional finite difference approximation and mesh refinement procedure discussed by Brown [1] within a "splitting" or ADI procedure to solve problems in two space dimensions. The feature of this algorithm that makes it unique among mesh refinement procedures is that one-dimensional mesh refinement techniques are applied directly to two-dimensional problems. A careful interpolation procedure is used to transfer information from the grid lines in one direction to those in the other direction in the calculation without degrading the resolution of the solution obtained on each set of one-dimensional meshes. This method is explained and discussed in more detail in §§ 2 through 4. Numerical examples illustrating the method are included in § 5.

**2. The mesh refinement procedure.** For a problem in two space dimensions in which the shock line is very nearly linear and oriented so as to be parallel to one set of coordinate lines, it is clear how to implement a mesh refinement in order to resolve its viscous profile. If, for example, the shock lies essentially parallel to a line  $x = x_0$ , a refinement in the direction normal to the shock (the  $x$ -direction) could be made (see Fig. 1). No refinement would probably be necessary in the  $y$ -direction in this case. It

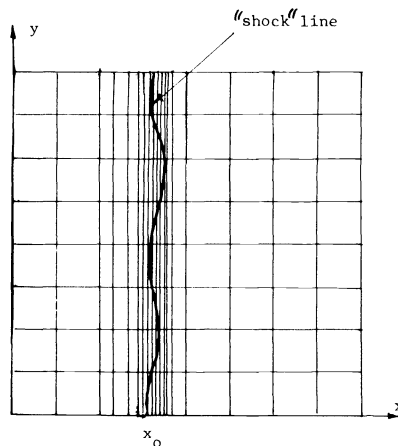


FIG. 1

is clear, however, that this will not always be true. We will not always have the freedom to choose the orientation of the computational mesh in such a way as to have "one-dimensional" rapid transitions oriented with the mesh. If the singular domain associated with the shock was not oriented with the mesh, then adding lines to refine the mesh would result in lines being added in both the  $x$  and  $y$  directions. In particular, we would also be refining the mesh in regions where the solution is smooth (see Fig. 2). This large number of added points in the mesh can clearly be reduced if we truncate the added lines so that they do not extend into smooth parts of the solution (see Fig. 3). The reduction in the number of meshpoints will, however, be at the expense of programming complexity.

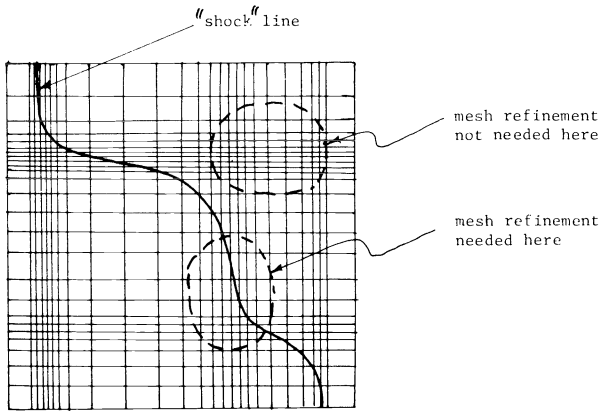
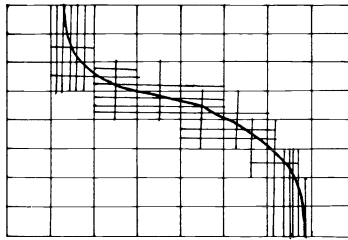
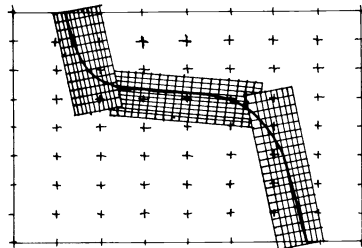


FIG. 2

FIG. 3. *Truncated mesh lines.*

A more rational approach to local mesh refinement is that of *composite meshes*. In this technique, local oriented meshes are embedded in the coarse mesh and interpolation is used to couple the solutions on the different meshes when solving the differential equations. Berger and Oliger [2], and Gropp [5] have used local oriented rectangular moving grids to accomplish this. Figure 4 illustrates the basic idea. With a simple

FIG. 4. *Method of Berger, Oliger and Gropp.*

extension of the grid generation approach of B. Kreiss [8], curvilinear grids could also be embedded in the coarse rectangular grid in such a way as to resolve a shock (see Fig. 5). Again interpolation would be used to connect the solutions on the two grids together.

Let us now consider the numerical solution of Burgers' equation in two space dimensions (1.1). We approximate the time derivative in (1.1) using the "implicit

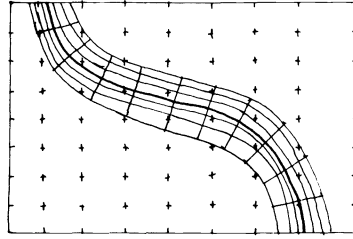


FIG. 5

Euler” approximation, giving

$$\begin{aligned}
 (2.1) \quad & \epsilon u_{xx}(x, y, t) + \epsilon u_{yy}(x, y, t) + f(u(x, y, t))_x + f(u(x, y, t))_y - \frac{1}{k} u(x, y, t) \\
 & = -\frac{1}{k} u(x, y, t - k)
 \end{aligned}$$

where  $u(\cdot, \cdot, \cdot)$  is an approximation to  $U(\cdot, \cdot, \cdot)$ ,  $k$  is the time step and for convenience we have defined  $f(u) := -\frac{1}{2}u^2$ . For numerical purposes it will also be necessary to restrict the domain of integration to be finite; we choose  $-1 \leq x, y \leq 1, 0 \leq t \leq T$  for some finite  $T$ . In addition to the initial conditions  $u(x, y, 0) = U(x, y, 0)$  we therefore must also specify boundary conditions:  $u(x, \pm 1, t)$  given for  $-1 \leq x \leq 1$  and  $u(\pm 1, y, t)$  given for  $-1 \leq y \leq 1$ .

A convenient way to implement an implicit difference scheme such as (2.1) is through operator splitting. This reduces the computational problem to a sequence of one-dimensional problems: We introduce an “underlying” coarse mesh  $\{x_i, y_j\}_{0,0}^{N,M}$  and solve by difference approximation the equations

$$(2.2a) \quad \epsilon \tilde{u}_{xx}(x, y_j) + f(\tilde{u}(x, y_j))_x - \frac{1}{k} \tilde{u}(x, y_j) = -\frac{1}{k} u(x, y_j, t - k) \quad \text{for } j = 0, 1, \dots, M$$

and

$$(2.2b) \quad \epsilon u_{yy}(x_i, y, t) + f(u(x_i, y, t))_y - \frac{1}{k} u(x_i, y, t) = -\frac{1}{k} \tilde{u}(x_i, y) \quad \text{for } i = 0, 1, \dots, N$$

with initial conditions  $u(x, y, 0) = u_0(x, y)$ . (This in general introduces an additional error into the solution which is  $O(k)$ .)

Note that with the obvious notation, each of equations (2.2) is of the form

$$(2.3) \quad \epsilon \frac{d^2 w}{dz^2} + \frac{d}{dz} [g(w)] + bw = r(z), \quad w = w(z),$$

on  $-1 \leq z \leq 1$  with  $w(-1), w(1)$  given. This is a singularly perturbed two-point boundary value problem. An extensive analytic theory exists for equations of the form (2.1) and (2.3) (see for example Kevorkian and Cole [7]). Numerical techniques for accurately resolving the features of solutions of problems of the type (2.3) have been developed by B. Kreiss and H. O. Kreiss [9]. We will use a modification of this method due to Brown [1, Chap. 3]. Briefly, each of the nonlinear problems (2.3) is solved by a functional Newton iteration. The linear problems which arise in the iteration are solved using a weighted one-sided difference approximation together with solution-adaptive mesh refinement. The details can be found in Brown [1].

Let us suppose that in the initial conditions  $u_0(x, y)$  there is a region of rapid transition oriented obliquely to the mesh. We should begin by solving (2.2a) on each of the lines  $y = y_j, j = 1, 2, \dots, M - 1$ . Because of the rapid transition region, automatic refinement will occur so that the solution of each of those one-dimensional problems would be resolved. We then solve equation (2.2b) on each of the lines  $x = x_i, i = 1, 2, \dots, N - 1$ . Again automatic mesh refinement will occur in the region near the rapid transition. Note, however, that the right-hand side of the equation for  $u(x_i, y, t)$  depends on values of the computed solution  $\tilde{u}(x_i, y)$  at the previous step. If points are added to the one-dimensional mesh between the coarse mesh lines  $y = y_j$ , this means that we will need values of  $\tilde{u}$  at points where they have not been computed (see Fig. 6). Simple linear interpolation parallel to the coordinate lines will not work well in general because the function being interpolated is not sufficiently smooth. This can be made clear by the following example (refer to Fig. 7): Consider the function  $f(x, y)$  defined by

$$f(x, y) = \begin{cases} f_1 & \text{if } x < 2(y - y_1) + x_2, \\ f_2 & \text{otherwise,} \end{cases}$$

where  $f_1 \neq f_2$ , for  $y_1 < y < y_2$ . We are interested in determining values  $f(\tilde{x}, y)$  for  $y_1 < y < y_2$  when we only know the functions  $f(x, y_1)$  and  $f(x, y_2)$ , where  $x_1 < \tilde{x} < x_2$ . (This corresponds to the situation in Fig. 6 as well.) Linear interpolation along the line  $x = \tilde{x}$  will clearly give an inappropriate answer if  $|f_1 - f_2|$  is large. Using simple

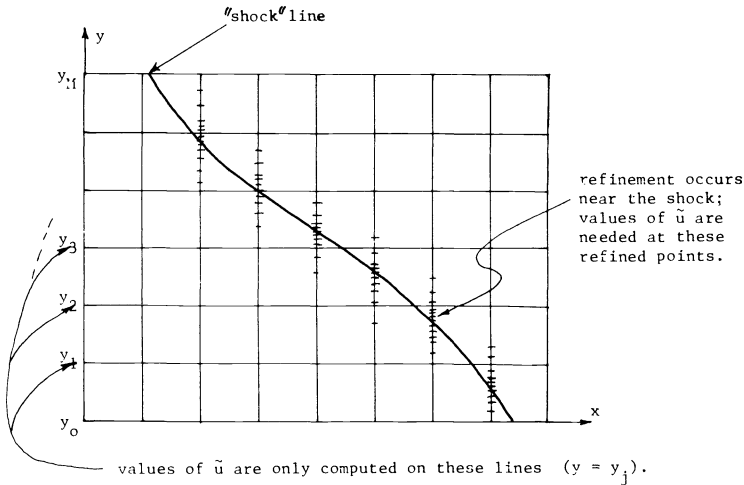


FIG. 6

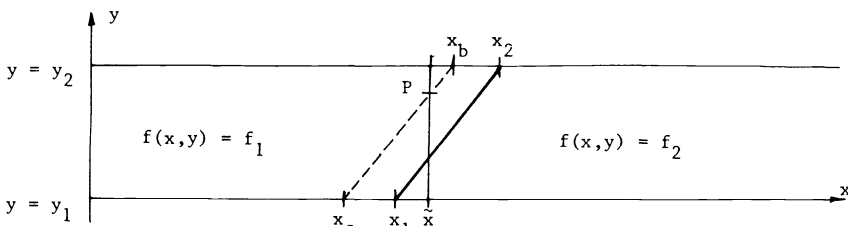


FIG. 7

linear interpolation along  $x = \tilde{x}$  we will always get a value between  $f_1$  and  $f_2$ , while the correct value should be *either*  $f_1$  or  $f_2$ . To remedy this problem, we can eliminate the restriction that interpolation always be made along lines of constant  $x$ . We can get a reasonable value for  $f$  at the point  $P$  if, for example, we interpolate linearly along the straight line between  $(x_a, y_1)$  and  $(x_b, y_2)$ . The interpolation procedure we have implemented is not much different from this simple explanation. However the conditions under which it will give accurate results and the technique for its practical implementation must be considered carefully. This is done in the next section.

**3. The interpolation procedure.** In this section we consider the problem of how to accurately reconstruct a function defined on only a finite number of appropriately chosen parallel lines, or cross-sections. The function is assumed to consist of large smooth regions separated by one or more nearly one-dimensional regions of singular behavior such as a shock profile. On each cross-section the function has been fully resolved by a sufficiently refined mesh. The method we describe is a completely local procedure. It is second order accurate in terms of the characteristic scale of the smooth parts of the solution (i.e.,  $O(h^2)$ ). The shape of any front is approximated by piecewise straight lines and the smooth parts of the solution by piecewise linear functions. Thus in order to interpolate  $u_0$  at the point  $(x_0, y_0)$  we only use the values  $u_0(x, y_{m+1})$  and  $u_0(x, y_m)$ , where  $y_m \leq y_0 \leq y_{m+1}$ . It is clear that the localness of the procedure is a particularly useful feature when the singular domain is topologically complex, for example when there are two or more possibly intersecting shocks.

Consider now the function  $u_0(x, y)$  defined on the strip  $(-\infty, \infty) \times [0, 1]$ . Then we wish to determine  $I(u_0)(x_0, y_0)$ , the interpolant of the function  $u_0(x, y)$  at  $(x_0, y_0)$ , from the function values of  $u_0$  for any  $x$  and for values of  $y$  that belong to the sequence  $0 = y_1 < \dots < y_N = 1$ .

Our aim is to interpolate using function values obtained from points which can be joined by a smooth curve lying entirely on the same smooth part of the solution. One should note that the points cannot lie at too great a distance from each other. From a practical point of view this means that the shock lines cannot be oriented too nearly parallel to the lines  $y = y_j$ . This is not a serious restriction, however, because it corresponds to the situation illustrated previously in Fig. 1 and can be taken care of using conventional mesh refinement. In the context of our mesh refinement procedure we would handle this case by simply adding one or more lines to the "underlying coarse mesh"  $y = y_j$ .

The method depends on the following assumptions about  $u_0(x, y)$ :

(i) The function should be smooth at distances greater than  $\varepsilon$  away from a nearly one-dimensional region where there is singular behavior. Here,  $\varepsilon$  is a positive number much smaller than the natural scale corresponding to changes of  $u_0(x, y)$  outside of this region. (Note that this  $\varepsilon$  is not necessarily the same one as appears in the differential equation (1.1), although if the function being interpolated is a solution of (1.1) then the two are certainly related. In this section we use  $\varepsilon$  to denote the scale of the region of singular behavior.) If the function  $u_0$  is only known at discrete values, we then assume that away from this singular region a mesh of size  $h$  completely resolves the function, where  $h \gg \varepsilon$ .

(ii) We assume that the singular region is the union of a finite number of smooth curves. In this way the curves can be isolated from each other, and if they intersect, the number of possible intersections is finite.

(iii) We assume that the singular behavior is of the shock type, i.e., a rapid but essentially monotone transition (and not a high frequency oscillation) matching two

different smooth states. We will use this assumption in the method in order to define the local orientation of the shock.

Referring to Fig. 8, the procedure for obtaining an interpolated value of  $u_0$  at  $(x_0, y_0)$  is as follows: Denote  $P_m = (x_0, y_m)$  and  $P_{m+1} = (x_0, y_{m+1})$  and introduce the jump in function values from top ( $y = y_{m+1}$ ) to bottom ( $y = y_m$ ) lines and the horizontal curvatures:

$$(3.1a) \quad \delta u_0(\tilde{x}, m) = |u_0(\tilde{x}, y_{m+1}) - u_0(\tilde{x}, y_m)|,$$

$$(3.1b) \quad \kappa(\tilde{x}, m) = \left| \frac{\partial^2 u_0}{\partial x^2}(\tilde{x}, y_{m+1}) - \frac{\partial^2 u_0}{\partial x^2}(\tilde{x}, y_m) \right|.$$

We also introduce the test function

$$(3.1c) \quad T(\tilde{x}, m) = \max(\delta u_0(\tilde{x}, m), \kappa(\tilde{x}, m)).$$

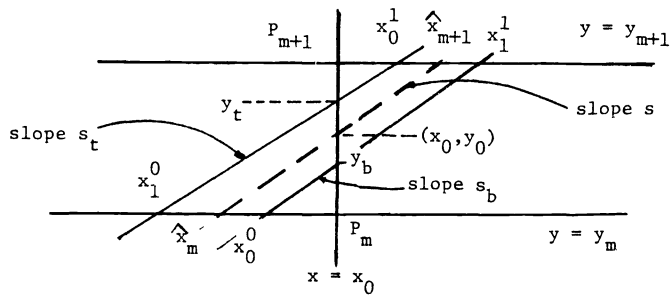


FIG. 8

We have two general cases:

*Case I.*  $T(x_0, m) \leq \beta h$ , where  $\beta$  is some positive constant defined by the requirement that away from the singular region, the magnitude of the gradients of  $u_0(x, y)$  are strictly bounded by  $\beta$ . In this case we assume that there is no shock structure nearby and we perform linear interpolation:

$$(3.2) \quad I(u_0)(x_0, y_0) = \frac{y_0 - y_m}{y_{m+1} - y_m} u_0(x_0, y_{m+1}) + \frac{y_{m+1} - y_0}{y_{m+1} - y_m} u_0(x_0, y_m).$$

*Remarks.*

1) Usually there is no need to interpolate in this case: there would be no need to know interpolated values of the right-hand sides of (2.2a) or (2.2b) if there is no shock structure nearby.

2) In a higher order approximation we should look at the  $T(\tilde{x}, \tilde{m})$ , for  $\tilde{x} = x_0 - h, x_0$  and  $x_0 + h$ , as well as for  $\tilde{m} = m - 1, m$  and  $m + 1$ , to determine the existence of a shock in the vicinity of the points where we are interpolating.

3) Note that, by definition,  $T(\cdot, \cdot)$  measures both the size of jumps in the solution and the curvature of the solution near the jumps. Practical experience has indicated that it is important to monitor not only the function values but also the curvature so that the top and bottom of the shock profile are not deformed by the interpolation procedure.

*Case II.*  $T(x_0, m) > \beta h$  (i.e., there is a singular structure in the vicinity). We have two different cases according to size of the jump  $\delta u_0(x_0, m)$ .

*Case II.a.* (See Fig. 8) If  $\delta u_0(x_0, m) > \beta h$ , then the shock line crosses the segment  $[P_m, P_{m+1}]$ . The aim now is to isolate the region of singular behavior. Practical experience



has shown that it is best to define this region with two curves. In Fig. 8 these are the lines defined by  $[(x_1^0, y_m), (x_0^1, y_{m+1})]$  and  $[(x_0^0, y_m), (x_1^1, y_{m+1})]$ . For convenience we assume that

$$(3.3) \quad \delta_1 = u_0(x_0, y_{m+1}) - u_0(x_0, y_m) > 0.$$

We now determine  $x_\gamma^1$ , for  $\gamma = 0, 1$ , by

$$(3.4a) \quad u_0(x_\gamma^1, y_{m+1}) = u_0(x_0, y_{m+1}) - l_\gamma \delta_1,$$

where we arbitrarily choose  $l_0 = \frac{1}{3}$  and  $l_1 = \frac{2}{3}$ . In a similar way we determine  $x_\gamma^0$  by

$$(3.4b) \quad u_0(x_\gamma^0, y_m) = y_0(x_0, y_m) + l_\gamma \delta_1.$$

(The slope of the function  $u_0$  in the shock region is of  $O(\epsilon^{-1})$  and so a change in the constants  $l_\gamma$  will only produce an  $O(\epsilon)$  change in the determination of the points  $x_\gamma^\sigma$  which themselves determine the direction of the shock.)

We now make sure that the points obtained are close enough to each other, that is, we compute

$$(3.5a) \quad (d_t)^2 = (y_{m+1} - y_m)^2 + (x_1^0 - x_0^1)^2,$$

$$(3.5b) \quad (d_b)^2 = (y_{m+1} - y_m)^2 + (x_0^2 - x_1^1)^2.$$

If either  $d_t$  or  $d_b$  is greater than  $2h$ , or if it is not possible to determine any of the points  $x_\gamma^\sigma$ , for  $\gamma = 0, 1$  and  $\sigma = 0, 1$ , then there is not enough information to perform an accurate interpolation. In this case, we need to obtain the values of  $u_0(x, \tilde{y})$ , where  $\tilde{y} = \frac{1}{2}(y_m + y_{m+1})$  and  $-\infty < x < \infty$ . In the shock problem this is done by going back to the last half time step computation. (See Figs. 9 and 10 for some possible situations in which this procedure asks for more information.)

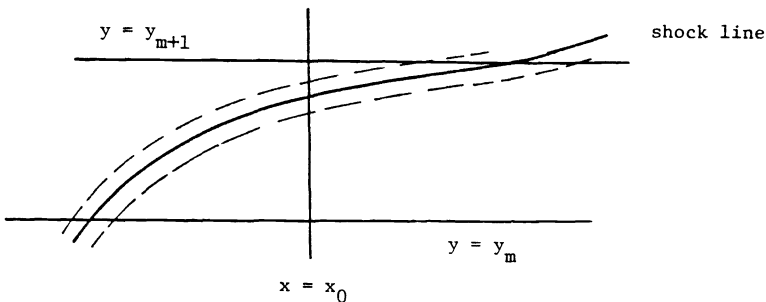


FIG. 9

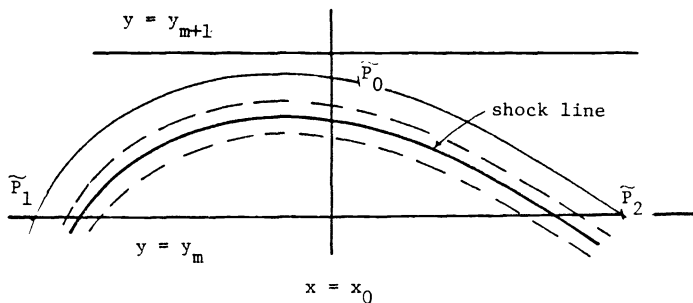


FIG. 10

We must still consider the possibility illustrated in Fig. 11, that is, when two or more singular regions come together at one point. In order for the method to recognize this situation we must also make sure that

$$|u_0(x_1^1 + h, y_{m+1}) - u_0(x_0, y_m)| \leq \beta h,$$

and similarly that

$$|u_0(x_1^0 - h, y_m) - u_0(x_0, y_{m+1})| \leq \beta h.$$

(The example in Fig. 11 would fail the second test because of the presence of shock #2.) If either of these tests fail, we must again ask for more information.

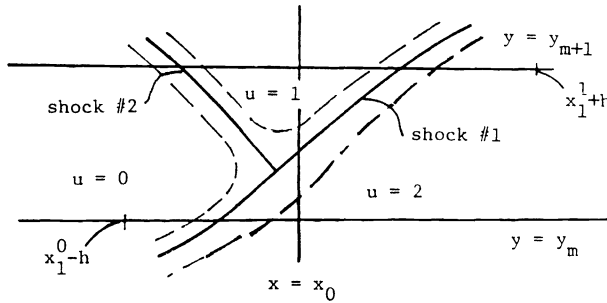


FIG. 11

Having ruled out the anomalous cases we now determine the slope of the segments joining the points that lie on the same part of the smooth solution

$$(3.6a) \quad s_t = \frac{y_{m+1} - y_m}{x_1^0 - x_0^1},$$

$$(3.6b) \quad s_b = \frac{y_{m+1} - y_m}{x_1^1 - x_0^0},$$

and the intersections of the segments with the vertical line through  $P_m$  and  $P_{m+1}$

$$(3.7a) \quad y_t = s_t(x - x_0^1) + y_m,$$

$$(3.7b) \quad y_b = s_b(x - x_0^0) + y_m.$$

Defining  $s$  in the following way

$$(3.8) \quad s = \begin{cases} s_t & \text{for } y_{m+1} \geq y_0 \geq y_t, \\ \frac{y_0 - y_b}{y_t - y_b} s_t + \frac{y_t - y_0}{y_t - y_b} s_b & \text{for } y_t \geq y_0 \geq y_b, \\ s_b & \text{for } y_b \geq y_0 \geq y_m, \end{cases}$$

we compute the intersection of a straight line through the point  $(x_0, y_0)$  with slope  $s$  with the top and bottom lines  $y = y_m$  and  $y = y_{m+1}$

$$(3.9a) \quad \hat{x}_{m+1} = x_0 + \frac{y_{m+1} - y_0}{s},$$

$$(3.9b) \quad \hat{x}_m = x_0 + \frac{y_0 - y_m}{s}.$$

Finally the required interpolated value  $I(u_0)(x_0, y_0)$  is determined by linear interpolation using the function values of  $u_0$  at the points  $(\hat{x}_m, y_m)$  and  $(\hat{x}_{m+1}, y_{m+1})$ :

$$(3.10) \quad I(u_0)(x_0, y_0) = \frac{y_0 - y_m}{y_{m+1} - y_m} u_0(\hat{x}_{m+1}, y_{m+1}) + \frac{y_{m+1} - y_0}{y_{m+1} - y_m} u_0(\hat{x}_m, y_m).$$

Case II.b. If  $\delta u_0(x_0, m) \leq \beta h$ , we then have a shock close to either point (or close to both points). Introduce

$$\tilde{\delta}(\tilde{x}, m, n) = \max (|u_0(\tilde{x}, y_m) - u_0(\tilde{x} + h, y_n)|, |u_0(\tilde{x}, y_m) - u_0(\tilde{x} - h, y_n)|);$$

and define  $\delta_2 = \max (\tilde{\delta}(x_0, m, m + 1), \tilde{\delta}(x_0, m + 1, m))$ . If  $\delta_2 \leq \beta h$  then we need more information (see the discussion following equations (3.5)). Otherwise we proceed as in Case II.a looking for either a vertical or an oblique or curved shock, as in Fig. 4. If it is not possible to find any such structure we again need extra information.

Notice that the interpolation procedure does not produce a continuous function of  $(x_0, y_0)$ ; this is because a different interpolation method is used in regions near the shock than in regions away from the shock. Recall that near the shock, linear interpolation along lines essentially parallel to the shock is used while away from the shock, linear interpolation parallel to the underlying coarse grid lines is used. The interpolation method chosen changes discontinuously as a function of the distance from the region of singularity. This is of no practical importance, however, because the interpolant is needed and computed at only a discrete set of  $x$  values.

In the cases corresponding to Figs. 9 and 10 the interpolation method will fail and will ask for extra lines  $y = const.$  to be added until  $\delta y$ , the distance between two consecutive horizontal lines, is  $O(\epsilon)$ , the width of the shock. Note that this failure is not due to the fact that we are representing the front with straight line segments. If we were to use a higher order fitting method to represent the front and even if we assume that its shape is known exactly, we would still need to add these extra horizontal lines. This is explained as follows: In order to find the value of the interpolant at  $P_0 = (x_0, y_0)$  (Fig. 10), we perform some interpolation along a curve parallel to the front. The interpolation formula will link values of the functions at points such as  $\tilde{P}_1$  and  $\tilde{P}_2$ , and possibly points on other horizontal lines. The distance between these points is  $O(\delta y^{1/2})$  in the cases of Figs. 9 and 10. This implies that if we only restrict ourselves to interpolate using points that are  $O(h)$  apart from each other then we must restrict  $\delta y$  to be  $O(h^2)$ . Hence, a higher order fitting method will reduce the number of operations when compared to the second order fitting method we have described, but there will be no saving in the number of coarse mesh lines needed.

**4. The interpolation error.** In order to understand the interpolation error, we consider the problem corresponding to  $\epsilon = 0$  (that is, an actual discontinuity) and to only one shock. In this case  $x_0^0 = x_0^1 = x_b$ ,  $x_0^1 = x_1^1 = x_t$  and  $s_t = s_b = s$  where  $x_b$  corresponds to the intersection of the shock line with the line  $y = y_m$  and similarly for  $x_t$  (see Fig. 8). Without any loss of generality we can consider  $x_b < x_t$ . In this case the interpolant evaluated at  $(x_0, y_0)$  (for  $y_m \leq y_0 \leq y_{m+1}$ ) is defined by equation (3.2) when  $x_0 \geq x_t$  or  $x_0 \leq x_b$ , and by equation (3.10) when  $x_b < x_0 < x_t$ .

It is not possible to obtain an error formula in the maximum norm. The existence of such an error formula would imply that it is possible to determine the shape of an arbitrary curve on the plane from a finite number of its points. Nevertheless we have the following obvious local error estimates (see Fig. 12): the error is  $O(\delta y^2)$  away from the shock (region I),  $O(\delta y^2 \sqrt{1 + s^{-2}})$  near the shock region (region II), and  $O(1)$  between the shock and the chord  $[(x_b, y_m), (x_t, y_{m+1})]$  (region III). The area of this last

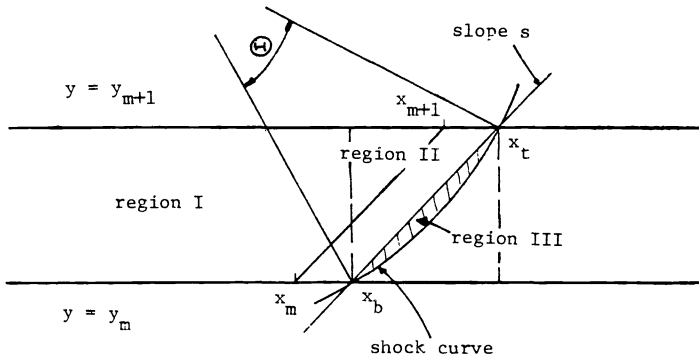


FIG. 12

region is  $A = \frac{1}{2}\kappa^{-2}(\theta - \sin(\theta))(1 + O(\delta y))$  where  $\kappa$  is a value characteristic of the curvature of the shock front when the front lies between the lines  $y = y_m$  and  $y = y_{m+1}$ , and  $\theta$  is the change in the angle of the tangents of the shock front as the shock moves from  $y = y_m$  to  $y = y_{m+1}$ . The angle  $\theta$  is determined by  $\sin(\frac{1}{2}\theta) = \frac{1}{2}\kappa \delta y \sqrt{1 + s^{-2}}$ .

In order to have an accurate interpolation, the shape of the front has to be resolved. We can assume that this has been achieved when  $A_{III}$ , the total area of regions of type III, is  $O(h^2)$ . Now, when  $\theta$  is small and  $|s| > \delta y$  we have that  $A \approx \kappa(\sqrt{1 + s^{-2}} \delta y)^3$ . Thus in order to resolve the shock we need  $A \delta y / s = O(h^2)$ . In this way when  $s = O(1)$  and when the shock is a smooth curve (i.e., when we have an upper bound for  $\kappa$ ), the condition on the area amounts to  $\delta y / s$  being  $O(h)$ . This relation between the orientation of the shock and the distance between two consecutive horizontal lines was enforced by making  $d_t$  and  $d_b$  smaller than  $2h$ . On the other hand when  $|s| < \delta y$ , we have that  $A \approx \delta y$ ; hence, in this case we need  $\delta y = O(h^2)$ . This implies that we should stop adding extra horizontal lines when  $\delta y$  is  $O(h^2)$ .

**5. Numerical examples.** In this section we present some numerical examples designed to test the interpolation procedure discussed in the last two sections. In particular we include some examples where the interpolation procedure has been used in conjunction with operator splitting to solve (1.1) for two sets of initial data.

*Example 1.* To test the performance of the interpolation procedure, we first used it to interpolate two known functions containing a region of rapid transition and compared the results of the interpolation with the original function. The functions considered were

$$(5.1) \quad u_0(x, y) = \tanh((y - \frac{1}{2}x^2 - \frac{1}{2}x) / \epsilon)$$

and

$$(5.2) \quad u_0(x, y) = \tanh((y - x) / \epsilon) - \frac{1}{4} \sin(\pi(x + y))$$

on  $0 \leq x \leq 1$  with  $\epsilon = .02$ . The function (5.1) models the case where the shock line is curved; (5.2) models the case where the smooth part of the function is not constant. We began by calculating the function  $u_0$  at specified points  $x = x_k^{(j)} \in [0, 1]$ ,  $k = 1, 2, \dots, n_j$  along uniformly spaced parallel lines  $y = y_j$ ,  $j = 0, 1, \dots, N$ ,  $N = 20$ . We also calculated the function along the additional boundary lines of  $0 \leq x, y \leq 1$  i.e.,  $x = 0, x = 1$ . The interpolation points were specified in such a way that the function was "resolved" on each cross-section.

In the first step of the interpolation, the function  $u_0(x, y)$  was interpolated onto  $N - 1$  uniformly spaced cross-sections  $x = x_l$ ,  $l = 1, 2, \dots, N - 1$  (perpendicular to the

original cross-sections). Again this was done at points along these cross-sections that were chosen so that the function would be well-resolved. We denoted the resulting function by  $u_1(x, y)$ .

This process was then repeated, interpolating the function  $u_1(x, y)$  back onto the original cross-sections  $y = y_j$ , obtaining a function  $u_2(x, y)$  defined at points  $x = x_k^{(j)}$  as above.

Let  $e(x, y) = u_2(x, y) - u_0(x, y)$ ; then we introduce the maximum norm, the  $L_1$ -norm and  $L_2$ -norm of the error by

$$(5.3) \quad \begin{aligned} \|e(x, y)\|_\infty &= \max_{x_k^{(j)}, y_j} |e(x_k^{(j)}, y_j)|, \\ \|e(x, y)\|_2^2 &= \sum_{j,k} (e(x_k^{(j)}, y_j))^2 h_{j,k} / N, \\ \|e(x, y)\|_1 &= \sum_{j,k} |e(x_k^{(j)}, y_j)| h_{j,k} / N, \end{aligned}$$

where

$$(5.4) \quad h_{j,k} = \begin{cases} x_1^{(j)} - x_0^{(j)} & \text{if } k = 0, \\ x_k^{(j)} - x_{k-1}^{(j)} & \text{if } k \neq 0, \end{cases}$$

is the "meshwidth" associated with the  $k$ th interval on the  $j$ th line  $y = y_j$ . The errors observed for the functions (5.1) and (5.2) were

$$(5.5) \quad \|e\|_\infty = .056, \quad \|e\|_2 = .0084, \quad \|e\|_1 = .0032,$$

and

$$(5.6) \quad \|e\|_\infty = .024, \quad \|e\|_2 = .0047, \quad \|e\|_1 = .0022,$$

respectively.

Figures 13 and 14 show the results of this interpolation graphically for the functions (5.1) and (5.2), respectively. Figures a show the original function, Figs. b the interpolant and Figs. c the interpolation error for each of these examples. The scale has been magnified somewhat in the error plots. We see that the function and its interpolant

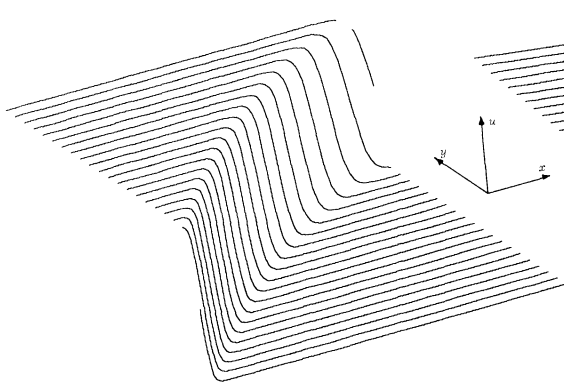


FIG 13a. Function to be interpolated.

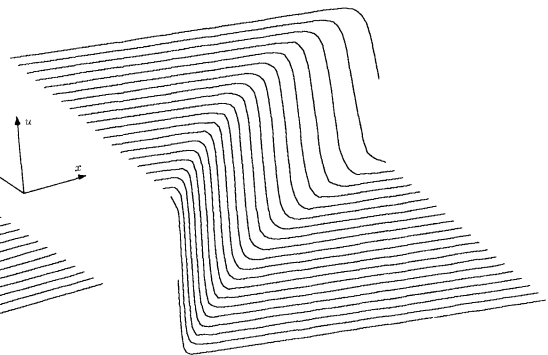
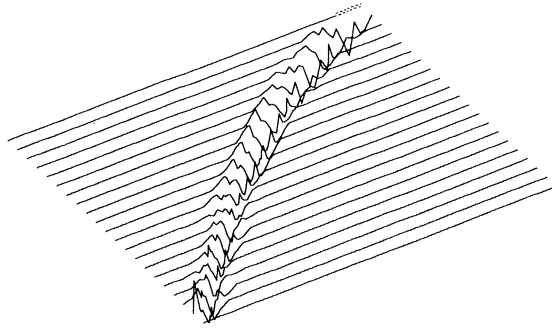
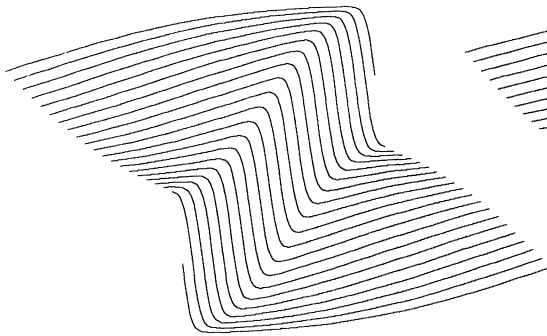
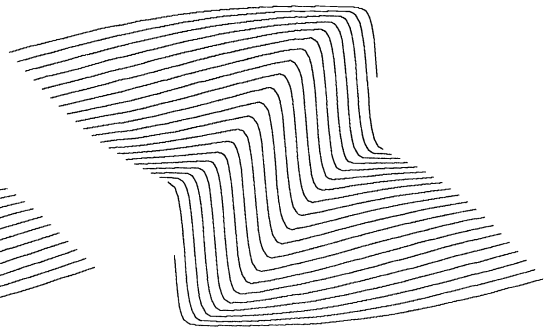
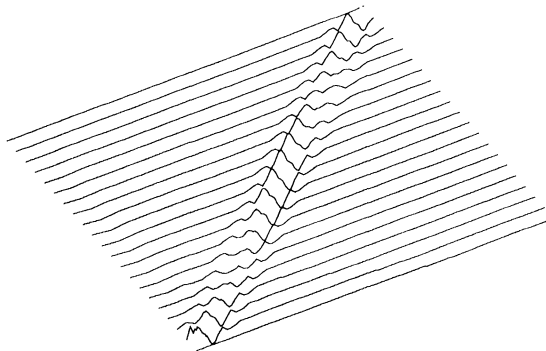


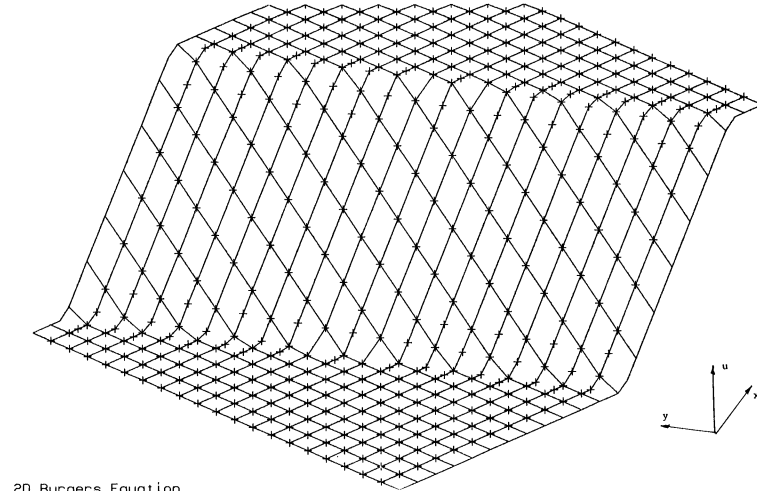
FIG. 13b. The interpolated function.

FIG. 13c. *The interpolation error.*

are difficult to distinguish; in particular, the transition region in the functions is quite well resolved even after being interpolated twice. As we would expect, the error is confined to the region of singularity in the function.

*Example 2.* We used our interpolation procedure in conjunction with operator splitting to solve the two-dimensional Burgers' equation with initial conditions that result in stationary rapid transitions oriented obliquely to the mesh.

FIG. 14a. *The function to be interpolated.*FIG. 14b. *The interpolated function.*FIG. 14c. *The interpolation error.*



2D Burgers Equation  
 $\epsilon = 0.00250$   
 $t = 0.00000$

FIG. 15a

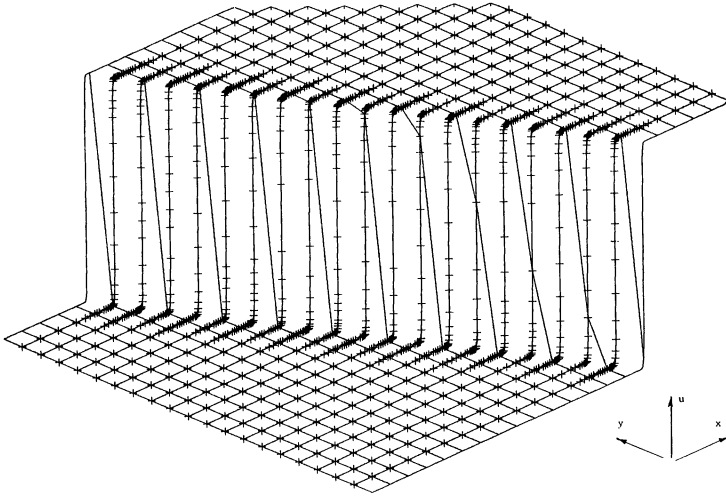
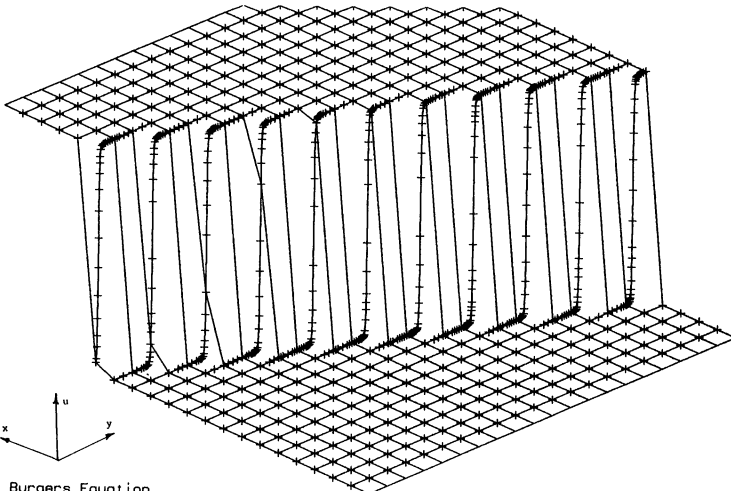


FIG. 15b



2D Burgers Equation  
 $\epsilon = 0.00250$   
 $t = 1.00000$

FIG. 15c

2D Burgers Equation  $\epsilon = 0.002500$   $t = 0.000000$

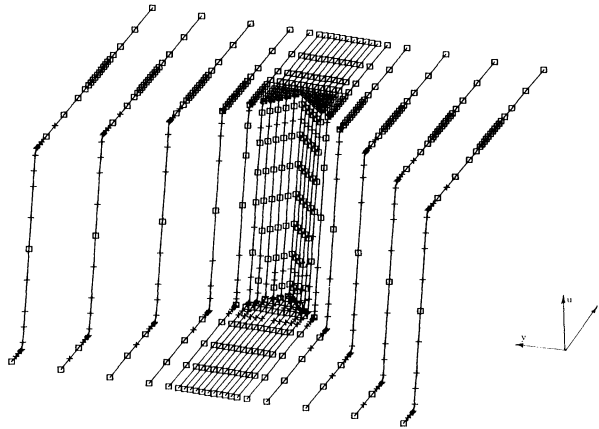


FIG. 16a

2D Burgers Equation  $\epsilon = 0.002500$   $t = 0.200000$

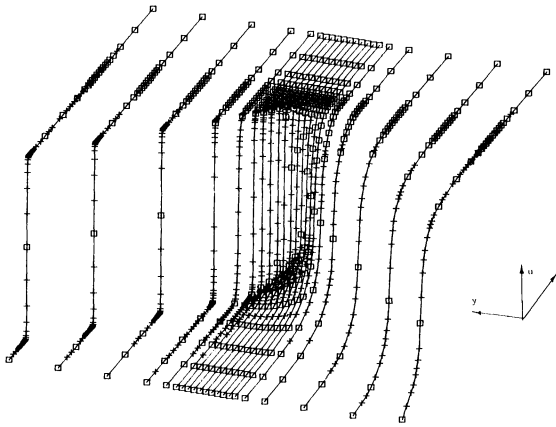


FIG. 16b

2D Burgers Equation  $\epsilon = 0.002500$   $t = 0.200000$

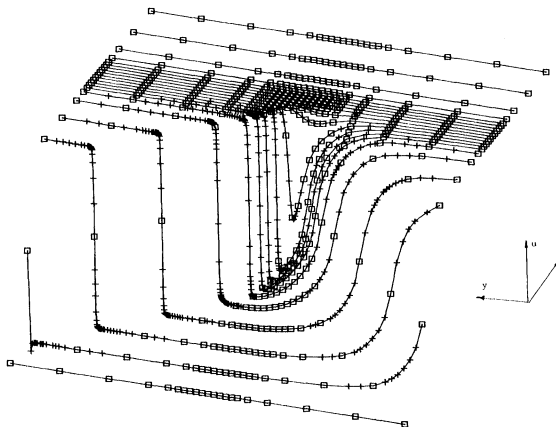


FIG. 16c



2D Burgers Equation eps = 0.002500 t = 1.000000

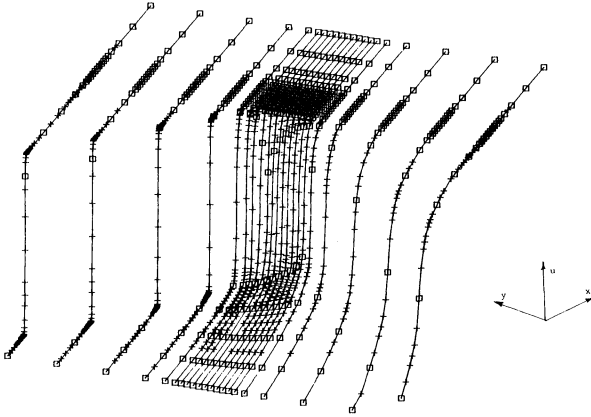


FIG. 16d

2D Burgers Equation eps = 0.002500 t = 1.000000

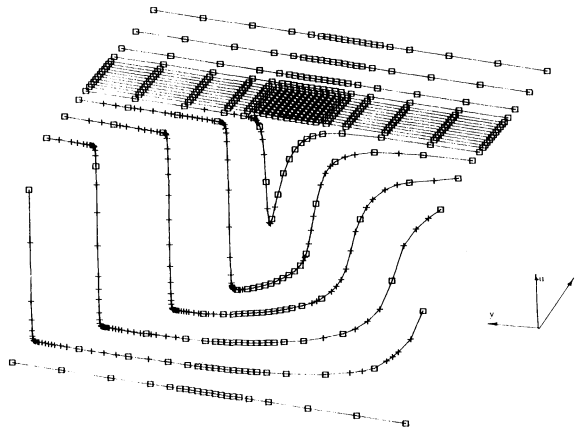


FIG. 16e

Figures 15 show the initial data and solution at time  $t = 1$  for a computation using this method. The initial data (Fig. 15a) is a ramp oriented obliquely with respect to the mesh connecting the constant values  $u = \pm 1$ . Fig. 15b shows the solution after the last sweep in  $x$  and Fig. 15c shows the solution after the last sweep in  $y$ . The plus signs “+” indicate the locations of the mesh points in the final refined mesh. Lines are also drawn in the direction perpendicular to the sweep direction (e.g., in the  $y$ -direction in Fig. 15a) to indicate the location of the underlying coarse mesh. (Note that Fig. 15c is reversed in orientation from the other two plots in this series.)

Figures 16 show the initial data and computed solutions at time  $t = 0.2$  and time  $t = 1$  for another example using this method. The coarse mesh in this case was not a uniform one, but was finer near the center of the domain where the corner of the “wedge” occurs. This was done in an attempt to resolve that corner. The initial data also consist of ramps connecting the two constant states  $u = \pm 1$ . The two ramps are oriented in such a way that the one on the left evolves into a shock while the one on the right forms a contact discontinuity. Because of the dissipative terms in Burgers’

equation, of course, the shock has finite width, and the contact discontinuity becomes wider with time. In this series of plots, the orientation is the same for all sweeps shown. The meshpoints are indicated with small squares and plus signs. The squares denote meshpoints that lie on the underlying coarse mesh. Figures 16b and 16d are the solutions after the  $x$ -sweep at  $t = 0.2$  and  $t = 1.0$ , respectively; Figs. 16c and 16e show the solutions after the corresponding  $y$ -sweeps. In all the computations presented in this section,  $\varepsilon = 1/400$ , and the time step was  $k = 1/20$ . Note that, in particular, the intended objective of this method, to resolve steady two dimensional rapid transitions, has been realized.

**Acknowledgment.** We would like to thank Michael Naughton for carefully reading the manuscript and making several helpful comments.

#### REFERENCES

- [1] D. L. BROWN, *Solution adaptive mesh procedures for the numerical solution of singular perturbation problems*, Ph.D. thesis, Dept. Applied Mathematics, California Institute of Technology, Pasadena.
- [2] M. BERGER AND J. OLIGER, *Adaptive mesh refinement for hyperbolic partial differential equation*, J. Comp. Phys. to appear.
- [3] A. J. CHORIN, *Random choice solution of hyperbolic systems*, J. Comp. Phys., 22 (1976), pp. 517-533.
- [4] B. ENGQUIST AND S. OSHER, *One-sided difference approximations for nonlinear conservation laws*, Math. Comp., 36 (1981) pp. 321-352.
- [5] W. D. GROPP, *A test of moving mesh refinement for 2-D scalar hyperbolic equations*, this Journal, 1 (1980), pp. 191-197.
- [6] 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.
- [7] J. KEVORKIAN AND J. COLE, *Perturbation Methods in Applied Mathematics*, Springer-Verlag, New York, 1981.
- [8] B. KREISS, *Construction of curvilinear grids*, this Journal, 4 (1983), pp. 270-279.
- [9] B. KREISS AND H. O. KREISS, *Numerical methods for singular perturbation problems*, SIAM J. Numer. Anal., 18 (1981), pp. 262-276.
- [10] H. O. KREISS, personal communication, 1980.
- [11] R. J. MILLER, S. K. DOSS AND K. MILLER, *The moving finite element method: applications to general partial differential equations with multiple large gradients*, J. Comp. Phys., 40 (1981), pp. 202-249.
- [12] S. OSHER AND F. SOLOMON, *Upwind difference schemes for hyperbolic systems of conservation laws*, Math. Comp., 38 (1982), pp. 339-374.

## FAST NUMERICAL SOLUTION OF NONLINEAR VOLTERRA CONVOLUTION EQUATIONS\*

E. HAIRER†, CH. LUBICH‡ AND M. SCHLICHTÉ†

**Abstract.** Numerical methods for general Volterra integral equations of the second kind need  $O(n^2)$  kernel evaluations and  $O(n^2)$  additions and multiplications. Here it is shown how the effort can be reduced for nonlinear convolution equations. Exploiting the convolution structure, most numerical methods need only  $O(n)$  kernel evaluations. With the use of Fast Fourier Transform techniques only  $O(n(\log n)^2)$  additions and multiplications are necessary. The paper closes with numerical examples and comparisons.

**Key words.** Volterra integral equation, convolution, fast Fourier transform, Runge-Kutta method

**1. Introduction.** We consider nonlinear second kind Volterra integral equations of convolution type

$$(1) \quad y(x) = f(x) + \int_{x_0}^x k(x-s)g(s, y(s)) ds, \quad x_0 \leq x \leq \bar{x}.$$

The kernel  $k$  and the functions  $f, g$  are assumed to be sufficiently smooth on  $[x_0, \bar{x}]$ , so that the solution  $y(x)$  is smooth, too. Problems of this type appear in biology, e.g. neurophysiology (an der Heiden [7]), epidemiology (Hethcote-Tudor [9]), and in the treatment of special hyperbolic differential equations (Friedlander [5]). Further applications are given in Corduneanu [3].

Usually the starting point for numerical methods is the more general equation

$$(2) \quad y(x) = f(x) + \int_{x_0}^x K(x, s, y(s)) ds, \quad x_0 \leq x \leq \bar{x}.$$

If the integration interval is discretized with  $n$  gridpoints, then algorithms for (2) need  $O(n^2)$  evaluations of  $K$ . For the special equation (1), which appears most often in applications, the number of function evaluations can be reduced to  $O(n)$   $k$ - and  $g$ -evaluations for suitably chosen methods, e.g. extended Runge-Kutta methods and linear multistep methods.

In § 2 we describe the extended classical Runge-Kutta method. If this method is implemented straightforwardly, it requires still  $O(n^2)$  additions and multiplications. It is shown in § 3, the central part of this paper, that this overhead can be reduced to  $O(n(\log n)^2)$ . In § 4 the asymptotic expansion of the global error is used for improving the accuracy of the numerical solution and estimating the global error. Some numerical results are given in § 5. Comparisons of our code VOLCON with existing codes are presented.

The ideas of this article are not restricted to Runge-Kutta methods, they are also applicable to multistep methods. Although we present the theory only for the scalar equation (1), it also pertains to systems of equations (1), to integrodifferential equations and to weakly singular integral equations of convolution type.

\* Received by the editors June 7, 1983, and in final form January 15, 1984.

† Institut für Angewandte Mathematik, Universität Heidelberg, Im Neuenheimer Feld 293, D-6900 Heidelberg 1, Germany.

‡ Institut für Mathematik und Geometrie, Universität Innsbruck, Technikerstr. 13, A-6020 Innsbruck, Austria.

**2. The extended classical Runge–Kutta method.** Extended Runge–Kutta methods have been introduced by Pouzet [11]. The classical 4th order method, applied to (2), is given by

$$\begin{aligned}
 y_n &= \tilde{F}_n(x_n), \\
 \tilde{F}_n(x) &= f(x) + \frac{h}{6} \sum_{j=0}^{n-1} \left\{ K(x, x_j, Y_1^{(j)}) + 2K\left(x, x_j + \frac{h}{2}, Y_2^{(j)}\right) \right. \\
 &\quad \left. + 2K\left(x, x_j + \frac{h}{2}, Y_3^{(j)}\right) + K(x, x_j + h, Y_4^{(j)}) \right\}, \\
 Y_1^{(j)} &= \tilde{F}_j(x_j), \\
 Y_2^{(j)} &= \tilde{F}_j\left(x_j + \frac{h}{2}\right) + \frac{h}{2} K\left(x_j + \frac{h}{2}, x_j, Y_1^{(j)}\right), \\
 Y_3^{(j)} &= \tilde{F}_j\left(x_j + \frac{h}{2}\right) + \frac{h}{2} K\left(x_j + \frac{h}{2}, x_j + \frac{h}{2}, Y_2^{(j)}\right), \\
 Y_4^{(j)} &= \tilde{F}_j(x_j + h) + hK\left(x_j + h, x_j + \frac{h}{2}, Y_3^{(j)}\right).
 \end{aligned}
 \tag{3}$$

Here  $h$  denotes the stepsize, and  $y_n$  approximates the solution at  $x_n = x_0 + nh$ . This method is convergent with a global error of  $O(h^4)$ . For a proof see Pouzet [11] or Hairer–Lubich–Nørsett [6]. The positions, where the kernel  $K(x, s, y)$  has to be evaluated, lie very regularly in the  $(x, s)$ -plane. They are indicated with crosses in Fig. 1.

Since these points lie on lines parallel to the diagonal and the  $x$ -axis, the number of function evaluations can be reduced for the convolution equation (1). Here the method (3) reads  $(\tilde{F}_0(x) = f(x))$

$$\begin{aligned}
 (4a) \quad y_n &= \tilde{F}_n(x_n), \\
 \tilde{F}_n(x) &= f(x) + \frac{h}{6} k(x - x_0)g(x_0, Y_1^{(0)}) \\
 &\quad + \frac{h}{3} \sum_{j=0}^{n-1} k\left(x - x_j - \frac{h}{2}\right) \left[ g\left(x_j + \frac{h}{2}, Y_2^{(j)}\right) + g\left(x_j + \frac{h}{2}, Y_3^{(j)}\right) \right] \\
 (4b) \quad &\quad + \frac{h}{6} \sum_{j=1}^{n-1} k(x - x_j) [g(x_j, Y_4^{(j-1)}) + g(x_j, Y_1^{(j)})] \\
 &\quad + \frac{h}{6} k(x - x_n)g(x_n, Y_4^{(n-1)}), \\
 Y_1^{(j)} &= \tilde{F}_j(x_j), \\
 Y_2^{(j)} &= \tilde{F}_j\left(x_j + \frac{h}{2}\right) + \frac{h}{2} k\left(\frac{h}{2}\right)g(x_j, Y_1^{(j)}), \\
 (4c) \quad Y_3^{(j)} &= \tilde{F}_j\left(x_j + \frac{h}{2}\right) + \frac{h}{2} k(0)g\left(x_j + \frac{h}{2}, Y_2^{(j)}\right), \\
 Y_4^{(j)} &= \tilde{F}_j(x_j + h) + hk\left(\frac{h}{2}\right)g\left(x_j + \frac{h}{2}, Y_3^{(j)}\right).
 \end{aligned}$$

It is seen that for the computation of  $y_n$  only  $2n + 1$   $k$ - and  $f$ -evaluations and  $4n$

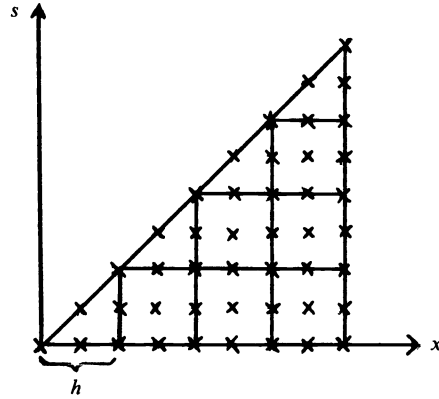


FIG. 1

$g$ -evaluations are necessary. If the formulas (4) are implemented straightforwardly,  $O(n^2)$  additions and multiplications still are needed.

**3. Fast computation of the lag-terms.** We describe in this section, how the overhead for method (4) can be reduced using FFT-techniques.

Assume that  $Y_i^{(j)}$  for  $i = 1, \dots, 4$  and  $j = 0, \dots, r - 1$  are computed directly by (4) (step I of the algorithm). With the notation

$$\begin{aligned} \kappa_j &= k((j+1)h/2), \quad j = 0, 1, \dots, 4r-1, \\ (5) \quad \gamma_0 &= \frac{h}{6} g(x_0, Y_1^{(0)}), \\ \gamma_{2j+1} &= \frac{h}{3} \left[ g\left(x_j + \frac{h}{2}, Y_2^{(j)}\right) + g\left(x_j + \frac{h}{2}, Y_3^{(j)}\right) \right], \quad j = 0, \dots, r-1, \\ \gamma_{2j} &= \frac{h}{6} [g(x_j, Y_4^{(j-1)}) + g(x_j, Y_1^{(j)})], \quad j = 1, \dots, r-1, \\ (6) \quad \gamma_{2r} &= \frac{h}{6} g(x_r, Y_4^{(r-1)}), \\ \gamma_j &= 0 \quad \text{for } j = 2r+1, \dots, 4r-1, \end{aligned}$$

the lag-term  $\tilde{F}_r(x)$  becomes

$$(7) \quad \tilde{F}_r(x) = f(x) + (\kappa * \gamma)_{2r+j} \quad \text{for } x = x_r + (j+1)h/2 \text{ and } j = 0, 1, \dots, 2r-1.$$

Here the convolution of the two  $4r$ -dimensional sequences  $\kappa = (\kappa_j)$  and  $\gamma = (\gamma_j)$  is given by

$$(\kappa * \gamma)_m = \sum_{i=0}^{4r-1} \kappa_{m-i} \gamma_i$$

This convolution can be computed efficiently using the fast Fourier transform (FFT). For a description see Henrici [8] and the references given there. These computations are illustrated in Fig. 2.

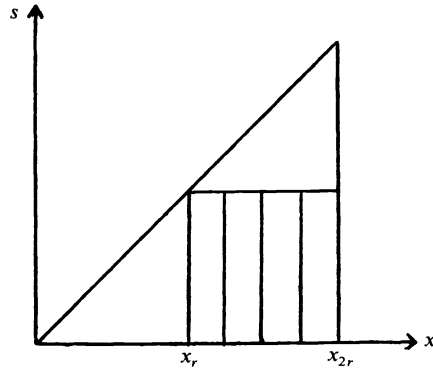


FIG. 2

Each vertical line in the square of Fig. 2 represents the lag-term  $\tilde{F}_r(x)$  as given by (4b) at the corresponding  $x$ -value. Formula (7) permits to compute the lag-terms of this square simultaneously with FFT. We have thus obtained step II in Fig. 3.

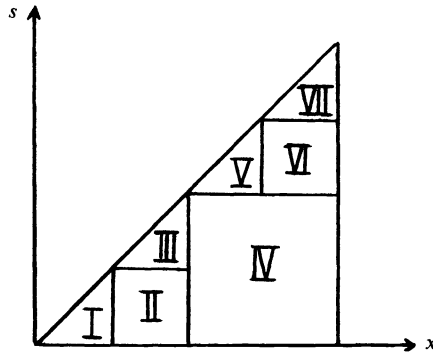


FIG. 3

For the computation of step III we observe that method (4) applied to

$$(8) \quad y(x) = \tilde{F}_n(x) + \int_{x_n}^x k(x-s)g(s, y(s)) ds, \quad x \geq x_n$$

(with  $n = r$ ) yields the same numerical solution for  $x \geq x_r$  as when applied to (1). This permits us to compute  $Y_i^{(j)}$  for  $j = r, \dots, 2r-1$  since the required  $\tilde{F}_r(x)$ -values are known by (7).

In step IV we employ the same arguments as in step II with  $r$  replaced by  $2r$  and compute

$$\tilde{F}_{2r}(x) \quad \text{for } x = x_{2r} + (j+1)h/2, j = 0, 1, \dots, 4r-1$$

simultaneously by FFT.

The steps V, VI and VII are now performed by applying the steps I, II and III to the integral equation (8) with  $n = 2r$ .

Proceeding by induction, we arrive at Fig. 4, where each square symbolizes the computation of one convolution by FFT.

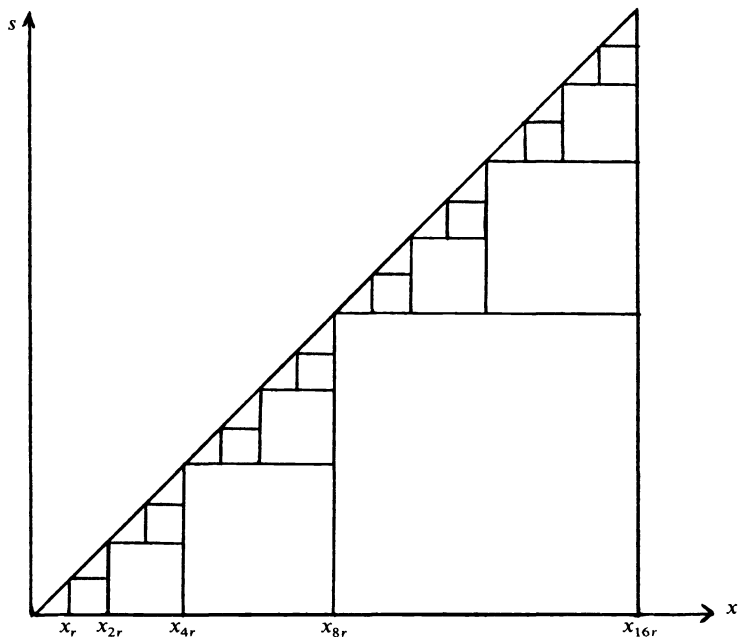


FIG. 4

Since the computation of the convolution of two  $n$ -vectors requires  $O(n \log n)$  additions and multiplications, our algorithm needs only

$$O\left(n \log n + 2\left(\frac{n}{2} \log \frac{n}{2}\right) + 4\left(\frac{n}{4} \log \frac{n}{4}\right) + \dots\right) + O(n) = O(n(\log n)^2)$$

additions and multiplications.

In Fig. 5 we compare the straightforward computation of (4) with our algorithm.

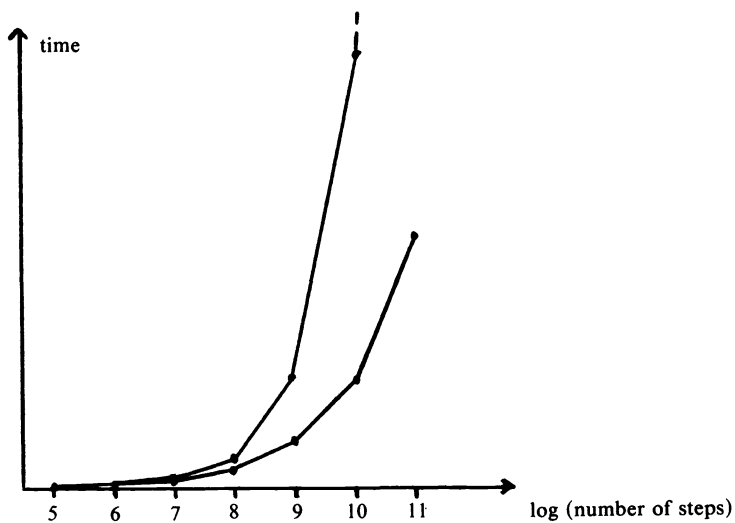


FIG. 5

Here we have chosen  $r = 2^5$ , which turned out to be optimal in numerical experiments. The difference in computer time is independent of the integral equation to be solved.

Although both algorithms are mathematically equivalent, that one, which uses the FFT, causes in general less rounding errors, since it needs fewer additions and multiplications. This is confirmed by numerical computations.

**4. Global error estimation.** Consider the method (4) and denote its numerical solution by  $y(x, h) = y_n$  if  $x = x_0 + nh$ , in order to indicate its dependence on the stepsize. It has been shown in [6] that the global error has an asymptotic expansion of the form

$$y(x) - y(x, h) = e_4(x)h^4 + e_5(x)h^5 + \dots + e_N(x)h^N + O(h^{N+1}).$$

The numerical solution at  $x$  is computed for the stepsizes  $h, h/2, h/4, h/8, \dots$  and is denoted by  $T_{i0} = y(x, h/2^i)$ . Then the extrapolation tableau (cf. [4])

$$\begin{array}{ccc} T_{00} & & \\ T_{10} & T_{11} & \\ T_{20} & T_{21} & T_{22} \\ \cdot & \cdot & \cdot \end{array}$$

is calculated according to the formula (Aitken–Neville algorithm)

$$T_{ik} = T_{i,k-1} + \frac{T_{i,k-1} - T_{i-1,k-1}}{2^{k+3} - 1}, \quad i \geq k \geq 1.$$

Since by this procedure the leading term in the asymptotic expansion of  $T_{i,k-1}$  is cancelled, we have

$$(9) \quad y(x) - T_{ik} = \gamma_{ik} e_{k+4}(x) h^{k+4} + O(h^{k+5}).$$

Observe that the leading term in (9) equals that of  $T_{i,k+1} - T_{ik}$ , which is a numerically available estimation of (9).

This motivates the following strategy: For a prescribed tolerance TOL, calculate the extrapolation tableau until we have for some indices  $i, k$

$$(10) \quad |T_{i,k+1} - T_{ik}| \leq \text{TOL}.$$

By the above considerations this difference estimates the error of  $T_{ik}$ . The more accurate value  $T_{i,k+1}$  is then accepted as a numerical approximation to  $y(x)$ .

Observe that the  $k$ - and  $f$ - values, which are needed for the computation of  $y(x, h)$ , can be used again for the computation of  $y(x, h/2)$ . Furthermore, the Fourier transforms of the kernel-values, which were computed before, can be used to reduce the effort for the computation of the kernel Fourier transforms for the stepsize  $h/2$ .

**5. Numerical experiments and comparisons.** Based on the theoretical considerations of §§ 2–4 the authors have written a FORTRAN subroutine VOLCON for the numerical solution of a scalar equation (1). This program can be obtained on request from the authors.

In this section we give some numerical results and compare them to those of the codes VE1 (due to Bowns [2]) and ORION (due to Bader–Kunkel [1]). We omit comparisons with the codes VOLTEX (due to Hock [10]) and INTSOL (due to Williams–McKee [12]), since ORION turned out to be competitive or even superior to these codes in extensive numerical tests (see [1]).



Our numerical experiments have been run on the IBM 370/168 of the University of Heidelberg in FORTRAN double precision (with about 16 decimal digits). The numerical examples to be presented are documented in terms of the quantities

TOL—prescribed absolute precision

TIME—execution time of the subroutine on the IBM 370/168

NFEV—number of  $f$ -evaluations

NKEV—number of kernel evaluations

NGEV—number of  $g$ -evaluations

ERRACT—actual absolute error

*Problem 1.* As a first test example we have taken the equation

$$y(x) = \cos x - 2 \int_0^x (x-s+2)^{-2}(y(s) + y^3(s)) ds$$

on the intervals  $[0, 10]$  and  $[0, 40]$ . The exact solution at the endpoints is  $y(10) = -0.4718905296$  and  $y(40) = -0.6501311013$ . These values have been obtained numerically using different codes with very stringent tolerances.

A linear version of this equation is formula (8b) in Friedlander [4]. We have introduced a nonlinearity, since for linear convolution equations the FFT can be used directly without applying the techniques of § 3.

In Tables 1 and 2 the numerical results of VOLCON are presented. ERREST denotes the difference of the best to the second-best approximation in the extrapolation tableau (cf. § 4 formula (10)).

TABLE 1  
*Results of VOLCON for Problem 1 (XEND = 10).*

TOL	TIME	NFEV	NKEV	NGEV	ERREST	ERRACT
$10^{-2}$	0.016	70	70	204	$0.159 \times 10^{-4}$	$0.440 \times 10^{-5}$
$10^{-4}$	0.017	74	74	220	$0.159 \times 10^{-4}$	$0.440 \times 10^{-5}$
$10^{-6}$	0.049	146	146	508	$0.138 \times 10^{-6}$	$0.493 \times 10^{-8}$
$10^{-8}$	0.144	322	322	1,208	$0.770 \times 10^{-10}$	$0.127 \times 10^{-9}$

TABLE 2  
*Results of VOLCON for Problem 1 (XEND = 40).*

TOL	TIME	NFEV	NKEV	NGEV	ERREST	ERRACT
$10^{-2}$	0.121	262	262	780	$0.902 \times 10^{-4}$	$0.212 \times 10^{-5}$
$10^{-4}$	0.122	266	266	796	$0.902 \times 10^{-4}$	$0.212 \times 10^{-5}$
$10^{-6}$	0.334	530	530	1,852	$0.733 \times 10^{-7}$	$0.230 \times 10^{-6}$
$10^{-8}$	0.860	1,090	1,090	4,088	$0.361 \times 10^{-8}$	$0.153 \times 10^{-8}$

In Tables 3 and 4 the results of the code VE1 are presented. This code approximates the kernel  $k$  by a degenerate one and solves the resulting system of differential equations by an ODE-solver. The parameter DIM, which has to be specified by the user, denotes the dimension of the ODE. It has been chosen as the minimal number such that ERREST, which represents the error caused by approximating the kernel is less than TOL.

TABLE 3  
Results of VE1 for Problem 1 (XEND = 10).

TOL	TIME	NFEV	NKEV	NGEV	DIM	ERREST	ERRACT
$10^{-2}$	0.13	104	58	408	8	$0.331 \times 10^{-2}$	$0.359 \times 10^{-3}$
$10^{-4}$	0.34	204	112	808	11	$0.856 \times 10^{-4}$	$0.128 \times 10^{-4}$
$10^{-6}$	0.92	387	242	1,540	16	$0.377 \times 10^{-6}$	$0.335 \times 10^{-6}$
$10^{-8}$	1.96	646	382	2,576	20	$0.896 \times 10^{-8}$	$0.418 \times 10^{-8}$

TABLE 4  
Results of VE1 for Problem 1 (XEND = 40).

TOL	TIME	NFEV	NKEV	NGEV	DIM	ERREST	ERRACT
$10^{-2}$	0.49	320	112	1,272	11	$0.835 \times 10^{-2}$	$0.256 \times 10^{-1}$
$10^{-4}$	1.45	615	242	2,452	16	$0.735 \times 10^{-4}$	$0.427 \times 10^{-2}$
$10^{-6}$				fail			
$10^{-8}$				fail			

We observe that for small integration intervals VE1 gives correct results, but the computer time is significantly higher than for VOLCON. For large intervals the kernel cannot be easily approximated by polynomials, so that VE1 produces incorrect results for  $TOL = 10^{-2}$  and  $10^{-4}$ . No value DIM ( $\leq 25$ ) could be found such that ERREST is smaller than TOL for  $TOL \leq 10^{-6}$ .

In order to demonstrate that it is worthwhile to exploit the convolution structure in (1), we give in Tables 5 and 6 the results of ORION. This code is written to solve general Volterra integral equations (2).

TABLE 5  
Results of ORION for Problem 1 (XEND = 10).

TOL	TIME	NFEV	NKEV	ERRACT
$10^{-2}$	0.17	37	1,562	$0.156 \times 10^{-3}$
$10^{-4}$	0.46	61	4,277	$0.142 \times 10^{-6}$
$10^{-6}$	1.25	100	11,012	$0.261 \times 10^{-7}$
$10^{-8}$	2.92	142	23,904	$0.533 \times 10^{-9}$

TABLE 6  
Results of ORION for Problem 1 (XEND = 40).

TOL	TIME	NFEV	NKEV	ERRACT
$10^{-2}$	1.09	128	12,593	$0.383 \times 10^{-3}$
$10^{-4}$	4.25	242	45,813	$0.128 \times 10^{-4}$
$10^{-6}$	11.03	345	107,017	$0.616 \times 10^{-7}$
$10^{-8}$	29.04	503	270,117	$0.221 \times 10^{-8}$

ORION gives the correct results. The number of K-evaluations, and therefore also the computer time, is very high, since no use of the convolution structure is made.

*Problem 2.* Equations of the following type arise in the analysis of neural networks with post-inhibitory rebound. The equation below has been modelled after a qualitative

description in an der Heiden [7] on pages 4, 9 and 10.

$$y(x) = 1 + \int_0^x (x-s)^3(4-x+s) e^{-x+s} \frac{y^4(s)}{1+2y^2(s)+2y^4(s)} ds.$$

The exact solution at  $x=10$  is  $y(10)=1.25995582337$ . This value has again been obtained numerically using different codes with very stringent tolerances. Table 7 gives the results of VOLCON.

TABLE 7  
Results of VOLCON for Problem 2 (XEND=10).

TOL	TIME	NFEV	NKEV	NGEV	ERREST	ERRACT
$10^{-2}$	0.017	70	70	204	$0.341 \times 10^{-4}$	$0.203 \times 10^{-4}$
$10^{-4}$	0.018	74	74	220	$0.341 \times 10^{-4}$	$0.203 \times 10^{-4}$
$10^{-6}$	0.052	162	162	504	$0.949 \times 10^{-6}$	$0.134 \times 10^{-5}$
$10^{-8}$	0.368	578	578	2,168	$0.252 \times 10^{-9}$	$0.476 \times 10^{-10}$

The kernel of Problem 2 is exactly decomposable. Problem 2 is therefore equivalent to a 5-dimensional system of ordinary differential equations (cf. [2]). Solving this ODE with the efficient code DIFEX1 (due to Deuffhard [4]), we obtain the results of Table 8.

TABLE 8  
Results of DIFEX1 for Problem 2 (XEND=10).

TOL	TIME	NGEV	ERRACT
$10^{-2}$	0.014	67	$0.776 \times 10^{-2}$
$10^{-4}$	0.039	203	$0.611 \times 10^{-3}$
$10^{-6}$	0.048	256	$0.408 \times 10^{-5}$
$10^{-8}$	0.070	380	$0.116 \times 10^{-7}$

It is seen that VOLCON and DIFEX1 are competitive for  $\text{TOL} \geq 10^{-6}$ . For integral equations with degenerate kernel, whose corresponding differential equation has a dimension greater than 5, VOLCON becomes superior. This is due to the fact that the dimension is significant for the overhead of the ODE-solver. For lower dimensions or for  $\text{TOL} < 10^{-6}$  an ODE-solver like DIFEX1 is to be preferred.

#### REFERENCES

- [1] G. BADER AND P. KUNKEL, *An adaptive multistep method for the solution of second kind Volterra integral equations*, in preparation.
- [2] J. M. BOWNS, *Theory and performance of a subroutine for solving Volterra integral equations*, Computing, 28, (1982), pp. 317-332.
- [3] C. CORDUNEANU, *Integral Equations and Stability of Feedback Systems*, Academic Press, New York, 1973.
- [4] P. DEUFLHARD, *Order and stepsize control in extrapolation methods*, Numer. Math., 41 (1983), pp. 399-422.
- [5] F. G. FRIEDLANDER, *The reflexion of sound pulses by convex parabolic reflectors*, Proc. Cambr. Philos. Soc., 37 (1941), pp. 134-149.
- [6] E. HAIRER, CH. LUBICH AND S. P. NØRSETT, *Order of convergence of one-step methods for Volterra integral equations of the second kind*, SIAM J. Numer. Anal., 20 (1983), pp. 569-579.

- [7] U. AN DER HEIDEN, *Analysis of Neutral Networks*, Lecture Notes in Biomathematics 35, Springer-Verlag, Berlin-Heidelberg-New York, 1980.
- [8] P. HENRICI, *Fast Fourier methods in computational complex analysis*, SIAM Rev., 21 (1979), pp. 481-527.
- [9] H. W. HETHCOTE AND D. W. TUDOR, *Integral equation models for endemic infectious diseases*, J. Math. Biol., 9 (1980) pp. 37-47.
- [10] W. HOCK, *An extrapolation method with step size control for nonlinear Volterra integral equations*, Numer. Math., 38 (1981), pp. 155-178.
- [11] P. POUZET, *Etude en vue de leur traitement numérique des équations intégrales de type Volterra*, Rev. Français Traitement Information (Chiffres) 6 (1963), pp. 79-112.
- [12] H. M. WILLIAMS AND S. MCKEE, *Variable step-size predictor-corrector schemes for second kind Volterra integral equations*, Math. Comp., to appear.

## THE PENETRATION OF A FINGER INTO A VISCIOUS FLUID IN A CHANNEL AND TUBE\*

D. A. REINELT† AND P. G. SAFFMAN†

**Abstract.** The steady-state shape of a finger penetrating into a region filled with a viscous fluid is examined. The two-dimensional and axisymmetric problems are solved using Stokes equations for low Reynolds number flow. To solve the equations, an assumption for the shape of the finger is made and the normal-stress boundary condition is dropped. The remaining equations are solved numerically by covering the domain with a composite mesh composed of a curvilinear grid which follows the curved interface, and a rectilinear grid parallel to the straight boundaries. The shape of the finger is then altered to satisfy the normal-stress boundary condition by using a nonlinear least squares iteration method. The results are compared with the singular perturbation solution of Bretherton (J. Fluid Mech., 10 (1961), pp. 166-188). When the axisymmetric finger moves through a tube, a fraction  $m$  of the viscous fluid is left behind on the walls of the tube. The fraction  $m$  was measured experimentally by Taylor (J. Fluid Mech., 10 (1961), pp. 161-165) as a function of the dimensionless parameter  $\mu U/T$ . The numerical results are compared with the experimental results of Taylor.

**Key words.** viscous fluid, fingering, singular perturbation, composite mesh

**1. Introduction.** We consider the penetration of a finger into a region which is initially filled with a viscous fluid. It is assumed that the viscosity of the fluid inside the finger is negligible when compared with the viscosity of the fluid exterior to the finger. The more general case, where the viscosity of the fluid inside is not neglected, can also be solved with the methods described below but is left for further study. The free boundary value problem for the steady-state shape of the finger is examined with two different geometries: the two-dimensional case of a finger between parallel plates and the axisymmetric case of a finger in a tube. It will be supposed that the gravitational and inertial forces are small in comparison with the viscous forces, and can be neglected. There is, in principle, no difficulty in incorporating their effect into the fingering problem with the present method.

The two-dimensional case is important in the study of fingering in a Hele-Shaw cell composed of two closely spaced parallel plates separated by a distance  $2b$ . The sides of the cell connecting the two plates are a distance  $2l$  apart, where  $l \gg b$ . A finger, shaped like a tongue, moves through the Hele-Shaw cell with constant velocity  $U$ . The thickness of the tongue is  $2\beta b$  and its width is  $2\lambda l$ , where the parameter  $\beta$  is equal to (thickness of finger)/(distance between plates) and the parameter  $\lambda$  is equal to (width of finger)/(width of cell). The determination of the value of  $\lambda$  has been a subject of much interest. Experiments examining the shape of a finger in a Hele-Shaw cell have been performed by Saffman and Taylor (1958) and Pitts (1980). Since the full three-dimensional problem is difficult to calculate, the problem of finding the shape of the finger in the plane parallel to the plates was approximated by averaging the velocity field across the gap between the two plates. This leads to two-dimensional equations in which the components of the mean velocity in the plane parallel to the plates are given by

$$(1) \quad u = -\frac{b^2}{3\mu} \frac{\partial p}{\partial x}, \quad w = -\frac{b^2}{3\mu} \frac{\partial p}{\partial z},$$

\* Received by the editors September 13, 1983, and in revised form January 13, 1984. This work was supported by the U.S. Department of Energy, Office of Basic Energy Sciences, and the Office of Naval Research.

† Department of Applied Mathematics, California Institute of Technology, Pasadena, California 91125.

where  $\mu$  is the viscosity of the fluid and  $p$  is the pressure in the fluid which is to this approximation a function only of  $x$  and  $z$ . The plates are taken parallel to the  $(x, z)$ -plane, and the  $y$ -axis is normal to the plates with origin in the mid plane. The continuity equation

$$(2) \quad \frac{\partial u}{\partial x} + \frac{\partial w}{\partial z} = 0$$

must also be satisfied. These equations hold in the region of the  $(x, z)$ -plane that is not occupied by the finger.

In the region of the  $(x, z)$ -plane where the finger is found, the approximate equations were given by Saffman (1982). In this region, there is on the surface of each plate a film of viscous fluid of total thickness  $mb$ , where  $m \rightarrow 1 - \beta$  away from the edge of the finger, in which the pressure is  $p_i$  and the components of mean velocity are

$$(3) \quad u_i = -\frac{m^2 b^2}{3\mu} \frac{\partial p_i}{\partial x}, \quad w_i = -\frac{m^2 b^2}{3\mu} \frac{\partial p_i}{\partial z}.$$

The continuity equation for the viscous fluid is

$$(4) \quad \frac{\partial m}{\partial t} + \frac{\partial(mu_i)}{\partial x} + \frac{\partial(mw_i)}{\partial z} = 0.$$

The remaining equation is

$$(5) \quad p_i + bT\nabla^2 m = p_0$$

where  $T$  is the surface tension and  $p_0$  is the constant pressure inside the finger, i.e. for  $|y| < (1 - m)b$ .

The two-dimensional solutions in the two regions are joined by boundary conditions at the edge of the finger. First, there is the kinematic condition as the boundary of the finger is approached;

$$(6) \quad (\mathbf{U} - \mathbf{u}) \cdot \mathbf{n} = m(\mathbf{U} - \mathbf{u}_i) \cdot \mathbf{n}$$

where  $\mathbf{U} = (U, 0)$  is the velocity of the finger,  $\mathbf{u} = (u, w)$ , and  $\mathbf{n} = (n_x, n_z)$  is the normal to the edge of the finger. Second, there is a dynamic condition relating the limits of the pressure on the two sides,

$$(7) \quad p - p_i = \Delta p.$$

Here, the limits are to be understood as outer limits in which the distance from the edge of the finger is small compared with  $l$ , but large compared with  $b$ . The limiting values of  $m$  and the pressure jump  $\Delta p$  are so far unknown, but under the postulated conditions, we expect them to have the form

$$(8) \quad m = F\left(\frac{\mu \mathbf{U} \cdot \mathbf{n}}{T}, \frac{b}{R}\right)$$

and

$$(9) \quad \Delta p = \frac{T}{b} f\left(\frac{\mu \mathbf{U} \cdot \mathbf{n}}{T}, \frac{b}{R}\right)$$

where  $R$  is the radius of curvature in the plane parallel to the plates. The functions  $F$  and  $f$  are to be determined by local (inner) solutions of the equations in the vicinity of the finger edge which take into account the  $y$ -dependence of the flow field and

shape of the interface. As such solutions have not been available, it has been the practice to approximate them by the relations

$$(10) \quad F = F_0, \quad f = f_0 + f_1 \frac{b}{R}$$

where  $F_0$ ,  $f_0$ , and  $f_1$  are constants.

Saffman and Taylor (1958) assumed further that the surface tension  $T$  could be neglected ( $f_1 = 0$ ) and were able to derive a closed form solution. They also found that the difference between the shape of the finger determined from their closed form solution and the shape observed from the experimental results was considerable unless  $\lambda$  is close to  $\frac{1}{2}$ , but the parameter  $\lambda$  was not determined by their analysis. McLean and Saffman (1981) have taken into account the effect of the surface tension  $T$  by setting  $f_1 = -1$  in their examination of the fingering problem. This removed the indeterminacy associated with  $T = 0$  and gave  $\lambda$  as a function of  $\mu U/T$ .<sup>1</sup> The shape of the finger with a given value of  $\lambda$  was found to be in close agreement with the shape given by experimental results with the same value of  $\lambda$ . However, a comparison between a plot of  $\lambda$  versus  $\mu U/T$  using these results and the same plot using the experimental results showed significant disagreement. Rough agreement would be obtained for  $f_1$  approximately equal to  $-\frac{1}{2}$ .

In this paper, we calculate

$$(11) \quad F\left(\frac{\mu U}{T}, 0\right) \quad \text{and} \quad f\left(\frac{\mu U}{T}, 0\right)$$

for finite  $\mu U/T$ . To determine these functions, it is necessary to solve the Stokes equations in the plane perpendicular to the plates. Bretherton (1961) has determined  $F$  and  $f$  for  $\mu U/T \ll 1$  ( $b/R = 0$ ) by perturbation methods, and recently Park and Homsy (1983) have determined the  $b/R$  correction for  $\mu U/T \ll 1$  and  $0 < b/R \ll 1$ . The use of boundary conditions incorporating finite  $\mu U/T$  effects and the  $b/R$  dependence can possibly be used to bring the plot of  $\lambda$  versus  $\mu U/T$  into closer agreement with experiments, and also explain the observed stability of the fingers. The results of Romero (1982) who explored the dependence of solutions on an assumed dependence of  $F$  and  $f$  on  $\mu U/T$  showed that the  $b/R$  term is essential to remove the degeneracy of the  $T = 0$  closed form solution.

Besides the two-dimensional problem, we also solve the penetration of an axisymmetric finger into a viscous fluid in a tube. The diameter of the tube is  $2b$  and the diameter of the finger moving through the tube with constant velocity  $U$  is  $2\beta b$ . The parameter  $\beta$  is equal to (diameter of finger)/(diameter of tube). This problem has been investigated experimentally by Taylor (1961) and Cox (1962). The numerical results are compared with the experimental results and the agreement is found to be remarkably good.

In order to determine the solution to the two-dimensional and axisymmetric problems for  $\mu U/T$  equal to  $O(1)$ , the free boundary value problem is solved in two stages. First, we begin with an initial guess for the shape of the finger. This can be found by starting with a small value for the parameter  $\mu U/T$  and using Bretherton's solution. Since we have assumed a shape for the finger, we are forced to drop one of the boundary conditions applied on the curved interface; the normal-stress boundary condition is dropped. A system of equations equivalent to the biharmonic equation

<sup>1</sup> But as later found by Romero (1982) and Vanden-Broeck (1983), there are in fact more than one value of  $\lambda$  for each  $\mu U/T$ .

must now be solved on a fixed domain. It is important to use a numerical method that not only gives accurate results in the interior of the domain but also gives accurate results on the curved interface. To accomplish this, we cover the domain with a composite mesh composed of a curvilinear grid which follows the curved interface, and a rectilinear grid which is parallel to the straight boundaries. These overlapping grids are stretched so that the number of grid points is greatest in regions where they are needed most. Interpolation equations are used to connect the two grids. Finite difference methods are used to calculate the numerical solution.

In the second stage, the shape of the finger is altered to satisfy the normal-stress boundary condition. The curved interface is expanded in terms of Chebyshev polynomials and the known asymptotic behavior of the finger as  $x \rightarrow -\infty$ . Using the solution calculated on the fixed domain, the expansion of the interface, and the normal-stress boundary condition, a new shape for the interface is determined by a nonlinear least squares iteration method. After several iterations, the normal-stress boundary condition is satisfied and we have a solution.

The use of a composite mesh to cover the domain was suggested by Prof. H. O. Kreiss. We considered the employment of boundary integral methods and finite element techniques, but found them less convenient and they did not appear to offer improved accuracy or cheaper computations. The finite element method would have required using higher order elements with one curved side to conform to the interface. Both the composite mesh discussed above and a finite element mesh must be altered each time the interface changes. This was accomplished easily and with a small amount of computation time using the composite mesh technique. Also, the present method allows easy incorporation of inertial and nonuniform fluid effects; this is not the case for the boundary integral method.

**2. Formulation of the two-dimensional problem.** We examine the penetration of a finger of fluid into the narrow region between two closely spaced parallel plates. As mentioned earlier, it is assumed that the viscosity of the fluid inside the finger is negligible when compared with the viscosity of the fluid exterior to the finger. This allows us to solve the equations only in the region exterior to the finger. The steady state problem is examined where the finger is moving with constant velocity  $U$  and is symmetrical about the center line of the channel. The plates are separated by a distance  $2b$  and the finger has asymptotic width  $2\beta b$ .

The Stokes equations for incompressible two-dimensional low Reynolds number flow are

$$(12a) \quad \hat{u}_{\hat{x}} + \hat{v}_{\hat{y}} = 0,$$

$$(12b) \quad \hat{p}_{\hat{x}} = \mu(\hat{u}_{\hat{x}\hat{x}} + \hat{u}_{\hat{y}\hat{y}}),$$

$$(12c) \quad \hat{p}_{\hat{y}} = \mu(\hat{v}_{\hat{x}\hat{x}} + \hat{v}_{\hat{y}\hat{y}}),$$

where  $\hat{p}$  is the pressure and  $\mu$  is the viscosity of the fluid. The velocities  $\hat{u}$  and  $\hat{v}$  of the fluid are in the  $\hat{x}$  and  $\hat{y}$  direction respectively. The  $\hat{y}$ -axis is taken normal to the plates with origin in the mid plane. The tip of the finger moves along the  $\hat{x}$ -axis. Boundary conditions are applied on the plates  $\hat{y} = \pm b$  and on the interface between the two fluids.

We now change to a reference frame moving with the finger. The tip of the finger is fixed at the origin. In this new reference frame, the velocities are independent of



time. Dimensionless variables are introduced by

$$x = \frac{\hat{x} - Ut}{b}, \quad y = \frac{\hat{y}}{b}, \quad R = \frac{\hat{R}}{b},$$

$$u = \frac{\hat{u} - U}{U}, \quad v = \frac{\hat{v}}{U}, \quad p = \frac{\hat{p}}{T/b},$$

where  $R$  is the radius of curvature and  $T$  is the surface tension. In the perturbation analysis of the fingering problem, it becomes clear that the appropriate scaling for  $\hat{p}$  is  $T/b$  and not  $\mu U/b$ . The use of the second scaling results in  $p \rightarrow -\infty$  as  $\mu U/T \rightarrow 0$ . We substitute these new variables into (12) to get

$$(13a) \quad u_x + v_y = 0,$$

$$(13b) \quad p_x = \text{Ca} (u_{xx} + u_{yy}),$$

$$(13c) \quad p_y = \text{Ca} (v_{xx} + v_{yy}),$$

where  $\text{Ca} = \mu U/T$ . The capillary number  $\text{Ca}$  is the ratio of the viscous force to the force of surface tension.

In solving the fingering problem numerically, it is convenient to express the equations in terms of the stream function and the vorticity. We substitute the stream function  $\psi$  defined by

$$u = \psi_y, \quad v = -\psi_x,$$

and the vorticity  $\omega$  defined by

$$\omega = v_x - u_y$$

into (13). If the pressure is eliminated from the equations, we obtain

$$(14a) \quad \psi_{xx} + \psi_{yy} = -\omega,$$

$$(14b) \quad \omega_{xx} + \omega_{yy} = 0.$$

On the interface, it is convenient to use an arc-length coordinate  $s$  equal to zero at the origin and increasing along the curved interface. Using the arc-length coordinate, the tangent vector  $\mathbf{t}$  is equal to  $(x_s, y_s)$ , and the normal vector  $\mathbf{n}$ , pointing into the finger, is equal to  $(-y_s, x_s)$ . The interface conditions are

$$(15a) \quad x_s \psi_x + y_s \psi_y = 0,$$

$$(15b) \quad (y_s^2 - x_s^2)(\psi_{yy} - \psi_{xx}) + 4x_s y_s \psi_{xy} = 0,$$

$$(15c) \quad p - 2 \text{Ca} [(y_s^2 - x_s^2)\psi_{xy} - x_s y_s (\psi_{yy} - \psi_{xx})] = p_0 - \frac{1}{R},$$

where

$$\frac{1}{R} = x_s y_{ss} - y_s x_{ss}.$$

Since the pressure can only be determined up to a constant, we are free to set the constant pressure  $p_0$  inside the finger equal to zero. These three interface conditions can be rewritten as

$$(16a) \quad \psi = 0,$$

$$(16b) \quad \omega - 2x_{ss}\psi_x - 2y_{ss}\psi_y = 0,$$

$$(16c) \quad p - 2 \text{Ca} [(y_s^2 - x_s^2)\psi_{xy} - x_s y_s (\psi_{yy} - \psi_{xx})] + x_s y_{ss} - y_s x_{ss} = 0.$$

The boundary condition (16b) is found by differentiating (15a) with respect to  $s$  and using this equation to eliminate the  $\psi_{xy}$  term in (15b).

It is assumed that the shape of the finger is symmetric in the  $y$  direction; it is then only necessary to solve (14) for  $y \geq 0$ . The symmetry conditions for  $x \geq 0$  are

$$(17a, b) \quad \psi(x, 0) = 0, \quad \omega(x, 0) = 0.$$

In the new reference frame, the no-slip condition on the wall becomes

$$(18a, b) \quad \psi(x, 1) = -(1 - \beta), \quad \psi_y(x, 1) = -1.$$

As  $x \rightarrow -\infty$ , the width of the finger approaches a constant; thus, we get a constant velocity between the finger and the solid boundary. Poiseuille flow develops as  $x \rightarrow \infty$ . The asymptotic behaviors are

$$\begin{aligned} \psi &\rightarrow -y + \beta \quad \text{and} \quad \omega \rightarrow 0 \quad \text{as} \quad x \rightarrow -\infty, \\ \psi &\rightarrow \frac{3}{2}\beta \left[ y - \frac{1}{3}y^3 \right] - y \quad \text{and} \quad \omega \rightarrow 3\beta y \quad \text{as} \quad x \rightarrow \infty. \end{aligned}$$

**3. Asymptotic properties of the solution.** The shape of a finger penetrating into a viscous fluid can be determined by using singular perturbation methods for small  $Ca$ . This work is described in Reinelt (1983). It is an extension of the work of Bretherton (1961) and his analysis of the motion of long bubbles in tubes. The work differs from Bretherton's work in that it outlines a procedure to develop a complete asymptotic expansion in terms of  $Ca$ . It also constructs the equations in the boundary layer region in terms of scaled coordinates of order unity. The method of matched asymptotic expansions is used to connect the inner and outer solutions.

From Bretherton's solution or the solution using singular perturbation methods,  $\beta$  is given by

$$(19) \quad \beta \sim 1.0 - 1.337Ca^{2/3}.$$

This expression, valid for small values of  $Ca$ , holds for both the two-dimensional and axisymmetric problems. It will be compared with the numerical results.

The asymptotic behavior of the solution as  $x \rightarrow -\infty$  can be expanded in powers of  $\exp(kx)$  for finite values of  $Ca$ . This leads to a relationship between  $Ca$ ,  $\beta$ , and  $k$ , the decay rate as  $x \rightarrow -\infty$ . The relationship will also be used to check the numerical results.

For the two-dimensional solution, the stream function takes the form

$$\psi(x, y) \sim -y + \beta + e^{kx}g(y) + O(e^{2kx}).$$

We substitute this expression into (14) to get an equation for  $g(y)$ ,

$$g_{yyyy} + 2k^2g_{yy} + k^4g = 0.$$

The solution to this equation is a combination of the functions  $\sin ky$ ,  $\cos ky$ ,  $y \sin ky$ , and  $y \cos ky$ . If we satisfy the boundary conditions on the wall (18), we get the following expressions for the stream function, vorticity, pressure, and the shape of the interface:

$$\begin{aligned} \psi(x, y) &\sim -y + \beta + e^{kx}[A[k(y-1) \cos k(y-1) - \sin k(y-1)] + Bk(y-1) \sin k(y-1)], \\ \omega(x, y) &\sim 2k^2 e^{kx}[A \sin k(y-1) - B \cos k(y-1)], \\ p(x, y) &\sim -2Ca k^2 e^{kx}[A \cos k(y-1) + B \sin k(y-1)], \\ y(x) &\sim \beta - D e^{kx}, \end{aligned}$$

where  $A$ ,  $B$ , and  $D$  are unknown constants. The above expressions are substituted into the three interface conditions (16a, b, c). If we keep only terms of  $O(\exp kx)$ , then the three equations for  $A$ ,  $B$ , and  $D$  are given by the matrix equation

$$\begin{bmatrix} -q \cos q + \sin q & q \sin q & 1 \\ q \sin q & \cos q & 1 \\ -2Ca(\cos q + q \sin q) & -2Ca q \cos q & 1 \end{bmatrix} \begin{bmatrix} A \\ B \\ D \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

where  $q = k(1 - \beta)$ . The determinant of the matrix must be set equal to zero for a solution other than the trivial solution. This leads to an equation for  $q$  in terms of  $Ca$ ,

$$(20) \quad 2q - \sin 2q + Ca(4q^2 - 4\cos^2 q) = 0.$$

The leading order expansion of (20) as  $Ca \rightarrow 0$  gives

$$(21) \quad q \sim (3Ca)^{1/3}$$

which agrees with the singular perturbation solution. The relationships (20) and (21) between  $k(1 - \beta)$  and  $Ca$  will be compared with the numerical results.

A similar procedure was applied to the axisymmetric problem by Cox (1962) which led to an equation involving the three parameters  $Ca$ ,  $\beta$ , and  $k$ . In the experiments, the value of  $k$  was determined by fitting the finger profile with an exponential curve. In the numerical treatment of the problem,  $k$  is one of the parameters used to describe the interface; its value will be determined by satisfying the equations and boundary conditions.

**4. Numerical solution on a fixed domain.** To solve the fingering problem numerically, we begin with an initial guess for the shape of the finger. The initial guess is found by starting with a small value for  $Ca$  and using the perturbation solution. Since we have assumed a shape for the finger, we are forced to drop one of the three interface conditions (16a, b, c); the normal-stress boundary condition (16c) is dropped. The shape of the finger will be altered to satisfy this condition.

It is important to develop numerical methods that not only give accurate results in the interior of the region, but also give accurate results at the boundaries. To satisfy the normal-stress boundary condition, it is necessary to compute the pressure and the stresses accurately on the boundary. To accomplish this, we cover the domain with a composite mesh composed of a curvilinear grid which follows the curved interface, and a rectilinear grid which is parallel to the straight boundaries. Kreiss (1983) has developed a numerical code that constructs a curvilinear grid using spline interpolation that follows the smooth boundary of a simply connected domain. The rest of the domain is covered with a uniformly spaced rectilinear grid. The overlapping grids are used to solve a system of hyperbolic differential equations. We have modified these methods to treat the elliptic problem in this paper.

In the numerical treatment of the fingering problem, we restrict the infinite domain given by  $-\infty < x < \infty$  and  $0 \leq y \leq 1$  to a finite domain given by  $x_{\min} \leq x \leq x_{\max}$  and  $0 \leq y \leq 1$ . If the values of  $x_{\min}$  and  $x_{\max}$  have been chosen properly, the difference between the numerical solution calculated on this domain and the solution calculated on an even larger domain will be small. As mentioned in § 2, the domain is further restricted to the region exterior to the finger.

In the fingering problem, stretching is used in the curvilinear grid to place more grid points at the tip of the finger and fewer grid points where the width of the finger approaches a constant. To construct the curvilinear grid, we begin with a square grid

with uniformly distributed grid points given by

$$(\hat{s}_i, \hat{r}_j) = (i - 1/N - 1, j - 1/M - 1), \quad \text{where } i = 1, 2, \dots, N \text{ and } j = 1, 2, \dots, M.$$

There are  $N$  grid points in the  $\hat{s}$  direction and  $M$  grid points in the  $\hat{r}$  direction. The curvilinear grid is defined by mapping this square grid onto a region which follows the curved interface using a transformation  $T_c$ . To simplify the interface conditions, it is convenient to use the arclength parameter  $s$  along the interface. Stretching is introduced by the transformations

$$\hat{s} = F(s), \quad \hat{r} = G(r),$$

where  $F$  and  $G$  are functions that produce a one-to-one mapping between the two sets of variables. The functions  $F$  and  $G$  are given in the appendix.

To construct the transformation  $T_c$ , cubic spline interpolation is used to approximate the shape of the curved interface through the  $N$  grid points on the interface of the finger. The boundary  $\hat{r} = 0$  of the square grid is mapped onto the interface curve  $C_1$  by

$$x(s, 0) = X_1(s), \quad y(s, 0) = Y_1(s),$$

where  $X_1$  and  $Y_1$  are cubic spline functions. Another set of  $N$  points is chosen on a curve that lies in the interior of the domain under consideration. The interior curve used is a modified version of the Saffman-Taylor solution

$$\exp [k_1(x - x_1)] = \cos \left[ \frac{\pi y}{2\beta_1} \right]$$

where  $k_1$ ,  $x_1$ , and  $\beta_1$  are chosen constants. The transformation of the boundary  $\hat{r} = 1$  onto the interior curve  $C_M$  is also done by cubic spline interpolation and given by

$$x(s, 1) = X_M(s), \quad y(s, 1) = Y_M(s).$$

The curves,  $C_1$  and  $C_M$ , form the two curved boundaries of the curvilinear grid. The corresponding grid points on these two curves are connected by straight lines. The complete transformation  $T_c$  is

$$(22) \quad \begin{aligned} x(s, r) &= (1 - r)X_1(s) + rX_M(s), \\ y(s, r) &= (1 - r)Y_1(s) + rY_M(s), \end{aligned}$$

with

$$(23) \quad s = F^{-1}(\hat{s}), \quad r = G^{-1}(\hat{r}).$$

The function  $G$  is chosen such that  $G(1) = 1$ . This transformation is one-to-one and its Jacobian is never singular. A typical curvilinear grid is shown in Fig. 1.

Stretching is also used in the rectilinear grid to place a smaller mesh size near  $y = 1$  where the fluid moves into the narrow region between the finger and the wall. In the  $x$  direction, we place fewer grid points near the boundaries at  $x_{\min}$  and  $x_{\max}$ . To construct the transformation  $T_r$ , we begin with another square grid with uniformly distributed grid points given by

$$(\hat{x}_i, \hat{y}_j) = \left( \frac{i - 1}{N_x - 1}, \frac{j - 1}{N_y - 1} \right), \quad \text{where } i = 1, 2, \dots, N_x \text{ and } j = 1, 2, \dots, N_y.$$

The number of grid points in  $\hat{x}$  and  $\hat{y}$  directions are  $N_x$  and  $N_y$ , respectively. The

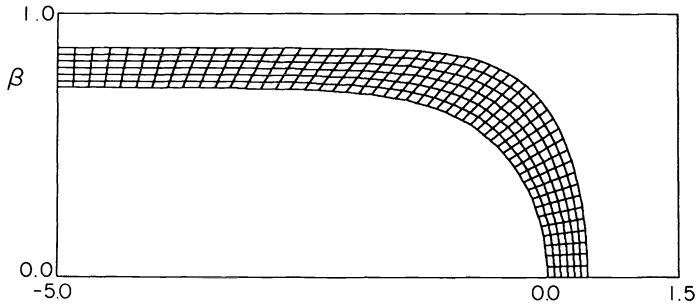


FIG. 1. Typical curvilinear grid.

transformation  $T_r$ , given by

$$(24) \quad \hat{x} = f(x), \quad \hat{y} = g(y),$$

maps the square grid onto the rectilinear grid. The functions  $f$  and  $g$  are given in the appendix.

Many of the grid points in the rectilinear grid are in the interior of the finger. These points are not used in the computation of the solution. Figure 2 gives an example of a rectilinear grid that shows only the grid points actually used. It is important that the grids overlap so that all grid points on  $C_M$  lie in the interior of the rectilinear grid. Also, the grid points on the jagged boundary of the rectilinear grid must lie in the interior of the curvilinear grid.

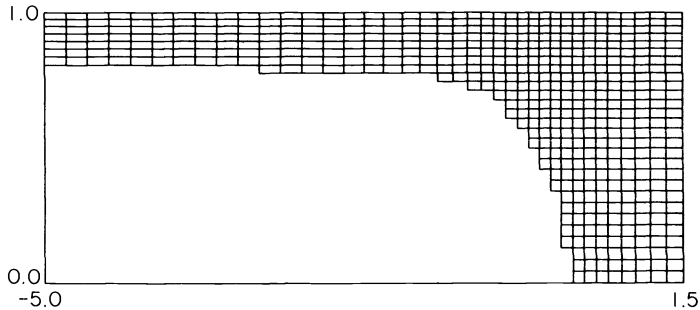


FIG. 2. Typical rectilinear grid.

In solving the equations on a composite mesh, the grid points can be divided into three categories. At interior points of each grid, difference equations that approximate the partial differential equations are applied. At grid points that lie on the boundary of the domain, boundary conditions are applied. The third type of grid points are those that lie on the interior curve  $C_M$  of the curvilinear grid and those that lie on the jagged boundary of the rectilinear grid. It is at these grid points that interpolation equations are used to connect the solutions on the two grids.

At interior points of each grid, the system of equations (14) is replaced by difference equations at the uniformly distributed grid points of the two square grids. These grids are related to the rectilinear and curvilinear grids through the transformations  $T_r$  and  $T_c$ . For example, the stream function equation (14a) is written in terms of  $\hat{x}$  and  $\hat{y}$  coordinates by using (24),

$$(25) \quad f'(x) \frac{\partial}{\partial \hat{x}} \left[ f'(x) \frac{\partial \psi}{\partial \hat{x}} \right] + g'(y) \frac{\partial}{\partial \hat{y}} \left[ g'(y) \frac{\partial \psi}{\partial \hat{y}} \right] = -\omega$$

where

$$x = f^{-1}(\hat{x}), \quad y = g^{-1}(\hat{y}).$$

Using the notation  $\psi_{i,j} = \psi(\hat{x}_i, \hat{y}_j)$ , a difference equation for (25) is given by

$$\begin{aligned} \frac{f'_i}{h_x} \left[ f'_{i+1/2} \left( \frac{\psi_{i+1,j} - \psi_{i,j}}{h_x} \right) - f'_{i-1/2} \left( \frac{\psi_{i,j} - \psi_{i-1,j}}{h_x} \right) \right] \\ + \frac{g'_j}{h_y} \left[ g'_{j+1/2} \left( \frac{\psi_{i,j+1} - \psi_{i,j}}{h_y} \right) - g'_{j-1/2} \left( \frac{\psi_{i,j} - \psi_{i,j-1}}{h_y} \right) \right] = -\omega_{i,j} \end{aligned}$$

where the mesh sizes  $h_x$  and  $h_y$  are

$$h_x = \frac{1}{N_x - 1}, \quad h_y = \frac{1}{N_y - 1}.$$

The expression  $f'_{i+1/2}$  is defined by

$$f'_{i+1/2} = \frac{1}{2}[f'(x_i) + f'(x_{i+1})].$$

This second order accurate difference equation is used at all interior grid points of the  $(\hat{x}, \hat{y})$  square grid. A similar procedure is used to find the difference equations to be applied at the uniformly distributed grid points of the  $(\hat{s}, \hat{r})$  square grid.

The computational boundary conditions for the fingering problem must be chosen carefully. One method of applying the boundary conditions at  $y = 1$  is to construct the grid with the boundary  $y = 1$  centered between the top two grid lines. This allows us to give the value of  $\psi$  on the top two grid lines. However, this approach leads to an  $O(1)$  error in the vorticity near the corner  $x = x_{\max}$  and  $y = 1$ . To avoid these problems, we construct the rectilinear grid with the top grid line coincident with  $y = 1$ . A second order accurate equation for the vorticity on the boundary  $y = 1$  can now be written using  $\psi$  on the top three grid lines and  $\psi_y$  given on the boundary. A more complete discussion of the boundary conditions can be found in Reinelt (1983).

On the interface, the curvilinear grid is constructed with the grid line  $(\hat{r} = 0)$  coincident with the shape of the interface curve. The boundary conditions applied on the interface are written in terms of  $s$  and  $r$  coordinates by

$$\psi = 0, \quad \omega - 2(x_{ss}r_x + y_{ss}r_y)\psi_r = 0.$$

The value of  $\psi_r$  is calculated to second order by using the first three grid lines in the  $r$  direction.

The values of  $\psi$  and  $\omega$  at the third type of grid point are determined by interpolating between the two grids. A nine point formula and a four point formula were examined for this interpolation. The nine point formula was chosen because a test of calculating an exact solution with inhomogeneous boundary conditions showed that a four point formula was not accurate enough. The interpolation equations are discussed in terms of a smooth function  $u$ . To simplify the interpolation formulas, we use formulas based on the uniformly distributed grid points of the two square grids. Each grid point on the curve  $C_M$ , given by  $(s_i, 1)$ , can be located in the interior of the  $(\hat{x}, \hat{y})$  square grid by using (22) and (24). If  $(\hat{x}_0, \hat{y}_0)$  is the location of one of these grid points, then the approximate value of  $u$  at this grid point can be found by using the nine point

interpolation formula given by

$$(26) \quad u(\hat{x}_0, \hat{y}_0) = \sum_{i=1}^3 \sum_{j=1}^3 d_i(\alpha) d_j(\gamma) u(\hat{x}_{I+i-2}, \hat{y}_{J+j-2}),$$

$$d_1(\alpha) = -\frac{1}{2}\alpha(1-\alpha), \quad d_2(\alpha) = (1-\alpha)(1+\alpha), \quad d_3 = \frac{1}{2}\alpha(1+\alpha),$$

$$\alpha = \frac{\hat{x}_0 - \hat{x}_I}{\hat{x}_{I+1} - \hat{x}_I}, \quad \gamma = \frac{\hat{y}_0 - \hat{y}_J}{\hat{y}_{J+1} - \hat{y}_J},$$

where  $(\hat{x}_I, \hat{y}_J)$  is the grid point closest to the point  $(\hat{x}_0, \hat{y}_0)$ .

To find the approximate value of  $u$  at each  $(x, y)$  grid point on the jagged boundary, we locate each of these grid points in the interior of the  $(\hat{s}, \hat{t})$  square grid. These values are found by using Newton's method and (22) and (23). Once these points are located, the interpolation formulas are identical with (26) where  $\hat{x}$  and  $\hat{y}$  are replaced by  $\hat{s}$  and  $\hat{t}$ .

**5. Iteration method.** To determine the degree to which the normal-stress boundary condition is satisfied, it is necessary to find the pressure and the stresses on the interface. The pressure is calculated from the vorticity solution by integrating along the interface. The pressure is given in terms of the vorticity by

$$p_x = -Ca \omega_y, \quad p_y = Ca \omega_x.$$

Using the transformation  $T_c$  and these relationships between the pressure and the vorticity, the derivative of the pressure with respect to arc length is

$$p_s = -Ca [x_s r_y - y_s r_x] \omega_r - Ca [x_s s_y - y_s s_x] \omega_s.$$

The stresses  $\psi_{xx}$ ,  $\psi_{xy}$ , and  $\psi_{yy}$  are calculated at each grid point on the curved interface from the stream function and vorticity solutions. We substitute the initial guess for the shape of the interface and the values of the pressure and stresses at each grid point on the interface into the normal-stress boundary condition (16c). If this boundary condition is satisfied, we have determined the shape of the finger. Normally, the right-hand side of the normal-stress boundary condition is not equal to zero at each grid point, but a residual  $R_i$  is present. These residuals  $R_i, i = 1, 2, \dots, N$  give the error in the boundary condition (16c) at each of the  $N$  grid points along the interface of the finger. In our calculations the value of  $N$  is seventy-six. The shape of the interface must now be changed until all the residuals are smaller than a chosen error tolerance.

To change the shape of the interface, it is convenient to expand the interface in terms of a set of functions and unknown parameters. The shape of the finger is determined by the numerical values of these parameters. The form of the expansion greatly affects the amount of computing time needed to converge to the interface shape that satisfies the normal-stress boundary condition. In fact, if the expansion is not chosen properly, the problem may never converge.

The interface is expanded as a function of  $y$ . The expansion for the shape of the finger is given by

$$(27) \quad x(y) = \frac{1}{k} \log \left\{ \left[ 1 - \left( \frac{y}{\beta} \right)^2 \right] \left[ 1 + \left( \frac{y}{\beta} \right)^2 \sum_{j=0}^m c_j T_{2j} \left( \frac{y}{\beta} \right) \right] \right\}$$

where  $\beta, k, c_0, c_1, \dots, c_m$  are the parameters that determine the shape of the interface. The expansion is constructed so that the tip of the finger is located at the origin and  $x(-y)$  is equal to  $x(y)$ . The functions  $T_{2j}$  are the even Chebyshev polynomials. If the

grid points on the interface are projected onto the  $y$ -axis, there are many more points near the ends of the interval,  $-\beta \leq y \leq \beta$ , than near the center of the interval. This is characteristic of the so-called Chebyshev abscissae. The Chebyshev polynomials are chosen because it is expected that they will converge rapidly given the distribution of grid points used in the fingering problem. This is indeed found to be the case.

The asymptotic behavior of the shape of the finger as  $x \rightarrow -\infty$  is

$$y \sim \beta - D \exp(kx).$$

This relationship is inverted to give

$$x \sim \frac{1}{k} \log \left[ \frac{\beta}{D} \left( 1 - \frac{y}{\beta} \right) \right]$$

as  $y \rightarrow \beta$ . The expansion is constructed so that this asymptotic behavior is included. If this is a good expansion, the value of  $c_j$  will decrease as  $j$  increases. This allows us to use the finite series from  $j = 1$  to  $m$  as a good approximation to the infinite series.

The problem is now reduced to finding the parameters  $\beta, k, c_0, c_1, \dots, c_m$  that satisfy the equations

$$(28a) \quad R_i(\beta, k, c_0, c_1, \dots, c_m) \approx 0, \quad i = 1, 2, \dots, N.$$

This is an overdetermined nonlinear system of equations because the number of grid points ( $N$ ) is larger than the number of parameters ( $m + 3$ ). These equations are solved by determining the parameters that minimize the function

$$\varphi(\beta, k, c_0, c_1, \dots, c_m) = R_1^2 + R_2^2 + \dots + R_N^2.$$

To do this, we linearize the equations (28a) about an initial set of parameters  $\beta^\nu, k^\nu, c_0^\nu, c_1^\nu, \dots, c_m^\nu$ .

$$(28b) \quad R_i(\beta^\nu, k^\nu, c_0^\nu, c_1^\nu, \dots, c_m^\nu) + \frac{\partial R_i}{\partial \beta}(\beta - \beta^\nu) + \dots + \frac{\partial R_i}{\partial c_m}(c_m - c_m^\nu) \approx 0.$$

This leads to a matrix equation containing the  $N \times (m + 3)$  Jacobian of (28a). The new values of the parameters are now determined by the method of least squares. The process is repeated until the values of  $R_i, i = 1, 2, \dots, N$  are smaller than  $1 \times 10^{-3}$ .

In the above calculations there is not a simple functional relationship between  $R_i$  and the unknown parameters because the values of the pressure  $p$  and the stresses  $\psi_{xx}, \psi_{xy},$  and  $\psi_{yy}$  depend on the parameters in some unknown way. In order to calculate the Jacobian of (28a), a small step size  $h$  is added to each parameter independently and the new values of  $R_i$  are determined. For example, we calculate

$$R_i(\beta^\nu + h, k^\nu, c_0^\nu, c_1^\nu, \dots, c_m^\nu)$$

which is used to determine the entries of the Jacobian

$$\frac{\partial R_i}{\partial \beta} = \frac{R_i(\beta^\nu + h, \dots) - R_i(\beta^\nu, \dots)}{h}.$$

$R_i$  is calculated  $m + 3$  times, once for each of the parameters. Each time the interface changes a new curvilinear grid is constructed. The calculations needed to determine the new grid and the transformation  $T_c$  are a very small portion of the total computing time. The major portion of the computing time is needed to determine the numerical values of  $\psi$  and  $\omega$  on each of the fixed domains. In calculating the entries of the Jacobian, we can greatly reduce this time by not solving the entire system of equations directly each time.



On each of the fixed domains a large sparse system of linear equations for the values of  $\psi$  and  $\omega$  at the grid points of the rectilinear and curvilinear grids must be solved. If  $\mathbf{v}$  is the vector that contains  $\psi$  and  $\omega$ , then the system of linear equations can be written

$$\mathbf{A}\mathbf{v} = \mathbf{b}.$$

To solve this system of linear equations, we determine the LU decomposition of the matrix  $\mathbf{A}$ , where  $\mathbf{L}$  is a lower triangular matrix and  $\mathbf{U}$  is an upper triangular matrix. This linear system of equations now decomposes into two triangular systems that are solved by forward substitution and back-substitution. This decomposition of  $\mathbf{A}$  involves a major portion of the computation time and is done using a sparse matrix solver (*odrv*, *ndrv*) developed at Yale University.

In order to calculate the values of  $R_i(\beta'' + h, k'', c_0'', \dots, c_m'')$  and the values of the  $R_i$ 's found by perturbing the other parameters, it is necessary to solve a new system of linear equations

$$\bar{\mathbf{A}}\bar{\mathbf{v}} = \bar{\mathbf{b}}.$$

Since this new system of equations is a perturbation of the original system of equations, it can be rewritten as

$$(\mathbf{A} + \mathbf{A}_1)\bar{\mathbf{v}} = (\mathbf{b} + \mathbf{b}_1)$$

where  $\mathbf{A}$  and  $\mathbf{b}$  are the matrices in the original system. The matrices  $\mathbf{A}_1$  and  $\mathbf{b}_1$  contain the small perturbations to the original system for small values of  $h$ . If we set

$$\bar{\mathbf{v}} = \mathbf{v} + \mathbf{v}_1 + \mathbf{v}_2 + \mathbf{v}_3 + \dots$$

then the solution to the new system of equations can be determined by solving the following equations:

$$\mathbf{A}\mathbf{v}_1 = \mathbf{b}_1 - \mathbf{A}_1\mathbf{v},$$

$$\mathbf{A}\mathbf{v}_2 = -\mathbf{A}_1\mathbf{v}_1,$$

$$\mathbf{A}\mathbf{v}_3 = -\mathbf{A}_1\mathbf{v}_2.$$

$$\vdots$$

Since the LU decomposition of  $\mathbf{A}$  is known and the right-hand side of each of these equations is known from the previous step, these equations are easily solved by forward substitution and back-substitution. In practice, the value of  $\bar{\mathbf{v}}$  is determined to six places by solving only two or three of these equations. Using this method, the computation time necessary to compute the Jacobian is essentially equivalent to the time needed to solve the original system.

**6. Numerical results for the two-dimensional problem.** The numerical results are calculated by beginning with  $\mu U/T = 0.01$  and using the shape of the perturbation solution. Several iterations are needed to satisfy the normal-stress boundary condition. The value of  $\mu U/T$  is then increased by small increments. The size of the increments varied from 0.02 for  $\mu U/T < 0.10$  to 0.20 for  $\mu U/T > 1.00$ . The shape of the interface at the previous value of  $\mu U/T$  is used as the basis for determining the new interface shape at the subsequent value of  $\mu U/T$ . Three or four iterations are needed for the normal-stress boundary condition to be satisfied at each value of  $\mu U/T$  which corresponds to about 25 minutes of CPU on a VAX 11/750.

The typical number of grid points used in each direction of the curvilinear and rectilinear grids is

$$N = 76, \quad M = 7, \quad N_x = 55, \quad N_y = 34.$$

The value of  $x_{\min}$  is determined by the choice of  $s_{\max}$ ;  $x_{\min}$  is approximately equal to  $-5.0$ . The value of  $x_{\max}$  is equal to  $2.0$ . The shape of the finger is determined by using nine parameters ( $m = 6$ ) for the expansion of the interface given in (27). The magnitude of the final coefficient  $c_6$  is  $O(10^{-4})$ . The inclusion of a greater number of parameters has very little effect on the shape of the finger.

Figure 3 is a plot of  $q = k(1 - \beta)$  versus  $\mu U/T$ . The solid line is a plot of (20) which was determined by expanding the solution in terms of eigenfunctions as  $x \rightarrow -\infty$ .

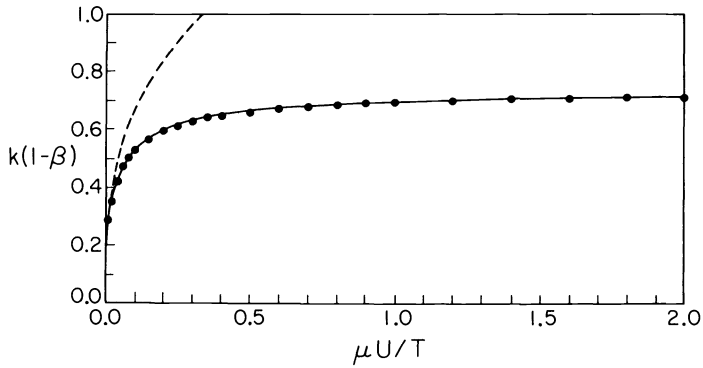


FIG. 3. The relationship  $k(1 - \beta)$  versus  $\mu U/T$ . •, numerical results; —, plot of equation (20); ---, perturbation result (21).

The dots shown on the plot are the values of  $k(1 - \beta)$  calculated from the numerical results. The numerical results are in close agreement with the analytical result. The dashed line is a plot of the perturbation solution (21). As mentioned in § 3, it is equivalent to the leading order behaviour of (20). The perturbation solution (21) is in error by no more than 10% provided  $\mu U/T < 2 \times 10^{-2}$ . In Fig. 4, the solid line is a plot of  $\beta$  versus  $\mu U/T$  calculated from the numerical solutions and the dashed line is a plot of (19) determined from the perturbation solution. The function  $F$  discussed

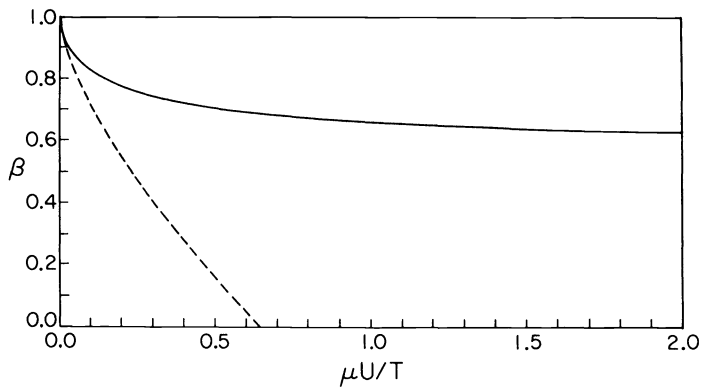


FIG. 4. The finger width  $\beta$  versus  $\mu U/T$  for the two-dimensional problem. —, numerical results; ---, perturbation result (19).

in the introduction is given by

$$1 - \beta = F\left(\frac{\mu U}{T}, 0\right).$$

As  $x \rightarrow \infty$ , the pressure is given by

$$p \sim -3\beta Ca x + c_p$$

where  $c_p$  is a constant. It is this constant that gives the pressure jump that is needed in the two-dimensional approximation of the flow in the Hele-Shaw cell,

$$c_p = f\left(\frac{\mu U}{T}, 0\right).$$

The dotted line in Fig. 5 is a plot of  $c_p$  versus  $\mu U/T$  and the solid line is the actual pressure drop  $\Delta p$  across the tip of the finger. The dashed line is the pressure drop calculated by Bretherton (1961) for  $Ca \rightarrow 0$  and is given by

$$\Delta p \sim -1.0 - 3.8 Ca^{2/3}.$$

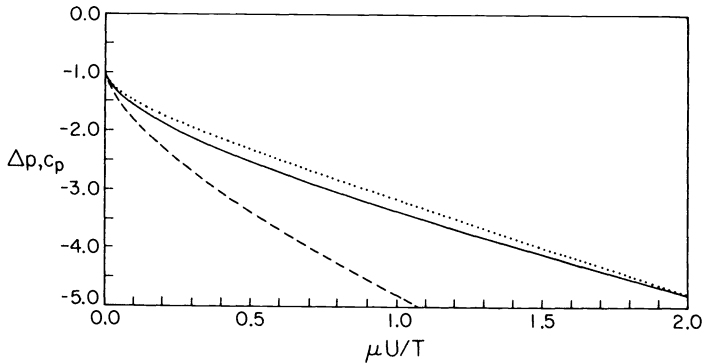


FIG. 5. Pressure drop  $\Delta p$  across tip of the finger versus  $\mu U/T$  for the two-dimensional problem. —, numerical results; ---, perturbation result; ····,  $c_p$  versus  $\mu U/T$ .

This perturbation solution is equal to  $c_p$  up to the order calculated. Both  $c_p$  and  $\Delta p$  have been normalized by  $T/b$ .

As  $x \rightarrow \infty$ , the velocity in the  $x$ -direction is

$$u \rightarrow \frac{3}{2}\beta(1 - y^2) - 1.$$

When the value of  $\beta$  is greater than  $\frac{2}{3}$ , the fluid near the  $x$ -axis moves with a velocity greater than that of the finger. In this case, two additional stagnation points are present on the interface. For all values of  $\beta$ , there is a stagnation point at the tip of the finger. Figure 6 gives examples of the streamlines in the two cases.

**7. Numerical solution of the axisymmetric problem.** We consider the penetration of a finger into a tube. As in the two-dimensional case, the steady state problem is examined and the finger moves parallel to the  $x$ -axis with constant velocity  $U$ . The diameter of the tube is  $2b$  and the diameter of the finger is  $2\beta b$ . The parameter  $\beta$  is equal to (diameter of finger)/(diameter of tube).

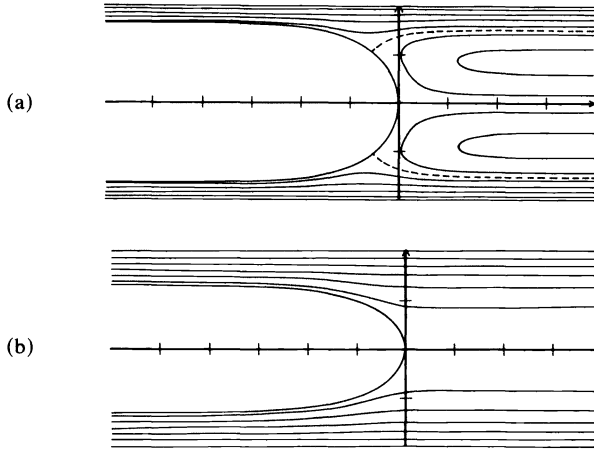


FIG. 6. Typical shape of streamlines relative to the finger. (a)  $\beta > \frac{2}{3}$ , two-dimensional case;  $\beta > 1/\sqrt{2}$ , axisymmetric case. (b)  $\beta > \frac{2}{3}$ , two-dimensional case;  $\beta > 1/\sqrt{2}$ , axisymmetric case.

The same dimensionless variables used in the two-dimensional case are used here. We write the equations in terms of the stream function  $\psi$  defined by

$$u = \frac{1}{y} \psi_y, \quad v = -\frac{1}{y} \psi_x,$$

and the vorticity  $\omega$  defined by

$$\omega = v_x - u_y.$$

The equations for  $\psi$  and  $\omega$  in axisymmetric Stokes flow are

$$(29a) \quad \omega_{xx} + \psi_{yy} - \frac{1}{y} \psi_y = -y\omega,$$

$$(29b) \quad \omega_{xx} + \omega_{yy} + \frac{1}{y} \omega_y - \frac{1}{y^2} \omega = 0.$$

The variable  $y$  is used for the radial coordinate to avoid confusion with the  $r$  coordinate used in the curvilinear grid.

The interface is described by  $(x(s), y(s))$  where  $s$  is the arclength along the interface curve. In the axisymmetric case, the boundary conditions on the interface are given by

$$(30a) \quad \psi = 0,$$

$$(30b) \quad y\omega - 2x_{ss}\psi_x - 2y_{ss}\psi_y = 0,$$

$$(30c) \quad p - 2 \text{Ca} [y_s^2 u_x - x_s y_s (v_x + u_y) + x_s^2 v_y] + \left( \frac{1}{R_1} + \frac{1}{R_2} \right) = 0,$$

where

$$u_x = \frac{1}{y} \psi_{xy}, \quad u_y = \frac{1}{y} \psi_{yy} - \frac{1}{y^2} \psi_y,$$

$$v_x = -\frac{1}{y} \psi_{xx}, \quad v_y = -\frac{1}{y} \psi_{xy} + \frac{1}{y^2} \psi_x.$$

The principal curvatures for the axisymmetric problem are

$$\frac{1}{R_1} = x_s y_{ss} - y_s x_{ss}, \quad \frac{1}{R_2} = -\frac{x_s}{y}.$$

The pressure  $p_0$  is the constant pressure inside the finger and is set equal to zero. The boundary conditions on the wall of the tube are

$$(31a, b) \quad \psi(x, 1) = -\frac{1}{2}(1 - \beta^2), \quad \psi_y(x, 1) = -1,$$

and the symmetry conditions on the centerline are

$$(32a, b) \quad \psi(x, 0) = 0, \quad \omega(x, 0) = 0.$$

The asymptotic behaviours of  $\psi$  and  $w$  are

$$\begin{aligned} \psi &\rightarrow -\frac{1}{2}(y^2 - \beta^2) \quad \text{and} \quad \omega \rightarrow 0 \quad \text{as} \quad x \rightarrow -\infty, \\ \psi &\rightarrow \frac{1}{2}\beta^2(2y^2 - y^4) - \frac{1}{2}y^2 \quad \text{and} \quad \omega \rightarrow 4\beta^2 y \quad \text{as} \quad x \rightarrow \infty. \end{aligned}$$

As in the two-dimensional case, the normal-stress boundary condition is dropped, and the numerical solution is computed on a fixed domain. The normal-stress boundary condition is used to find the shape of the finger.

The results for the axisymmetric problem are very similar to the two-dimensional results. In Fig. 7, the solid line is a plot of  $\beta$  versus  $\mu U/T$  calculated from the numerical

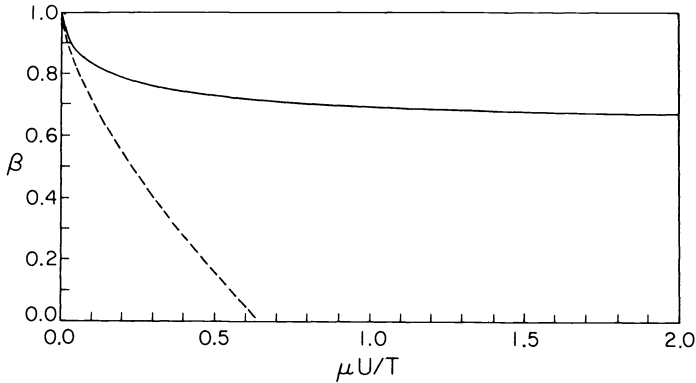


FIG. 7. The finger width  $\beta$  versus  $\mu U/T$  for the axisymmetric problem. —, numerical results; ---, perturbation result (19).

solutions and the dashed line is the perturbation result (19). As  $x \rightarrow \infty$ , the pressure is given by

$$p \sim -8\beta^2 Ca x + c_p$$

where  $c_p$  is a constant. The solid line in Fig. 8 shows  $\Delta p$ , the pressure drop across the tip of the finger, and the dotted line gives  $c_p$ . Both  $\Delta p$  and  $c_p$  have been normalized by  $T/b$ .

As  $x \rightarrow \infty$ , the velocity in the  $x$ -direction is

$$u \rightarrow 2\beta^2(1 - y^2) - 1.$$

For  $\beta$  greater than  $1/\sqrt{2}$ , the fluid near the  $x$ -axis moves faster than the finger. Taylor (1961) discusses the two simplest types of flows that might occur: a stagnation point

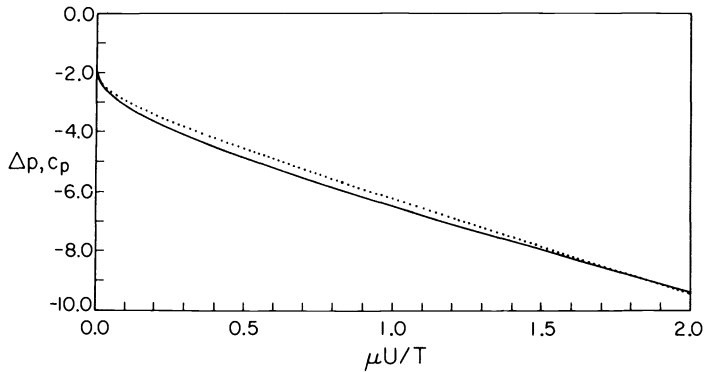


FIG. 8. —, pressure drop  $\Delta p$  across tip of the finger versus  $\mu U/T$  for the axisymmetric problem;  $\cdots$ ,  $c_p$  versus  $\mu U/T$ .

at the origin with a stagnation ring on the interface of the finger or two stagnation points on the  $x$ -axis, one of which is at the origin. By examining Fig. 6, it is clear that a stagnation ring is present for  $\beta$  greater than  $1/\sqrt{2}$ .

When the axisymmetric finger moves through the tube, a fraction  $m$  of the viscous fluid is left behind on the walls of the tube. The fraction  $m$  was measured experimentally by Taylor as a function of  $\mu U/T$ . Figure 9 compares the numerical results with the experimental results where  $m$  is equal to  $1 - \beta^2$ . The numerical results are in excellent agreement with the experimental results.

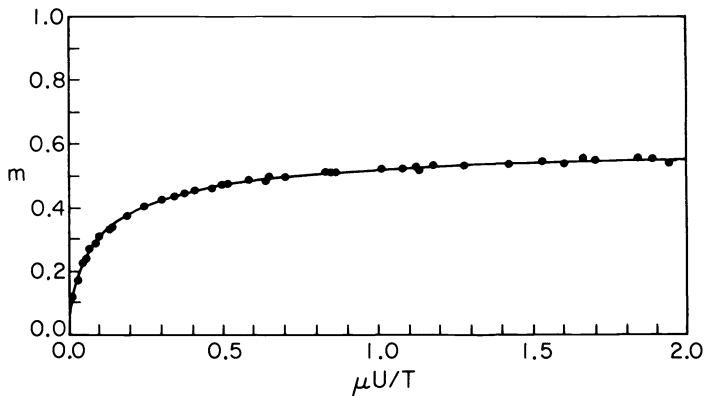


FIG. 9. Fraction  $m$  of viscous fluid left behind on the walls of the tube versus  $\mu U/T$ . —, numerical result;  $\bullet$ , Taylor experimental results.

**8. Conclusion.** In solving the fingering problem, we have used a composite mesh to cover the domain. The resulting numerical solution is not only accurate in the interior of the region but also on the boundaries of the domain. The amount of computing time necessary to construct the grids is a very small percentage of the time necessary to compute the solution to the fingering problem.

The employment of a composite mesh creates enough flexibility that it can be used to treat problems with many different types of geometries. It can also be used in determining solutions that exhibit singular behavior. The composite mesh can be composed of as many grids as necessary to solve a given problem. The grids are easily constructed to include stretching which places grid points where they are needed most.

The numerical methods employed work very well in the treatment of this free surface problem. Many other free surface problems could be examined by extending the methods to include the effects of the inertia terms. The effects of gravity on the shape of the finger for the two-dimensional and axisymmetric geometries can also be calculated. The methods could also be extended to handle time-dependent free surface problems. In these problems, the curvilinear grid would move with the interface at each time step of the calculation.

**Appendix.** As discussed earlier, stretching functions are introduced to place grid points where they are needed most. In the  $x$  direction, fewer grid points are needed near  $x_{\min}$  and  $x_{\max}$  where the solution tends to a function of  $y$  only. The  $x$  dependence is a decaying exponential. The function  $f$  takes the form

$$\hat{x} = f(x) = Ax + B + C \cdot D \tanh\left(\frac{x - x_0}{D}\right)$$

and the first derivative is

$$\frac{d\hat{x}}{dx} = f'(x) = A + C \operatorname{sech}^2\left(\frac{x - x_0}{D}\right).$$

$A$  and  $C$  are chosen such that there is a larger mesh size near the boundaries and a smaller mesh size in an interior region centered about the point  $x_0$ . The constant  $D$  is the decay rate from the smaller mesh size to the larger one.  $B$  is chosen such that  $f(x_{\min}) = 0$ .

In the perturbation problem, it was found that for small  $Ca$  the finger nearly fills the channel. To numerically solve the fingering problem for this case, it is necessary to have a small mesh size near  $y = 1$ . The stretching in the  $y$  direction takes the form

$$\hat{y} = g(y) = Ay + C \cdot D \left[ \exp\left(\frac{-(1-y)}{D}\right) - \exp\left(\frac{-(1+y)}{D}\right) \right]$$

where the first derivative is given by

$$\frac{d\hat{y}}{dy} = g'(y) = A + C \left[ \exp\left(\frac{-(1-y)}{D}\right) + \exp\left(\frac{-(1+y)}{D}\right) \right].$$

The constants  $A$  and  $C$  are chosen to produce a small mesh size near  $y = 1$  and a larger one away from  $y = 1$ .  $D$  is the decay rate between the two mesh sizes.

In the  $s$  direction, we use a stretching transformation that produces more grid points in the region near the tip of the finger and fewer in the region where the width approaches a constant. The transformation is given by

$$\hat{s} = F(s) = As + C \cdot D \tanh\left(\frac{s}{D}\right)$$

where the derivative is

$$\frac{d\hat{s}}{ds} = F'(s) = A + C \operatorname{sech}^2\left(\frac{s}{D}\right).$$

Again,  $A$  and  $C$  are chosen to produce the appropriate mesh sizes and  $D$  is the decay rate. In the fingering problem it is not necessary to stretch in the  $r$  direction, so we simply set  $\hat{r} = G(r) = r$ .

**Acknowledgments.** We wish to thank Prof. H. O. Kreiss for suggesting the composite mesh method and B. Kreiss for help with the initial implementation.

## REFERENCES

- G. K. BATCHELOR (1967), *An Introduction to Fluid Mechanics*, Cambridge Univ. Press, London.
- F. P. BRETHERTON (1961), *The motion of long bubbles in tubes*, J. Fluid Mech., 10, pp. 166-188.
- B. G. COX (1962), *On driving a viscous fluid out of a tube*, J. Fluid Mech., 14, pp. 81-96.
- G. DAHLQUIST AND A. BJORCK (1974), *Numerical Methods*, Prentice-Hall, Englewood Cliffs, NJ.
- B. KREISS (1983), *Construction of a curvilinear grid*, this Journal, 4, pp. 270-279.
- H. LAMB (1932), *Hydrodynamics*, Sixth Edition, Dover, New York.
- J. W. MCLEAN AND P. G. SAFFMAN (1980), *The effect of surface tension on the shape of fingers in a Hele-Shaw cell*, J. Fluid Mech., 102, pp. 455-469.
- C. W. PARK AND G. M. HOMSY (1983), *Two-phase displacement in Hele-Shaw cells*, submitted for publication.
- E. PITTS (1980), *Penetration of fluid into a Hele-Shaw cell: the Saffman-Taylor experiment*, J. Fluid Mech., 97, pp. 53-64.
- D. A. REINELT (1983), Ph.D. dissertation, California Institute of Technology, Pasadena.
- P. J. ROACHE (1976), *Computational Fluid Dynamics*, Hermosa, Albuquerque, NM.
- L. A. ROMERO (1982), Ph.D. dissertation, California Institute of Technology, Pasadena.
- P. G. SAFFMAN (1982), *Fingering in porous media*, Lecture Notes in Physics, 154, Burridge et al., eds., Springer-Verlag, pp. 208-214.
- P. G. SAFFMAN AND G. I. TAYLOR (1958), *The penetration of a fluid into a porous medium or Hele-Shaw cell containing a more viscous liquid*, Proc. Roy. Soc. A, 245, pp. 312-329.
- G. I. TAYLOR (1961), *Deposition of a viscous fluid on the wall of a tube*, J. Fluid Mech., 10, pp. 161-165.
- J.-M. VANDEN-BROECK (1983), *Fingers in a Hele-Shaw cell with surface tension*, Phys. Fluids, 26 (8), pp. 2033-2034.



## ON THE MINRES METHOD OF FACTOR ANALYSIS\*

FRANKLIN T. LUK†

**Abstract.** The minres method is an effective means for estimating factor loadings under the condition that the sum of squares of the off-diagonal residuals be minimized. This paper concerns the efficient implementation and the convergence properties of the method.

**Key words.** minres method, factor analysis, least squares, communality

**1. Introduction.** A basic problem in factor analysis (Harman [8, Chap. 5]) is the resolution of  $n$  observed variables  $z_j$  linearly in terms of a smaller number  $m$  of common factors  $f_p$ . We choose the classical model:

$$z_j = \sum_{p=1}^m a_{jp}f_p + u_j \quad \text{for } j = 1, 2, \dots, n,$$

where  $a_{jp}$  are unknown parameters called common-factor loadings and  $u_j$  represent the errors. In matrix form we have

$$z = Af + u,$$

where  $A \in \mathbf{R}^{n \times m}$ . Under the assumption of uncorrelated factors, the fundamental theorem of factor analysis (Thurstone [15, p. 70]) states that a matrix  $\hat{C} \in \mathbf{R}^{n \times n}$  of reproduced correlations is given by

$$\hat{C} = AA^T.$$

Our problem is to determine  $A$  so that the reproduced matrix  $\hat{C}$  is a good approximation to the observed correlation matrix  $C \equiv (c_{ij}) \in \mathbf{R}^{n \times n}$ , which is a symmetric and positive semi-definite (psd) matrix with ones down the principal diagonal. The fit of  $\hat{C}$  to  $C$  naturally improves with increasing value of  $m$ . The problem of choosing a proper number of common factors is discussed in [8, pp. 183-186]. We assume a fixed  $m$  for this paper.

Suppose we desire a psd matrix  $\hat{C}$  such that

$$\|C - \hat{C}\|_F = \min,$$

where  $\|\cdot\|_F$  denotes the Frobenius matrix norm. We compute an eigenvalue decomposition:

$$C = QDQ^T,$$

where  $Q = (q_1, \dots, q_n)$  is orthogonal,  $D = \text{diag}(d_1, \dots, d_n)$  and  $d_1 \geq \dots \geq d_n \geq 0$ . The solution is given by

$$\hat{C} = Q_1 D_1 Q_1^T,$$

where  $Q_1 = (q_1, \dots, q_m) \in \mathbf{R}^{n \times m}$  and  $D_1 = \text{diag}(d_1, \dots, d_m) \in \mathbf{R}^{m \times m}$ . It follows that

$$A = Q_1 D_1^{1/2}.$$

---

\* Received by the editors August 25, 1981, and in revised form October 10, 1983. This research was supported in part by the U.S. Army Research Office under grant DAAG 29-79-C0124 and by the National Science Foundation under grant MCS-8213718.

† Department of Computer Science, Cornell University, Ithaca, New York 14853.

This procedure is known as the principal-component method [8, Chap. 8] and it optimally reproduces the total variance of the variables. However, a major objective of factor analysis is to best reproduce the observed correlations; Thurstone [16, p. 61] states, "The object of a factor problem is to account for the tests, or their intercorrelations, in terms of a small number of derived variables, the smallest possible number that is consistent with acceptable residual errors." Consequently, Harman and Jones [10] propose that one approximates only the off-diagonal elements of  $C$ . That is, they want to choose  $A$  to minimize the objective function

$$(1.1) \quad f(A) \equiv \sum_{j=1}^n \sum_{\substack{k=1 \\ k \neq j}}^n \left( c_{jk} - \sum_{p=1}^m a_{jp} a_{kp} \right)^2.$$

Several optimization procedures are compared in [10] and the conclusion is drawn that a simple Gauss-Seidel approach by the name of the minres method ("minimum residuals") is most efficient. However, in order that the factor solution be acceptable, the diagonal elements (the communalities) of  $\hat{C}$  must be bounded by unity in value [8, pp. 116-117], i.e.,

$$\sum_{p=1}^m a_{jp}^2 \leq 1 \quad \text{for } j = 1, 2, \dots, n.$$

Harman and Fukuda [9] present a modified minres method for the constrained problem

$$(1.2) \quad \min \left\{ f(A) : \sum_{p=1}^m a_{jp}^2 \leq 1, \text{ for } j = 1, 2, \dots, n \right\}$$

and remark that the new method is just as efficient as the original approach.

This paper concerns the efficient implementation and the convergence properties of the minres method.

**2. Minres method.** Harman and Jones [10] adopt the idea of the block Gauss-Seidel technique for linear equations. The minres method is an iterative procedure in which changes are made to a row of the factor matrix  $A$  at a time. With the notation

$$A \equiv \begin{pmatrix} a_1^T \\ a_2^T \\ \vdots \\ a_n^T \end{pmatrix} \equiv (a_{ij}),$$

let us examine the effects of replacing  $a_j^T$  by the vector

$$x^T \equiv (x_1, x_2, \dots, x_m).$$

The new reproduced correlations of variable  $j$  with any other variable  $k$  is

$$c_{jk} = \sum_{p=1}^m a_{kp} x_p,$$

and the sum of squares of residual correlations of variable  $j$  with all other variables becomes

$$(2.1) \quad f_j(x) = \sum_{\substack{k=1 \\ k \neq j}}^n \left( c_{jk} - \sum_{p=1}^m a_{kp} x_p \right)^2.$$

Harman and Jones want to minimize  $f_j(x)$ , i.e., they are interested in the least squares problem

$$(2.2) \quad \min \{ \|b_j - A_{-j}x\|_2 \},$$

where

$$b_j = (c_{j1}, \dots, c_{j,j-1}, c_{j,j+1}, \dots, c_{jn})^T$$

and

$$A_{-j} = \begin{pmatrix} a_1^T \\ \vdots \\ a_{j-1}^T \\ a_{j+1}^T \\ \vdots \\ a_n^T \end{pmatrix}.$$

They solve the normal equations:

$$A_{-j}^T A_{-j} x = A_{-j}^T b_j,$$

implicitly assuming that the matrix  $A_{-j}$  has full column rank. Since the objective function can be rewritten as

$$(2.3) \quad f(A(x)) = \sum_{\substack{i=1 \\ i \neq j}}^n \sum_{\substack{k=1 \\ k \neq i,j}}^n \left( c_{ik} - \sum_{p=1}^m a_{ip} a_{kp} \right)^2 + 2f_j(x),$$

we see that minimizing  $f_j(x)$  is equivalent to minimizing  $f(A(x))$ . From here on we shall let  $\| \cdot \| \equiv \| \cdot \|_2$  denote the Euclidean vector norm.

As mentioned in the Introduction, Harman and Fukuda [9] replace (2.2) by the inequality-constrained least squares problem:

$$(2.4) \quad \min \{ \|b_j - A_{-j}x\| : \|x\| \leq 1 \}.$$

They use the method of Lagrange multipliers, i.e., an  $\lambda \geq 0$  is to be found such that if

$$(2.5) \quad (A_{-j}^T A_{-j} + \lambda I)x = A_{-j}^T b_j,$$

then either  $\lambda = 0$  and  $\|x\| \leq 1$ , or  $\lambda > 0$  and  $\|x\| = 1$ . Let an eigenvalue decomposition of  $A_{-j}^T A_{-j}$  be

$$A_{-j}^T A_{-j} = P \Delta P^T,$$

where  $P$  is orthogonal,  $\Delta = \text{diag}(\delta_1, \dots, \delta_m)$  and  $\delta_1 \geq \dots \geq \delta_m > 0$ . With  $h = P^T A_{-j}^T b_j \equiv (h_1, \dots, h_m)^T$  and  $y = \Delta P^T x \equiv (y_1, \dots, y_m)^T$ , (2.5) becomes

$$(I + \lambda \Delta^{-1})y = h.$$

So

$$\lambda = \frac{\delta_i (h_i - y_i)}{y_i},$$

for  $i = 1, \dots, m$ . Given  $y_1$ , the unknowns  $y_p$  are computed from

$$(2.6) \quad y_p = \frac{h_p \delta_p y_1}{(\delta_p - \delta_1) y_1 + h_1 \delta_1},$$

for  $p = 2, \dots, m-1$ , and  $y_m$  from

$$(2.7) \quad \frac{y_1^2}{\delta_1} + \dots + \frac{y_m^2}{\delta_m} = 1.$$

Under the assumption, all  $h_p > 0$  and  $0 < y_1 < \min \{h_1, \sqrt{\delta_1}\}$ , it is proved in [9] that

$$\delta_1 \left[ 1 - \frac{h_1}{y_1} \right] \geq \delta_m \left[ 1 - \frac{h_m}{y_m} \right] \Rightarrow y_1 \geq \hat{y}_1,$$

where  $\hat{y}_1$  denotes the true solution for  $y_1$ . The value of  $y_1$  can therefore be determined to any specified accuracy. However, no explicit updating formula for  $y_1$  is given in [9].

The key step of the minres method is the replacement of the  $j$ th row of  $A$  by the solution to the inequality-constrained problem (2.4), for  $j = 1, 2, \dots, n$ . This sequence of  $n$  problems constitutes a major iteration cycle. The initial matrix is composed of the first  $m$  principal components of  $C$ . Since  $c_{ii} = 1$  for all  $i$ , the  $j$ th row  $a_j^{(0)T}$  of the starting matrix satisfies

$$\|a_j^{(0)}\| \leq 1,$$

for  $j = 1, 2, \dots, n$ . The convergence criterion in [9], [10] is that the factor matrices  $A^c \equiv (a_{jk}^c)$  and  $A^- \equiv (a_{jk}^-)$  at the end of the current and the previous cycle, respectively, satisfy the condition:

$$(2.8) \quad \max_{j,k} |a_{j,k}^c - a_{j,k}^-| < \tau,$$

where  $\tau$  is some prechosen tolerance. Harman [8, p. 188] states that  $\tau = .001$  is satisfactory for most problems. We may summarize the minres method as follows.

#### ALGORITHM MINRES

##### Initialization phase.

Compute an eigenvalue decomposition of  $C$ :

$$C = QDQ^T,$$

where  $Q = (q_1, \dots, q_n)$ ,  $D = \text{diag}(d_1, \dots, d_n)$  and  $d_1 \geq \dots \geq d_n \geq 0$ ;

Set

$$A = Q_1 D_1^{1/2},$$

where  $Q_1 = (q_1, \dots, q_m)$  and  $D_1 = \text{diag}(d_1, \dots, d_m)$ ;

##### Iterative phase.

repeat until convergence

for  $j = 1, \dots, n$  do

begin

Solve  $A_{-j}^T A_{-j} x = A_{-j}^T b_j$ ;

if  $\|x\| > 1$  then

begin

Determine an eigenvalue decomposition:

$$A_{-j}^T A_{-j} = P \Delta P^T;$$

Let  $h = P^T A_{-j}^T b_j$ ;

Set  $y_1 = \frac{h_1}{[\sum_{p=1}^m (h_p^2 / \delta_p)]^{1/2}}$ ;

```

repeat until convergence {find  $y_1$  s.t.  $|\delta_1(1 - h_1/y_1) - \delta_m(1 - h_m/y_m)|$  is small}
begin
  Update  $y_1$ ; {unnecessary when loop is first entered}
  Use (2.6) to compute  $y_p$ , for  $p = 2, \dots, m - 1$ ;
  Use (2.7) to compute  $y_m$ 
end;
Set  $x = P\Delta^{-1}y$ 
end;
Replace the  $j$ th row of  $A$  by  $x^T$ 
end.
    
```

**3. Implementation.** We consider the matrix equation (2.5), assuming that  $A_{-j}$  has full column rank. Numerical examples ([8], [9], [10] and § 5) indicate it is likely that  $\lambda = 0$  at the solution. The equivalent least squares problem (2.2) is best solved via an orthogonal triangularization of  $A_{-j}$  (Golub [6] and LINPACK [3]):

$$(3.1) \quad A_{-j} = QR,$$

where  $Q \in \mathbf{R}^{(n-1) \times m}$  has orthonormal columns and  $R \in \mathbf{R}^{m \times m}$  is upper triangular. For  $j \geq 2$ , the two matrices  $A_{-j}$  and  $A_{-(j-1)}$  differ only in their  $(j-1)$ st row:

$$A_{-j} = A_{-(j-1)} + e_{j-1}(a_{j-1}^T - a_j^T),$$

where  $e_{j-1}$  denotes the  $(j-1)$ st unit coordinate vector. For  $j = 1$  note that

$$A_{-1} = \Pi[A_{-n} + e_1(a_n^T - a_1^T)],$$

where  $\Pi$  is the permutation matrix of order  $n-1$ :

$$\Pi = (e_{n-1}, e_1, e_2, \dots, e_{n-2}).$$

The updating procedure of Daniel et al. [2] may be applied to compute a QR-factorization of  $A_{-j}$  (respectively  $A_{-1}$ ) from that of  $A_{-(j-1)}$  (respectively  $A_{-n}$ ), using approximately  $2(\xi + 3)nm + 3m^2$  operations (multiplications and additions), where  $\xi$  is the number of orthogonalization steps ( $\xi - 1$  reorthogonalizations). We may use the special structure of the rank 1 matrix and save  $nm$  operations. The solution  $x$  is given by

$$Rx = Q^T b_j.$$

A better (though more expensive) numerical strategy is to compute the change  $u \equiv x - a_j$ . That is, we solve

$$Ru = Q^T (b_j - A_{-j} a_j),$$

and set

$$x = a_j + u.$$

If  $\lambda > 0$  we recognize (2.5) as the normal equations for the problem:

$$(3.2) \quad \min \left\{ \left\| \begin{pmatrix} A_{-j} \\ \sqrt{\lambda} I \end{pmatrix} x - \begin{pmatrix} b_j \\ 0 \end{pmatrix} \right\| \right\},$$

or, equivalently, .

$$(3.3) \quad \min \left\{ \left\| \begin{pmatrix} R \\ \sqrt{\lambda} I \end{pmatrix} x - \begin{pmatrix} Q^T b_j \\ 0 \end{pmatrix} \right\| \right\},$$

to which we may apply the algorithms of Elden [4], Golub [7] or Moré [12] (Gander [5] contains a detailed discussion). All three procedures involve a preliminary reduction of  $R$  to some simple form:

$$R = WJY^T,$$

where  $W$  and  $Y$  are orthogonal and  $J$  is upper bidiagonal in [4],  $W$  and  $Y$  are orthogonal and  $J$  is nonnegative diagonal in [7], and  $W$  is orthogonal,  $J$  is upper triangular and  $Y$  is a permutation matrix in [12]. That is, Elden [4] bidiagonalizes  $R$ , Golub [7] determines its singular value decomposition (SVD), and Moré [12] computes its QR-factorization with column pivoting. Since no advantage can be taken of the zeros in  $R$  ([1], [4]), the bidiagonalization procedure requires  $4m^3/3$  operations. The SVD costs  $20m^3/3$  operations because the right transformations must be accumulated. Interestingly enough, a straightforward SVD (i.e., not trying to preserve the zeros of  $R$ ) requires only  $6m^3$  operations [1]. The technique of Moré requires  $2m^3/3$  operations, but is free if we forgo the column pivoting.

Hence (3.3) simplifies to

$$(3.4) \quad \min \left\{ \left\| \begin{pmatrix} J \\ \sqrt{\lambda} I \end{pmatrix} y - \begin{pmatrix} \hat{b} \\ 0 \end{pmatrix} \right\| \right\},$$

where  $y = Y^T x$  and  $\hat{b} = W^T Q^T b_j$ . An orthogonal matrix  $Q_\lambda$  is determined such that

$$Q_\lambda^T \begin{pmatrix} J & \hat{b} \\ \sqrt{\lambda} I & 0 \end{pmatrix} = \begin{pmatrix} J_\lambda & z_1 \\ 0 & z_2 \end{pmatrix},$$

where  $J_\lambda$  has the same form as  $J$ . The solution  $y$  is given by

$$J_\lambda y = z_1.$$

This procedure requires  $16m$  operations and  $2m$  square roots (respectively  $m$  operations) if  $J$  is bidiagonal (respectively diagonal) [4]. For an upper triangular  $J$  a sequence of  $m(m+1)/2$  Givens rotations would be required, costing about  $2m^3/3$  operations and  $m^2/2$  square roots [12].

Our remaining task is to determine  $\lambda > 0$  such that the solution to (3.4) satisfies

$$\|y(\lambda)\| = 1.$$

Elden [4] discusses numerical methods for finding the positive zero of the function

$$\phi(\lambda) \equiv y(\lambda)^T y(\lambda) - 1.$$

Reinsch [14] suggests the function

$$\phi(\lambda) \equiv \frac{1}{y(\lambda)^T y(\lambda)} - 1,$$

which may have better convergence properties. The most effective algorithm for finding  $\lambda$  is given by Moré [12], who adopts a rational approximation approach [11]. He considers

$$\phi(\alpha) \equiv \|y(\alpha)\| - 1,$$

and accepts  $\alpha > 0$  if

$$(3.5) \quad |\phi(\alpha)| \leq \sigma,$$

where  $\sigma$  is some error bound. His iterative scheme is

$$(3.6) \quad \alpha_{k+1} = \alpha_k - (\phi(\alpha_k) + 1) \frac{\phi(\alpha_k)}{\phi'(\alpha_k)},$$

where

$$\phi'(\alpha) = -\|y(\alpha)\| \left\| J_{\alpha}^{-T} \left( \frac{y(\alpha)}{\|y(\alpha)\|} \right) \right\|^2.$$

The algorithm is safeguarded by the use of upper and lower bounds  $u_k$  and  $l_k$ . The procedure (3.6) is modified whenever  $\alpha_{k+1}$  is outside of  $(l_{k+1}, u_{k+1})$ . The initial values are

$$u_0 = \|J^T \hat{b}\|, \quad l_0 = -\frac{\phi(0)}{\phi'(0)}, \quad \alpha_0 = \max \{0.001 u_0, (l_0 u_0)^{1/2}\};$$

the updating formulas are given in [12]. Moré remarks that his algorithm always converges, the rate is ultimately quadratic and less than two iterations (on the average) are required to find an  $\alpha$  when  $\sigma = 10^{-1}$ . A computer program is available in MINPACK [13]. Our numerical experience, reported in § 5, confirmed the claims in [12]; an average of at most 5 iterations is required when  $\sigma = 10^{-8}$ . Suppose  $\psi$  iterations are required. Then the operation count for the bidiagonalization scheme [4] is

$$4m^3/3 + [20m + 2m \text{ square roots}](\psi + 1),$$

for the SVD scheme [7] it is

$$6m^3 + 4m(\psi + 1),$$

and for the QR-factorization scheme [12] it is

$$[2m^3/3 + m^2/2 \text{ square roots}](\psi + 1).$$

It is clear that for  $\psi \geq 1$ , we should choose the bidiagonalization procedure (cf. LINPACK [3]).

We propose the following procedure:

#### ALGORITHM NEW MINRES

##### *Initialization phase.*

Compute an initial matrix  $A$  using the same technique as in Algorithm minres;

Determine a QR-factorization of  $A_{-1}$ ;

##### *Iterative phase.*

repeat until convergence

for  $j = 1, \dots, n$  do

begin

Use the updating technique [2] to get a QR-factorization of  $A_{-j}$ :

$A_{-j} = QR$ ; {skip this step when loop is entered for the first time}

Solve  $Ru = Q^T(b_j - A_{-j}a_j)$ ;

if  $\|a_j + u\| \leq 1$  then

Add  $u^T$  to the  $j$ th row of  $A$

else

begin

Bidiagonalize  $R$ :  $R = WJY^T$ ;

Let  $\hat{b} = W^T Q^T b_j$ ;  
 repeat until convergence {find  $\alpha$  s.t.  $1 - \sigma \leq \|y(\alpha)\| \leq 1 + \sigma$ }  
 begin  
 Use the procedure of Moré [12] to determine  $\alpha$  and its bounds;  
 Find an orthogonal matrix  $Q_\alpha$  such that

$$Q_\alpha^T \begin{pmatrix} J & \hat{b} \\ \sqrt{\alpha} I & 0 \end{pmatrix} = \begin{pmatrix} J_\alpha & z_1 \\ 0 & z_2 \end{pmatrix} \text{ and } J_\alpha \text{ is bidiagonal};$$

Solve  $J_\alpha y(\alpha) = z_1$   
 end;  
 Set  $x = Yy$ ;  
 Replace the  $j$ th row of  $A$  by  $x^T$   
 end  
 end.

**4. Convergence.** Let us discuss the convergence properties of the minres method. Denote the current factor matrix by  $A^{(l)} \equiv (a_{ij}^{(l)})$  and suppose we are at the  $j$ th step of a major iteration cycle. Recall the objective function

$$(2.3) \quad f(A^{(l)}(x)) \equiv \sum_{\substack{i=1 \\ i \neq j}}^n \sum_{\substack{k=1 \\ k \neq i, j}}^n \left( c_{ik} - \sum_{p=1}^m a_{ip}^{(l)} a_{kp}^{(l)} \right)^2 + 2f_j(x) \geq 0.$$

We determine a new  $j$ th row ( $=x^T$ ) of  $A^{(l)}$  so as to minimize the term  $2f_j(x)$  (the first term is not affected). Let  $u^{(l)} = x - a_j^{(l)}$ . We can show that

$$f(A^{(l)}) - f(A^{(l+1)}) = 2\|A_{-j}^{(l)} u^{(l)}\|^2 + 4\lambda^{(l)}(x^T x - a_j^{(l)T} x),$$

where  $\lambda^{(l)}$  is the Lagrange parameter of (2.5). Since either  $\lambda^{(l)} = 0$ , or  $\lambda^{(l)} > 0$  and  $x^T x = 1$  (also,  $\|a_j^{(l)}\| \leq 1$ ), we get

$$f(A^{(l)}) - f(A^{(l+1)}) \geq 2\|A_{-j}^{(l)} u^{(l)}\|^2 \geq 2(\sigma_{\min}^{(l)} \|u^{(l)}\|)^2,$$

where  $\sigma_{\min}^{(l)}$  denotes the smallest singular value of  $A^{(l)}$ . Note that  $\sigma_{\min}^{(l)}$  is positive by the assumption in §§ 2 and 3 that  $A_{-j}^{(l)}$  has full column rank. Suppose there exists a constant  $\rho > 0$  such that  $\sigma_{\min}^{(l)} \geq \rho$  for all  $l$ . Since

$$A^{(l+1)} = A^{(l)} + e_j u^{(l)T},$$

we have

$$(4.1) \quad f(A^{(l)}) - f(A^{(l+1)}) < \varepsilon \Rightarrow \|A^{(l)} - A^{(l+1)}\|_F^2 < \frac{\varepsilon}{2\rho^2},$$

for any given  $\varepsilon > 0$ .

Now, the sequence  $\{f(A^{(l)})\}$  is nonincreasing and thus has a limit. As  $A^{(l)}$  is contained in the compact set  $\{A: \|A\|_F^2 \leq n\}$ , there is a matrix  $\hat{A} \in \mathbf{R}^{n \times m}$  that is the limit of a convergent subsequence of  $\{A^{(l)}\}$ , and

$$f(\hat{A}) \leq f(A^{(l)}), \quad l = 0, 1, \dots$$

There exists a row index  $j$  and a subsequence  $\{A^{(l_j)}\}$  of the convergent subsequence above so that each term, say  $M$ , of  $\{A^{(l_j)}\}$  solves

$$(2.4) \quad \min \{\|b_j - M_{-j} x\| : \|x\| \leq 1\}.$$

It follows that  $\hat{A}$  satisfies

$$(4.2) \quad (\hat{A}_{-j}^T \hat{A}_{-j} + \hat{\lambda}_j I) \hat{a}_j = \hat{A}_{-j}^T b_j,$$



where  $\hat{\lambda}_j \geq 0$  and  $\hat{a}_j^T$  denotes the  $j$ th row of  $\hat{A}$ . We want to prove that  $\hat{A}$  satisfies (4.2) for  $j = 1, 2, \dots, n$ . Consider the subsequence  $\{A^{(j+1)}\}$ . A subsequence of this subsequence will converge to a limit, say  $\bar{A}$ . However,  $\bar{A}$  will satisfy

$$(4.3) \quad (\bar{A}_{-j-1}^T \bar{A}_{-j-1} + \bar{\lambda}_{j+1} I) \bar{a}_{j+1} = \bar{A}_{-j-1}^T b_{j+1},$$

where  $\bar{\lambda}_{j+1} \geq 0$  and  $\bar{a}_{j+1}^T$  denotes the  $(j+1)$ st row of  $\bar{A}$ . Note that, to be precise, we should have written  $(j+1) \bmod n$  instead of  $(j+1)$ . Since  $f(\hat{A}) = f(\bar{A})$  we get from (4.1) that  $\hat{A} = \bar{A}$ . Hence (4.3) is also valid for  $\hat{A}$  and the proof is complete.

Yet the matrix  $\hat{A}$  need not minimize the function (1.1). Let

$$C = \begin{pmatrix} 1 & -.01 & .01 \\ -.01 & 1 & .01 \\ .01 & .01 & 1 \end{pmatrix}$$

and  $m = 1$ . Then  $\hat{A}$  can be any one of the three vectors:

$$\begin{pmatrix} -.1 \\ .1 \\ 0 \end{pmatrix}, \begin{pmatrix} .01 \\ .01 \\ 1 \end{pmatrix} \text{ or } \begin{pmatrix} -.01 \\ 1 \\ .01 \end{pmatrix},$$

but only the last two minimize (1.1). The minres method would converge to either the second or the third choice if the initial matrix were computed using principal components ( $C$  has a multiple eigenvalue at 1.01).

It is well known [8, pp. 27-28] that the common-factor loadings are not unique. For example,

$$\hat{A}\hat{A}^T = (\hat{A}Q)(\hat{A}Q)^T$$

for any  $m \times m$  orthogonal matrix  $Q$ . A canonical form is usually selected for the accepted factor matrix, say  $\tilde{A}$ , so that successive factors will account for the maximum possible variance [8, pp. 164-166]: let

$$\tilde{A} = U\Sigma V^T$$

be an SVD; The canonical form for  $\tilde{A}$  equals  $U\Sigma$ .

**5. Numerical results.** The following examples have been chosen:

1. Five hypothetical variables [9],
2. Five socio-economic variables [8],
3. Eight physical variables [10],
4. Eight emotional traits [8],
5. Twelve naval recruits temperament scales [8],
6. Twenty box measurements [8], and
7. Twenty-four psychological tests [8].

We have extended the convergence test (2.8) of Harman et al. to include function values. At the end of an iteration cycle the current factor matrix  $A^C$  would be accepted if either

$$\|A^- - A^C\|_F^2 < \tau_A * \max \{1, \|A^C\|_F^2\}$$

or

$$f(A^-) - f(A^C) < \tau_f * \max \{1, f(A^C)\},$$

where  $A^-$  denotes the matrix at the end of the previous iteration cycle.

A computer program implementing the minres method was written in FORTRAN. The preceding examples were run on a VAX-11/780 at Cornell University using double floating data types: each number is binary normalized, with an 8-bit signed exponent and a 57-bit signed fraction whose most significant bit is not represented. This is equivalent to a precision of approximately 17 decimal digits. The tolerances were set at  $\tau_A = \tau_f = 10^{-8}$ . We observed that

$$\|A^- - A^C\|_F^2 \approx f(A^-) - f(A^C)$$

when  $A^C$  approached  $\hat{A}$ . The parameter NC stands for the number of major cycles required for convergence, NZ for the number of least squares solutions with  $\lambda = 0$  and NP for the number of solutions with  $\lambda > 0$ . Note that

$$n * NC = NZ + NP.$$

The parameter NM represents the number of iterations required by the algorithm of Moré. The criterion  $\sigma$  in (3.5) was chosen as  $10^{-8}$ , i.e., equal to  $\tau_A$  and  $\tau_f$ . The root-mean-square deviation [8, p. 177]

$$\text{rms} = \left[ \frac{f(\hat{A})}{n(n-1)} \right]^{1/2}$$

measures how well the model fits the observed data. The results are shown in Table 1. The rate of convergence was linear. We used larger tolerances for Problem 4\*:  $\tau_A = \tau_f = 10^{-6}$ , but  $\sigma$  still equaled  $10^{-8}$ .

TABLE 1

Problem	$n$	$m$	NC	NZ	NP (NM)	rms
1	5	1	6	25	5 (0)	.0210
2	5	2	7	30	5 (3)	.0098
3	8	2	7	56	0	.0207
3	8	3	35	278	2 (5)	.0127
4	8	2	10	80	0	.0515
4	8	3	15	94	26 (4.96)	.0305
4	8	4	>100			
4*	8	4	20	114	46 (4.22)	.0222
5	12	4	36	432	0	.0271
6	20	3	4	71	9 (4.22)	.0061
7	24	4	7	168	0	.0408
7	24	5	27	644	4 (4.75)	.0370
7	24	6	39	933	3 (5)	.0318

**Acknowledgments.** The author owes J. J. Moré his sincerest gratitude for the expert advice and uncommon patience. He also would like to thank M. J. Todd and the two referees for their many valuable suggestions.

## REFERENCES

- [1] T. F. CHAN, *An improved algorithm for computing the singular value decomposition*, ACM Trans. Math. Software, 8 (1982), pp. 72-83.
- [2] J. W. DANIEL, W. B. GRAGG, L. KAUFMAN AND G. W. STEWART, *Reorthogonalization and stable algorithms for updating the Gram-Schmidt QR factorization*, Math. Comput., 30 (1976), pp. 772-795.
- [3] J. J. DONGARRA, J. R. BUNCH, C. B. MOLER AND G. W. STEWART, *LINPACK Users' Guide*, Society for Industrial and Applied Mathematics, Philadelphia, 1979.

- [4] L. ELDEN, *Algorithms for the regularization of ill-conditioned least squares problems*, BIT, 17 (1977), pp. 134–145.
- [5] W. GANDER, *On the linear least squares problem with a quadratic constraint*, Report STAN-CS-78-697, Computer Science Dept., Stanford Univ., Stanford, CA, 1978.
- [6] G. H. GOLUB, *Numerical methods for solving linear least squares problems*, Numer. Math., 7 (1965), pp. 206–216.
- [7] ———, *Some modified matrix eigenvalue problems*, SIAM Rev., 15 (1973), pp. 318–334.
- [8] H. H. HARMAN, *Modern Factor Analysis*, 3rd ed., Univ. Chicago Press, Chicago, 1976.
- [9] H. H. HARMAN AND Y. FUKUDA, *Resolution of the Heywood case in the minres solution*, Psychometrika, 31 (1966), pp. 563–571.
- [10] H. H. HARMAN AND W. H. JONES, *Factor analysis by minimizing residuals (minres)*, Psychometrika, 31 (1966), pp. 351–368.
- [11] M. D. HEBDEN, *An algorithm for minimization using exact second derivatives*, Atomic Energy Research Establishment report R6799, Harwell, England, 1973.
- [12] J. J. MORÉ, *The Levenberg-Marquardt algorithm: implementation and theory*, in Lecture Notes in Mathematics 630: Numerical Analysis, Springer-Verlag, Berlin, 1978, pp. 105–116.
- [13] J. J. MORÉ, B. S. GARBOW AND K. E. HILLSTROM, *User Guide for MINPACK-1*, Report ANL-80-74, Argonne National Laboratory, Argonne, IL, 1980.
- [14] C. H. REINSCH, *Smoothing by spline functions. II*, Numer. Math., 16 (1971), pp. 451–454.
- [15] L. L. THURSTONE, *The Vectors of Mind*, Univ. Chicago Press, Chicago, 1935.
- [16] ———, *Multiple Factor Analysis*, Univ. Chicago Press, Chicago, 1947.

## A BRANCH AND BOUND PROCEDURE FOR SELECTION OF VARIABLES IN MINIMAX REGRESSION\*

SUBHASH C. NARULA† AND JOHN F. WELLINGTON‡

**Abstract.** An efficient branch and bound algorithm is proposed for the selection of variables in a multiple linear regression model using the minimization of the maximum weighted absolute error criterion. Besides using a special preoptimality test, fathoming test and choice rule, the proposed algorithm takes advantage of the special features of the new linear programming formulation of the problem. A few suggestions are also given to accelerate the proposed algorithm. The steps of the algorithm are illustrated with an example.

**Key words.** absolute error, implicit enumeration,  $L_\infty$ -norm, subset selection

**1. Introduction.** The least squares (or equivalently, the minimization of the sum of squared errors, MSSE) regression has dominated the statistical literature for a long time. This dominance and popularity of the least squares regression can be ascribed, at least partially, to the fact that the theory is simple, well developed and well documented. The computer packages for least squares regression are easily available. The least squares regression is optimal and results in the maximum likelihood estimators of the unknown parameters of the model if the errors are independent and follow a normal distribution with mean zero and a common (though unknown) variance  $\sigma^2$ . However, the least squares regression is far from optimal in many non-Gaussian situations.

In many practical problems, the errors may not follow a normal distribution, as pointed out by Box (1967) and Cox (1967) in the discussion of Anscombe's (1967) paper.

G. E. P. Box (1967) stated, "I feel certain that platykurtic (*short-tailed*) distributions do occur in practice for one reason because of deliberate or unconscious truncations and these ought not to be ruled out."

On examining several sets of data, D. R. Cox (1967) observed that, "the surprising conclusion was that while there were frequent departures from normality these were about equally often towards long-tailed and towards short-tailed distributions."

The results of Monte Carlo studies of Bourdon (1974) and Forth (1974) on the relative performance of the minimum sum of absolute errors, MSAE, the MSSE and the minimization of the maximum absolute error, MMAE, criteria for various symmetric (error) distributions indicate that if the error distribution is known to be long-tailed, medium-tailed or short-tailed, the MSAE, the MSSE or the MMAE regression, respectively, performs best. In this paper we shall discuss the MMAE criterion, which is also known as the minimax criterion, Chebyshev criterion and the  $L_\infty$ -norm. We refer the reader to Daniel and Wood (1980), Draper and Smith (1981) and Montgomery and Peck (1982) for the MSSE regression and Narula and Wellington (1982) for the MSAE regression.

The MMAE estimators are maximum likelihood estimators of the parameters if the random errors are independent and follow a uniform distribution with a wide range. Rice and White (1964) have recommended the MMAE criterion in experimental situations where the observed response is subject to errors that can be described by probability distributions having sharply defined extremes with high probability. An

---

\* Received by the editors June 2, 1982, and in revised form February 13, 1984.

† Virginia Commonwealth University, Richmond, Virginia 23284.

‡ Gannon University, Erie, Pennsylvania 16541.

example of such a situation occurs when the observed values are truncated to some specified number of decimal places. Stiefel (1960) and Barrodale and Young (1966) used the criterion for the computation of polynomial approximations to functions. Barrodale and Phillips (1974) found the criterion useful when the response variable contains only small inherent errors. It may be pointed out that the MMAE criterion is very sensitive to outliers and as such should be used only when the underlying assumptions are satisfied. For example, Appa and Smith (1973) caution against the use of the MMAE criterion in certain econometric studies (especially those where error variance may be nonfinite).

No matter what criterion is used to estimate the parameters of a multiple linear regression model, many times the initial model contains a large number of predictor (regressor) variables. These variables, one hopes, include all the relevant variables and their appropriate functions, but at times also include some extraneous variables and their functions. Thus it is possible that a model with only a few variables may adequately explain the data or predict the response. A model with only a few variables may also be desirable for economic and practical reasons since it is less expensive to collect data and thus maintain a model (with fewer variables). It has also been shown, Narula and Ramberg (1972) and Walls and Weeks (1969), that a model with fewer variables may have more desirable statistical properties. Further, models with fewer variables may also alleviate certain computational difficulties.

Our objective in this paper is to propose an efficient algorithm for selection of variables in a multiple linear regression model using the minimization of the maximum weighted absolute error, MMWAE, criterion. Note that the MMWAE criterion subsumes the MMAE criterion. Recently Armstrong and Beck (1983) proposed a procedure for subset selection using the MMAE criterion. However, the algorithm proposed in this paper differs from their procedure in a number of minor and major aspects. The most significant differences which make the proposed algorithm more attractive are that the proposed algorithm uses (a) a new formulation (see formulation F or F' in § 2) for the MMAE regression which is more suitable for the selection of variable problems than the original formulation of Wagner (1959); (b) a biologically constructed tree (see § 3.2) that requires less bookkeeping and makes it easier to understand the branch and bound procedure; and (c) any bounded variable algorithm may be used to solve individual problems. However, to improve the efficiency of the algorithm, it is recommended that one use a modified bounded variable algorithm that incorporates various features of the Barrodale and Phillips (1974) algorithm, which takes advantage of the special structure of the problem formulation. It may be pointed out that the Armstrong and Beck (1983) algorithm can be easily modified to select variables using the MMWAE criterion. For selection of variables using the MSSE and the MSAE criteria, the reader may refer to Hocking (1976) and Narula and Wellington (1979), respectively.

The rest of the paper is organized as follows: In § 2, we give a new linear programming formulation of the problem for estimation of the unknown parameters in a multiple linear regression model using the MMWAE criterion. We also state the problem of selecting variables using the MMWAE criterion. In § 3, we state several features of the problem and describe a branch and bound algorithm to find the "best" models with fewer variables. We illustrate the proposed algorithm in § 4 and conclude the paper with a few remarks in § 5.

**2. Problem formulation.** Let  $\mathbf{y}$  be a  $n \times 1$  vector of a response variable corresponding to  $\mathbf{X}$ , a  $n \times k$  matrix of predictor variables. Then, the multiple linear regression

model can be stated as

$$\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\epsilon},$$

where  $\boldsymbol{\beta}$  is a  $k \times 1$  vector of unknown parameters and  $\boldsymbol{\epsilon}$  a  $n \times 1$  vector of random errors. For  $w_i > 0$ , the MMWAE estimate  $\mathbf{b}$  of  $\boldsymbol{\beta}$

$$\underset{\mathbf{b}}{\text{minimizes}} \quad \underset{i=1, \dots, n}{\text{maximum}} \quad |y_i - \mathbf{x}'_i \mathbf{b}| w_i,$$

where  $\mathbf{x}'_i$  is the  $i$ th row of  $\mathbf{X}$ . For  $w_i = 1$ ,  $i = 1, \dots, n$ ,  $\mathbf{b}$  is an MMAE estimator of  $\boldsymbol{\beta}$ .

Let  $e = \underset{i=1, \dots, n}{\text{maximum}} |y_i - \mathbf{x}'_i \mathbf{b}| w_i$ . Then the preceding formulation can be written as:

$$\begin{aligned} &\text{minimize } e, \\ &\text{subject to } -e\mathbf{1} \leq \mathbf{W}(\mathbf{y} - \mathbf{X}\mathbf{b}) \leq e\mathbf{1}, \quad \mathbf{b} \text{ unrestricted in sign,} \end{aligned}$$

where  $\mathbf{1}$  is an  $n \times 1$  vector of ones and  $\mathbf{W}$  is an  $n \times n$  diagonal matrix of  $(w_1, w_2, \dots, w_n)$ .

However, by defining  $e^* = 1/e$ , we can restate the problem as:

$$(F) \quad \begin{aligned} &\text{maximize } e^*, \\ &\text{subject to } \mathbf{W}[\mathbf{y} - \mathbf{X}\mathbf{b}]e^* + \mathbf{S} = \mathbf{1}, \quad \mathbf{0} \leq \mathbf{S} \leq 2\mathbf{1}, \end{aligned}$$

where  $\mathbf{S}$  is an  $n \times 1$  vector of slack variables and  $\mathbf{0}$  an  $n \times 1$  vector of zeros.

The selection of variables in relation to the MMWAE criterion can now be stated as: for each  $p$ ,  $p = 1, \dots, k-1$ , (i) find the subset of variables, from among  $\binom{k}{p}$  possible subsets, associated with the smallest maximum weighted absolute error; and (ii) for the set of  $p$  variables found in (i), compute the estimates of the corresponding  $\boldsymbol{\beta}$ 's. Thus, the problem of the selection of variables can be viewed as a solution to a series of problems of the form (F). More specifically, let  $J_p = \{i_1, i_2, \dots, i_p\}$  be a set of  $p$  indices from  $I = \{1, 2, \dots, k\}$ , and  $\bar{J}_p$  be the complement of  $J_p$ . Further, let

$$\mathbf{X} = [\mathbf{X}_{J_p} \quad \mathbf{X}_{\bar{J}_p}], \quad \boldsymbol{\beta} = \{\boldsymbol{\beta}_{J_p} \quad \boldsymbol{\beta}_{\bar{J}_p}\}' \quad \mathbf{b} = [\mathbf{b}_{J_p} \quad \mathbf{b}_{\bar{J}_p}]'$$

Then for a given  $J_p$ , the problem is:

$$(F') \quad \begin{aligned} &\text{maximize } e^* \\ &\text{subject to } \mathbf{W}[\mathbf{y} - \mathbf{X}_{J_p} \mathbf{b}_{J_p}]e^* + \mathbf{S} = \mathbf{1}, \quad \mathbf{0} \leq \mathbf{S} \leq 2\mathbf{1}. \end{aligned}$$

We have chosen this formulation rather than the modification of the primal and the dual formulation of the MMAE criterion as given by Wagner (1959) for the following reasons. When solving the MMWAE problem for the model with  $k$  variables only, the dual formulation of Wagner (1959) can be solved efficiently by an algorithm of Barrodale and Phillips (1974); however, we need to solve a series of problems. This can be accomplished more efficiently and easily by the deletion (addition) of columns rather than rows. Further, we can use modified bounded variable algorithms that incorporate special features of the Barrodale and Phillips (1974) algorithm.

### 3. Solution procedure.

**3.1. Some observations.** Branch and bound algorithms achieve computational efficiency by (i) reducing the computations required to examine a given subset of variables, and (ii) selecting the best subset of a given size without examining all the subsets. To achieve these objectives, we make the following observations.

1. Let  $e_p$  be the least maximum weighted absolute error associated with the "best" set of  $p$  variables,  $p = 1, 2, \dots, k$ . Then

$$e_k \leq e_{k-1} \leq \dots \leq e_2 \leq e_1.$$

2. The least maximum weighted absolute error for a set of  $p$  variables is less than or equal to the least maximum weighted absolute error for a set of  $t(\leq p)$  variables that can be obtained by deleting variable(s) from the  $p$  variables. Thus a *fathoming rule* can be stated as:

If the least maximum weighted absolute error associated with a set of  $p$  variables currently under investigation exceeds the least maximum weighted absolute error for the best set of  $t(\leq p)$ , then models that can be obtained from the set of  $p$  variables need not be examined.

3. Since formulation F (or F') is a maximization problem, the objective function value  $e^*$  is nondecreasing. Thus we can state: a *pre-optimality test* as:

If the objective function value (at any iteration) for the set of  $p$  variables under investigation exceeds the "best" objective function value associated with a set of  $p$  variables found so far, we need not investigate the set of  $p$  variables under investigation any further.

4. The feasibility of F' is lost (maintained) when one or more variables are added (deleted) to (from) a set of variables. When the problem is dual feasible, the pre-optimality test and the fathoming test can be applied *before* each iteration.

5. Since the problem for a given  $J_P$  can be obtained by adding or deleting columns in F', we can start the solution procedure from a previously solved problem. Computational savings can be achieved by generating the problem in certain sequences.

6. In branch and bound procedures it is advantageous to find strong bounds early in the solution process since it helps in fathoming. Thus, given a choice of variables for deletion, it may help to select the variable that results in the smallest increase in the maximum (weighted) absolute error (or equivalently, the largest increase in  $e^*$ ). Thus a *choice rule* may be stated as

When a choice exists for removing a variable from a model, choose the variable that results in the maximum increase in  $e^*$ .

**3.2. A tree.** We describe the construction of a tree and movement through the tree that takes advantage of the preceding observations. However, before doing so, we establish the following terminology. If a set of  $p-1$  variables is obtained from a set of  $p$  variables by deleting one variable, then the set of  $p$  variables is called the father (mother), and the set of  $p-1$  variables, a son (daughter). All the subsets of  $p-1$  variables obtained from the same set of  $p$  variables are called brothers (sisters).

The root of the tree represents the model with all  $k$  variables, and the interior nodes represent models with fewer variables. The interior nodes are constructed in a "biologically feasible" order, that is, with the restriction that "a parent be born before a child." Such a tree can be constructed by obtaining the interior nodes by deleting from the left (or right) one predictor variable at a time. One such tree for a four variable problem is given in Fig. 1. The indices of the variables included in the model, at each node, are also shown in Fig. 1.

The movement through the tree that takes advantage of the special features of the problem can be stated as follows.

Start at the root of the tree and move from a parent to a child. At a terminal node (a node with no children), move to the next younger brother (sister), or if there is no brother (sister), to the father's (mother's) next younger brother (sister), or if there is

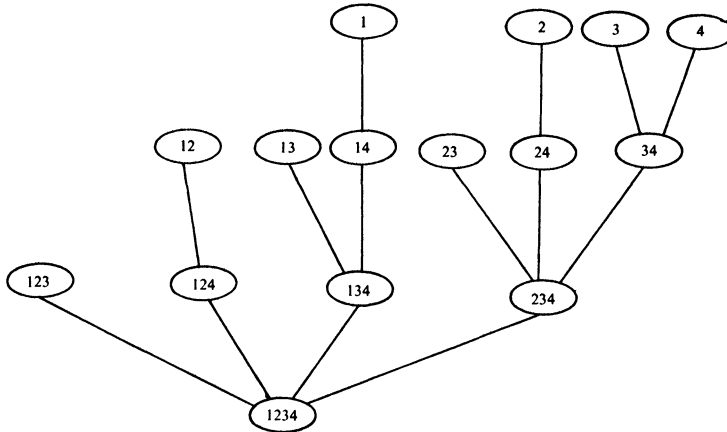


FIG. 1. A complete tree for a four variable problem.

no remaining uncle (aunt), to the grandfather's (grandmother's) next younger brother (sister), and so on. For the tree in Fig. 1, one possible movement is 1234, 234, 34, 4, 3, 24, 2, 23, 134, 14, 1, 13, 124, 12 and 123.

Any labelling procedure can be used for movement through the tree. However, for such problems, an efficient labelling procedure appears in Narula and Wellington (1979).

**3.3. Algorithm.** An algorithm for selecting the best model with  $p$  variables,  $p = 1, 2, \dots, k$ , using the MMWAE criterion, can be described as follows. Let  $p$  denote the number of variables included in the model under investigation.

*Step 0.* Obtain initial bounds  $e_p^0$ ,  $p = 1, \dots, k$ . Go to Step 1.

*Step 1.* Set up the linear programming problem of the form  $F'$  for the model under investigation. If the current solution is dual feasible, go to Step 3; if it is primal feasible, go to Step 4; if it is neither primal nor dual feasible, go to Step 2.

*Step 2.* Obtain dual feasibility. Go to Step 3.

*Step 3.* Use the dual simplex method for bounded variables to solve the problem. Solve the problem to optimality. Apply the fathoming test. If successful go to Step 7; otherwise go to Step 5.

*Step 4.* Use the primal simplex method for bounded variables to solve the problem. Proceed as follows:

(a) Before each iteration, apply the fathoming test. If successful go to Step 7; otherwise apply the preoptimality test. If successful go to Step 6; otherwise go to (b).

(b) Perform the simplex iteration. If solution is optimal go to Step 5; otherwise go to (a).

*Step 5.* If the current objective function value is smaller than the "best" found so far for a model with  $p$  variables, store the current solution as the best model of  $p$  variables. Go to Step 6.

*Step 6.* If possible, take a forward step and go to Step 1; otherwise go to Step 7.

*Step 7.* If possible take a backward step and go to Step 6; otherwise problem is solved and Stop.

**3.4. Some improvements.** The proposed algorithm can be made more efficient by including one or more of the following features in the procedure.



(i) Set a lower limit or an upper limit or both on the number of variables to be included in the models to be searched.

(ii) Find a model with  $m(1, 2, \dots, k-1)$  variables that does not differ in its minimax value from the best model with  $m$  variables by more than 100  $\delta$  percent,  $\delta \geq 0$ .

(iii) Find only those models which do not differ in their minimax value from the model with  $k$  variables (i.e., the full model) by more than 100  $\lambda$  percent,  $\lambda \geq 0$ .

The way to implement any of these features in the branch and bound procedures is described in detail by Narula and Wellington (1983). However, it should be clearly understood that if (ii) or (iii) or both are included in the procedure, the algorithm does *not* have to find the best model with  $m (=1, \dots, k)$  variables.

**4. An illustrative example.** We illustrate the proposed algorithm with a data set from Wood (1973). It consists of 82 observations on a response variable and four predictor variables. To demonstrate the features of the proposed algorithm, the MMAE values associated with all possible models are given in Fig. 2.

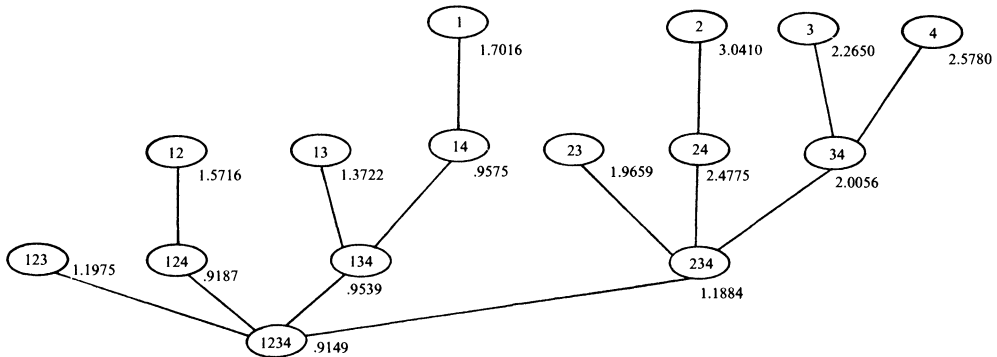


FIG 2. The MMAE value associated with all possible models for Wood (1973) data.

To obtain the initial bounds, the least squares models of sizes 1, 2 and 3 were found by forward selection procedure. The maximum absolute error corresponding to these models provides the initial bounds (Step 0). These bounds are given in Table 1.

Formulation F is solved for all the four variables in the model. The least maximum absolute error for the model is .9149. A forward step results in a model with variables 234. The problem is dual feasible and the fathoming test is successful. Thus, we need not investigate models that can be obtained from the model.

Next, a backward step followed by a forward step results in a model with variables 134 with least maximum absolute error of .9539. This value now becomes the new bound for a model with three variables. However, at this stage the branch cannot be

TABLE 1  
Initial bounds for models of various sizes for Wood (1973) data.

No. of variables in the model	Predictor variables	Max. absolute value
1	1	1.7016
2	12	1.5716
3	123	1.1975
4	1234	0.9149

fathomed. A model with variables 14 is obtained on a forward step by deleting variable 3, and the problem remains dual feasible. The problem had to be solved to optimality, resulting in a least maximum absolute error of .9575, which becomes the bound for a model with two variables. A model with variable 1 is obtained on the next forward step by deleting variable 4. The least maximum absolute error for the model is 1.7016, a new bound for models with one variable. We reach a model with variables 13 by taking two backward steps followed by a forward step. However, at this stage fathoming is successful prior to solving the problem to optimality.

Next we take two backward steps followed by a forward step to arrive at a model with variables 124. We examine this model by deleting variable 3 from the optimal tableau for the full model. We refer to this as restarting the procedure. The problem is dual feasible and results in a least maximum absolute error of .9178. Thus this value becomes a new bound for models with three variables. On the next forward step we reach a model with variables 12 that has a least maximum absolute error of 1.5716.

Now we take a backward step followed by a forward step arriving at the model with variables 123. The fathoming test is successfully applied prior to solving the problem to optimality. On the next backward step we reach the root of the tree. The algorithm stops.

These steps for the example appear in Fig. 3 and Table 2. Table 3 gives the best MMAE models and their corresponding MMAE values for  $p = 1, 2, 3, 4$ .

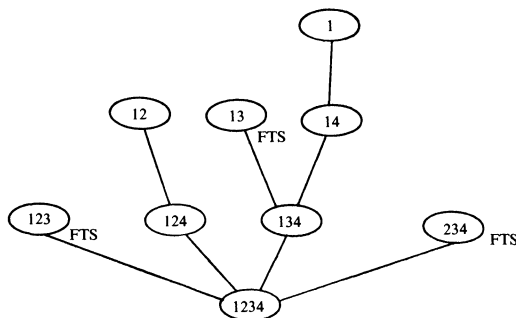


FIG. 3. The solution tree for the problem.

**5. Some remarks.** The MMAE criterion for the estimation of parameters in a multiple linear regression model should be used with care as it is very sensitive to outliers. However, if all the underlying assumptions are satisfied and it is considered appropriate to use the criterion, then one may use the algorithm proposed in this paper for the selection of variables in the multiple linear regression model using the MMAE criterion. The use of the new formulation affords the maximum exploitation of the special structure of the problem in the variable selection algorithm. Special algorithms (in Steps 3 and 4 of the algorithm) will reduce the computational effort. However, if such algorithms are not available, the proposed procedure can be used with any algorithm to solve a linear programming problem. No restriction, such as that every submatrix of  $X$  be of full rank, are placed on the problem. A suggestion of Narula and Wellington (1983) to accelerate the variable selection algorithms can be used with the proposed procedure. One such program has been developed by Wellington and Narula (1981b).

The working array required to solve F is of the order  $n \times k$ . Our limited computational experience with the proposed procedure has been similar to the results reported

TABLE 2  
*Summary of solution procedure for Wood (1973) data.*

Variables in the model	MMAE value	Remarks
1234	.9149	
234	1.1884	FTS
134	.9539	Store results as the current "best" model with three variables.
14	.9575	Store results as the current "best" model with two variables.
1	1.7016	Store results as the current "best" model with one variable.
13	—	FTS prior to optimality.
124	.9187	Store results as the current "best" model with three variables.
12	1.5716	This is an inferior model with two variables compared to the current "best."
123	—	FTS prior to optimality. Search terminated.

FTS: fathoming test successful

TABLE 3  
*Best models of various sizes for Wood (1973) data.*

No. of variables in the model	Predictor variables	MMAE value
1	1	1.7016
2	14	0.9575
3	124	0.9187
4	1234	0.9149

by Wellington and Narula (1981a) for the minimum sum of weighted absolute errors criterion.

**Acknowledgment.** We thank Dr. J. H. Friedman for his suggestions which have improved the presentation of the paper.

#### REFERENCES

- F. J. ANSCOMBE (1967) *Topics in the investigation of linear relations fitted by the method of least squares*, J. Royal Statist. Soc. Ser. B29, pp. 1-52.
- G. APPA AND C. SMITH (1973), *On  $L_1$  and Chebyshev estimation*, Math. Programming, 5, pp. 73-87.
- R. D. ARMSTRONG AND P. O. BECK (1983), *The best parameter subset using the Chebyshev curve fitting criterion*, Math. Programming, 27, pp. 64-74.
- I. BARRODALE AND C. PHILLIPS (1974), *An improved algorithm for discrete Chebyshev linear approximation*, Proc. 4th Manitoba Conference on Numerical Mathematics, Univ. Manitoba, Winnipeg, Manitoba, pp. 177-190.
- I. BARRODALE AND A. YOUNG (1966), *Algorithm for best  $L_1$  and  $L_\infty$  linear approximation on a discrete set*, Numer. Math., 8, pp. 295-306.
- G. A. BOURDON (1974), *A Monte Carlo sampling study for further testing of the robust regression procedures based upon the kurtosis of the least squares residuals*, unpublished M.S. thesis, Air Force Institute of Technology, Wright-Patterson AFB, Ohio.
- G. E. P. BOX (1967), *Discussion*, J. Royal Statist. Soc. Ser. B29, pp. 42-43.
- D. R. COX (1967), *Discussion*, J. Royal Statist. Soc. Ser. B29, p. 29.

- C. DANIEL AND F. S. WOOD (1980), *Fitting Equations to Data*, 2nd edition, John Wiley, New York.
- N. R. DRAPER AND H. SMITH (1981), *Applied Regression Analysis*, 2nd edition, John Wiley, New York.
- C. R. FORTH (1974), *Robust estimation techniques for population parameters and regression coefficients*, unpublished M.S. thesis, Air Force Institute of Technology, Wright-Patterson AFB, Ohio.
- R. R. HOCKING (1976), *The analysis and selection of variables in linear regression*, *Biometrics*, 32, pp. 1-49.
- D. C. MONTGOMERY AND E. A. PECK (1982), *Introduction to Linear Regression Analysis*, John Wiley, New York.
- S. C. NARULA AND J. S. RAMBERG (1972), *Letter to the Editor*, *Amer. Statist.*, 26, pp. 42.
- S. C. NARULA AND J. F. WELLINGTON (1979), *Selection of variables in linear regression using the minimum sum of absolute error criterion*, *Technometrics*, 21, pp. 299-306.
- (1982), *The minimum sum of absolute regression: a state of the art survey*, *Internat. Statist. Rev.*, 50, pp. 317-326.
- (1983), *Selection of variables in linear regression: a pragmatic approach*, *J. Statist. Comput. Simul.*, 17, pp. 159-172.
- J. R. RICE AND J. S. WHITE (1964), *Norms for smoothing and estimation*, *SIAM Rev.*, 6, pp. 243-256.
- E. STIEFEL (1960), *Note on Jordan elimination, linear programming and Tchebycheff approximation*, *Numer. Math.*, 2, pp. 1-17.
- H. M. WAGNER (1959), *Linear programming techniques for regression analysis*, *J. Amer. Statist. Assoc.*, 54, pp. 206-212.
- R. E. WALLS AND D. L. WEEKS (1969), *A note on the variance of a predicted response in regression*, *Amer. Statist.*, 23, pp. 24-26.
- J. F. WELLINGTON AND S. C. NARULA (1981a), *Variable selection in multiple linear regression using the minimum sum of weighted absolute errors criterion*, *Comm. Statist.: Part B*, 10, pp. 641-648.
- (1981b), *Variable selection in multiple linear regression using the minimization of the weighted maximum absolute error criterion*, Research Report No. 37-81-P7, School of Management, Rensselaer Polytechnic Institute, Troy, NY.
- F. S. WOOD (1973), *The use of individual effects and residuals in fitting equations to data*, *Technometrics*, 15, pp. 677-695.

## A NUMERICAL MODEL OF TWO-DIMENSIONAL, TWO-COMPONENT, SINGLE PHASE MISCIBLE DISPLACEMENT IN A POROUS MEDIUM\*

THOM POTEPA†

**Abstract.** A new numerical procedure for modeling single phase miscible displacement in a porous medium is presented. This model is based upon a material balance that is similar to that which is used to derive the differential equations that govern single phase miscible displacement. Since the procedure is defined in terms of a material balance, the data structures arising in a computational implementation are compatible with those that are present in finite difference models of miscible displacement. This procedure obeys a maximum principle due to the upstream weighting of the convective transport terms. This new procedure does not exhibit the grid orientation effect present in the five-point finite difference models of this process.

**Key words.** numerical modeling, reservoir engineering, material balance

**1. Introduction.** Due to the growth in the computational power of the digital computer in recent years, numerical models of physical phenomena which cannot be described analytically have proved increasingly useful in engineering process design. In the field of petroleum reservoir engineering, numerical simulations of complex enhanced recovery mechanisms such as steamflooding [1], *in situ* combustion [2], chemical flooding [3], and miscible displacement by carbon dioxide [4] have been undertaken. These simulation efforts have the potential of determining the economic viability of a particular recovery mechanism without the need for an expensive and time-consuming pilot study.

A phenomenon which limits the usefulness of current reservoir simulators in field level simulations is the grid orientation effect. A numerical discretization procedure that is used in a physical simulator is said to exhibit the grid orientation effect if the approximate solution is sensitive to the spatial orientation of the grid. In the reservoir engineering literature, a spatial discretization that forms channels that are parallel to the streamlines connecting the injection and production wells is known as a parallel grid [5]. Similarly, a partition of the domain which forms channels that are diagonal to the streamlines connecting the injection and production wells is called a diagonal grid. Not only do the approximate solutions generated by the parallel and diagonal grids differ at discretization sizes practical for field simulation work, but the asymptotic solutions that are generated by the two grids differ considerably. Grid effects were first reported for a model problem by Hirasaki, O'Dell and Todd [5], and were later observed in the simulation of steamflooding by Chu, Coats, George, and Marcum [1]. Since the qualitative aspects of grid orientation effect are similar for both the model problems and problems of practical significance, the test problem generally used to determine the sensitivity of a numerical discretization procedure to the orientation of the grid is a variant of the single phase miscible displacement problem.

Many numerical discretization procedures have been proposed to deal with the grid effects. These procedures include finite difference techniques, finite element techniques, and procedures based upon the method of characteristics. Each class of procedures is discussed in greater detail below.

Five-point finite difference procedures are widely used in production reservoir simulators today. Procedures which utilize either an expanded or a modified differen-

---

\* Received by the editors January 25, 1983, and in revised form March 5, 1984.

† Department of Mathematical Sciences, Rice University, Houston, Texas 77251.

cing operator can be implemented in existing reservoir models with very little additional effort. Hirasaki, O'Dell and Todd [5] propose a two-point upstream weighting technique. As is shown by McCracken and Yanosik [6], the two-point upwinding procedure exhibits significant grid effects at moderate mobility ratios. McCracken and Yanosik [6] propose a nine-point finite difference technique that eliminates grid effects at moderate mobility ratios, but exhibits unrealistic phenomena at a mobility ratio of fifty. Coats and Ramesh [7] report that the McCracken and Yanosik procedure exhibits a significant grid effect when simulating a steamflood in an inverted seven-spot pattern. Robertson and Woo [8] propose a scheme that utilizes a curvilinear grid. Although this procedure eliminates a substantial portion of the grid effects present in a steamflood model, the method has two drawbacks. First, an a priori estimate of the fluid streamlines is required to define the coordinate system. Fluid streamlines in a realistic process can vary rapidly as a function of time. Secondly, a considerable amount of user interaction is required for a transmissibility modifications associated with this method.

Finite element methods have the potential of eliminating the grid effects in the simulation of the miscible displacement problem. Finite element procedures which do not exhibit grid effects generally add diffusion in some manner. Dupont, Price, and Settari [9] report that grid orientation is present in a finite element procedure employing a  $C^0$  approximating space and is not present when a  $C^1$  approximating space is employed. Approximating spaces having continuous first derivatives add diffusion to the calculated solution. Douglas [10] reports that the grid orientation present in finite element methods which employ a  $C^0$  space can be eliminated if a term which penalizes jumps in the derivative of the approximate solution is added to the discretization procedure. A penalty term on the jumps of the approximate solution adds diffusion. Young [11] adds diffusion through a rotationally invariant physical dispersion term. He reports no grid orientation in finite element procedures employing approximating spaces with continuous first derivatives. The grid effects disappear as the norm of the mesh approaches zero when using a procedure based upon a  $C^0$  approximating space. There is, however, a significant grid effect in the procedure which uses a  $C^0$  approximating space at discretization sizes practical for field simulation work.

The need to add diffusion to finite element procedures is further demonstrated by the linear immiscible displacement problem. A finite element procedure can converge to a false solution if diffusion is not present. Price, Settari, and Spivak [12] demonstrate that finite element procedures employing approximating spaces with continuous first derivatives converge to the correct solution of the Buckley-Leveret problem and that similar procedures employing  $C^0$  approximating spaces do not converge unless diffusion is added to the differential equation by a capillary pressure term. Darlow, Douglas, Kendall, and Wheeler [13] show that the addition of a mesh dependent diffusion term forces a finite element method based upon a  $C^0$  approximating space to converge to the correct solution of the linear immiscible displacement problem. Furthermore, they relate single-point upstream weighting to this mesh dependent diffusion term. Douglas [10] notes significant overshoot in the computed solution for the miscible displacement problem unless diffusion is added to the approximate solution.

These studies indicate that a certain amount of numerical or physical diffusion must be present in a numerical model of fluid displacement to eliminate both grid effects and formation of false fronts. Although five-point finite difference procedures add mesh dependent dispersion to the numerical model [13], the added dispersion is not rotationally invariant in more than one space variable. The physical dispersion

proposed by Young [11] is rotationally invariant, which permits the asymptotic convergence of a finite element method that employs a  $C^0$  approximating space.

Finite difference procedures employing an extended differencing operator can be implemented in existing five-point finite difference reservoir models with little additional effort. The major drawback with applying finite element procedures to reservoir engineering models is that it is not possible to easily implement the numerical discretization procedure in existing models of complex physical phenomena. For example, often associated with finite element procedures are penalty terms [10], logarithmic approximations to the sources and sinks [14], a need for the pointwise evaluation of the Darcy velocity [9], [10], [11], [14], and capacity matrices that do not allow the implementation of an IMPES [15] solution technique. Young [11] directly or indirectly addresses several of these points, especially noting that a technique which employs reduced numerical quadrature allows the efficient implementation of an IMPES solution technique. Indeed, the procedure he describes that employs a  $C^0$  linear approximating space reduces to a five-point finite difference formulation. Regularization of the computed solution is achieved through a rotationally invariant physical dispersion term instead of by upwinding. As is noted earlier, the grid effects present in the  $C^0$  procedure are significant at practical discretization sizes.

Procedures based upon a modification of the method of characteristics eliminate grid orientation in the miscible displacement model problem. Furthermore, these procedures do not add numerical diffusion to the computed solutions. Examples of such work include Glimm, Isaacson, Marchesin, and McBryan [16], Russell [17], and Ewing, Russell, and Wheeler [18]. The major difficulty in applying these techniques in a general setting is twofold. Computational experience with these methods is limited for compressible problems, and the data structures arising in an implementation are inconsistent with those found in existing reservoir engineering models.

A new numerical model of single phase miscible displacement in a porous media is developed below. Instead of defining this procedure using the differential equations governing the physical process, this procedure is developed directly from material balance considerations. Hence this procedure is readily implemented in any existing code which views the numerical model as a material balance on the computational molecules associated with the spatial discretization. All finite difference procedures are compatible with this material balance framework. Upstream weighting is used to add dispersion to the numerical procedure. Although no attempt is made to prove either consistency with the differential equations governing single phase miscible displacement or convergence to the true solution of these equations, the theoretical results of Bell, Shubin, and Wheeler concerning this procedure are mentioned briefly. Lastly, computational results are presented which demonstrate the lack of grid effects in this procedure.

**2. Definition of the model.** The two-dimensional, two-component, single phase miscible displacement problem is a simple and unrealistic model problem. Nonetheless, the qualitative aspects of this model adequately represent some features encountered in the displacement of oil in a petroleum reservoir. A fluid that is known as the resident fluid is assumed to be originally present in the reservoir. A second fluid known as the invading fluid is injected at wells that are known as injection wells in an attempt to produce the resident fluid at other wells that are known as production wells. The process is assumed to be independent of the vertical coordinate, and is modeled in a two-dimensional domain. The height of the reservoir is assumed to be unity for convenience.

Over the domain  $\Omega$  entities that are known as computational molecules are defined. The computational molecules are denoted by  $\Omega_p$ . Denote the number of computational molecules by  $m$ . A precise definition of the computational molecules is not necessary at this point. However, the computational molecules formed by a five-point finite difference approximating technique are rectangles formed by the spatial discretization.

A modification of the standard analysis of fluid mechanics [20] that is ultimately employed to derive the differential equations governing a physical process is used below. For every computational molecule  $\Omega_p$ , the law of conservation of mass is expressed as a material balance given by

$$(1) \quad \{\text{accumulation of mass in } \Omega_p\} = \{\text{mass flux into } \Omega_p\} + \{\text{source terms acting on } \Omega_p\}.$$

Consider first the total flow of fluid into a computational molecule  $\Omega_p$ , irrespective as to whether the fluid is resident fluid or invading fluid. Let  $V_p$  denote the pore volume of the computational molecule  $\Omega_p$ . Under the assumption of single phase flow, the average density of the fluid in the computational molecule  $\Omega_p$  is represented by a single quantity  $\rho_p$ . The accumulation of mass in  $\Omega_p$  between time  $t_n$  and  $t_{n+1}$  is given by

$$(2) \quad \text{accumulation} = V_p[\rho_p(t_{n+1}) - \rho_p(t_n)].$$

Assume that the number of sources and sinks present in the domain  $\Omega$  is denoted by  $s$ . Let  $Q_p^k$  denote the strength of the  $k$ th source term with respect to the computational molecule  $\Omega_p$ . For a sink, this quantity is negative. The source term has units expressed in terms of mass injected per unit time. Letting  $Q_p^k(t_{n+1/2})$  denote the average value of the injection rate between time  $t_n$  and  $t_{n+1}$ , the contribution of the source terms is given by

$$(3) \quad \text{Sources} = \Delta t \sum_{k=1}^s Q_p^k(t_{n+1/2}).$$

Fluid flowing into the computational molecule  $\Omega_p$  that is not covered by the source terms must flow out of another computational molecule  $\Omega_q$  by convective flow. Denote the mass transfer rate between the computational molecule  $\Omega_p$  and  $\Omega_q$  as  $\Gamma_p^q$ . Since the amount of fluid flowing into  $\Omega_q$  that flows out of  $\Omega_p$  must be the negative of the amount of fluid flowing into  $\Omega_p$  that flows out of  $\Omega_q$ , the mass transfer rates satisfy the property that  $\Gamma_p^q = -\Gamma_q^p$ .

Assume that the system satisfies zero flux boundary conditions, i.e., that no fluid flow occurs across the boundary  $\partial\Omega$  of the domain  $\Omega$ . The total mass flux into a computational molecule  $\Omega_p$  is thus the sum of all mass transfer rates of the form  $\Gamma_p^q$ , where  $q$  can range from 1 to  $m$ . Let  $\Gamma_p^q(t_{n+1/2})$  denote the average of the mass transfer rate  $\Gamma_p^q$  between time  $t_n$  and time  $t_{n+1}$ . The total mass flux into a computational molecule  $\Omega_p$  between time  $t_n$  and time  $t_{n+1}$  due to convection is given by

$$(4) \quad \text{Mass flux} = \Delta t \sum_{q=1}^m \Gamma_p^q(t_{n+1/2}).$$

Given (2), (3), and (4), the material balance for the computational molecule  $\Omega_p$  is rearranged to yield

$$(5) \quad V_p[\rho_p(t_{n+1}) - \rho_p(t_n)] = \Delta t \sum_{q=1}^m \Gamma_p^q(t_{n+1/2}) + \Delta t \sum_{k=1}^s Q_p^k(t_{n+1/2}).$$

Let  $c$  denote the concentration of the invading fluid. A material balance that can be derived in a fashion similar to the derivation of (5) governing the concentration of the



invading component  $c_p$  in a computational molecule  $\Omega_p$ . This is given by

$$(6) \quad \begin{aligned} &V_p[\rho_p(t_{n+1})c_p(t_{n+1}) - \rho_p(t_n)c_p(t_n)] \\ &= \Delta t \sum_{q=1}^m \Gamma_p^q(t_{n+1/2})c_p^q(t_{n+1/2}) + \Delta t \sum_{k=1}^s Q_p^k(t_{n+1/2})\tilde{c}_p(t_{n+1/2}). \end{aligned}$$

The quantity  $c_p$  represents the average concentration of the invading fluid in the computational molecule  $\Omega_p$ . The quantity  $c_p^q$  appearing in (6) represents an average concentration of the invading fluid with respect to the computational molecules  $\Omega_q$  and  $\Omega_p$ . The average concentration is written with a tilde for the source terms to indicate that the invading fluid concentration is by definition unity at the injection wells and that it depends upon the local fluid composition at the production wells. Precisely, let

$$(7) \quad \tilde{c}_p(t_{n+1/2}) = \begin{cases} 1 & \text{if } Q_p^k > 0, \\ c_p(t_{n+1/2}) & \text{if } Q_p^k < 0. \end{cases}$$

The pressure distribution  $P$  and the distribution of the concentration of the invading fluid  $c$  are assumed to be a set of primary unknowns from which the values of all quantities appearing in the material balances (5) and (6) are determined. Average values  $P_p$  and  $c_p$  of the primary variables are defined for every computational molecule  $\Omega_p$ . The density of the fluid is assumed to be a function of the pressure and the concentration of the invading fluid. In this investigation, the density is related to the primary variables by

$$(8) \quad \rho(P, c) = \rho_i[1 + \alpha_i P]c + \rho_r[1 + \alpha_r P][1 - c].$$

The quantities  $\rho_i$  and  $\rho_r$  represent the densities of uncontaminated invading fluid and resident fluid, respectively. The quantities  $\alpha_i$  and  $\alpha_r$  represent the compressibilities of each fluid. The average density  $\rho_p$  of the fluid in a computational molecule  $\Omega_p$  is related by (8) to the average pressure  $P_p$  and the average concentration of the invading fluid  $c_p$ . Let

$$(9) \quad \rho_p = \rho(P_p, c_p).$$

Assume that the domain  $\Omega$  is a rectangular region of the form  $(0, x_L) \times (0, y_L)$ . A partition  $\pi_x = \{x_0, x_1, \dots, x_{n_x}\}$  of the  $x$  coordinate axis is formed from the rule

$$(10) \quad 0 = x_0 < x_1 < x_2 < \dots < x_{n_x} = x_L.$$

In a similar manner, a partition  $\pi_y$  of the  $y$  coordinate axis is formed from the rule

$$(11) \quad 0 = y_0 < y_1 < y_2 < \dots < y_{n_y} = y_L.$$

The partitions defined by (10) and (11) are extended by reflection to included additional points on the  $x$  or  $y$  coordinate axis. For example, the point  $x_{-1}$  is obtained by reflection of the point  $x_1$  through the point  $x_0$ .

Functions of a single space variable are associated with the partitions  $\pi_x$  and  $\pi_y$ . Functions denoted as  $X_i$  that are associated with the partition  $\pi_x$  are defined by

$$(12) \quad X_i(x) = \begin{cases} \frac{x - x_{i-1}}{x_i - x_{i-1}}, & x_{i-1} < x < x_i, \\ \frac{x_{i+1} - x}{x_{i+1} - x_i}, & x_i < x < x_{i+1}, \\ 0, & x < x_{i-1} \text{ or } x > x_{i+1}. \end{cases}$$

In a similar fashion, functions  $Y_j$  that are associated with the partition  $\pi_y$ , are defined by

$$(13) \quad Y_j(y) = \begin{cases} \frac{y - y_{j-1}}{y_j - y_{j-1}}, & y_{j-1} < y < y_j, \\ \frac{y_{j+1} - y}{y_{j+1} - y_j}, & y_j < y < y_{j+1}, \\ 0, & y < y_{j-1} \text{ or } y > y_{j+1}. \end{cases}$$

Define a node  $n_{ij}$  to be an element of the Cartesian product  $\pi_x \times \pi_y$ . Associated with each node  $n_{ij}$  is a function  $W_{ij}$  that is generated by the functions  $X_i$  and  $Y_j$ . Let

$$(14) \quad W_{ij}(x, y) = X_i(x) Y_j(y).$$

Associated with each node  $n_{ij}$  is a computational molecule  $\Omega_{ij}$ . Although the computational molecule  $\Omega_{ij}$  escapes a precise geometrical interpretation, it is in some sense related to the area under the curve formed by the function  $W_{ij}$  [21]. Let  $\phi$  denote the porosity of the formation. Consistent with the interpretation of the computational molecule  $\Omega_{ij}$  as the area under curve of  $W_{ij}$ , the pore volume  $V_{ij}$  is defined as

$$(15) \quad V_{ij} = \int_{x_{i-1}}^{x_{i+1}} \int_{y_{j-1}}^{y_{j+1}} \phi W_{ij}(x, y) dy dx.$$

For the purpose of discussion, a grid block is defined to be a spatial region of the form  $(x_i, x_{i+1}) \times (y_j, y_{j+1})$ . The computational molecule  $\Omega_{ij}$  can be thought of as existing over the four grid blocks  $(x_i, x_{i+1}) \times (y_j, y_{j+1})$ ,  $(x_i, x_{i-1}) \times (y_j, y_{j+1})$ ,  $(x_i, x_{i+1}) \times (y_j, y_{j-1})$ , and  $(x_i, x_{i-1}) \times (y_j, y_{j-1})$ , since the associated function  $W_{ij}$  is nonvanishing, i.e. supported, only over these grid blocks. For the purposes of discussion, a quantity  $\Delta x_i$  is associated with every element  $x_i$  of the partition  $\pi_x$  by

$$(16) \quad \Delta x_i = \frac{1}{2}[x_{i+1} - x_{i-1}].$$

A quantity  $\Delta y_j$  is defined for every element of  $\pi_y$  in a similar fashion.

Let  $u$  denote the mass flux distribution in the domain  $\Omega$  having directional components denoted by  $u_x$  and  $u_y$ . The average value between time  $t_n$  and  $t_{n+1}$  of the mass transfer rate  $\Gamma_{ij}^{gh}(t_{n+1/2})$  between any two computational molecules  $\Omega_{ij}$  and  $\Omega_{gh}$  is defined in terms of the average in value of the components of the mass flux distribution  $u_x(t_{n+1/2})$  and  $u_y(t_{n+1/2})$  and the functions  $X_g$ ,  $Y_h$ , and  $W_{ij}$ . Let

$$(17) \quad \Gamma_{ij}^{gh}(t_{n+1/2}) = \int_{x_{i-1}}^{x_{i+1}} \int_{y_{j-1}}^{y_{j+1}} \left\{ |i-g| u_x(x, y, t_{n+1/2}) \frac{\partial W_{ij}(x, y)}{\partial x} Y_h(y) + |j-h| u_y(x, y, t_{n+1/2}) \frac{\partial W_{ij}(x, y)}{\partial y} X_g(x) \right\} dy dx.$$

A mass transfer rate  $\Gamma_{ij}^{gh}$  between two computational molecules  $\Omega_{ij}$  and  $\Omega_{gh}$  calculated from (17) is nonzero only if the node  $n_{ij}$  is one of the eight nodes surrounding the node  $n_{ij}$ . The mass transfer rates calculated from (17) are, as desired, antisymmetric, i.e.,  $\Gamma_p^q = -\Gamma_q^p$ . Mass transfer rates from (17) can be calculated from any approximation to the mass flux distribution, including the exact distribution. A geometric interpretation of (17) is given by Potempa [21].

The material balance on total mass represented by (5) is rewritten as

$$(18) \quad V_{ij}[\rho_{ij}(t_{n+1}) - \rho_{ij}(t_n)] = \Delta t \sum_{g=-1}^1 \sum_{h=-1}^1 \Gamma_{ij}^{gh}(t_{n+1/2}) + \Delta t \sum_{k=1}^s Q_{ij}^k(t_{n+1/2}).$$

In a similar manner, the material balance (6) on the invading component is rewritten as

$$(19) \quad V_{ij}[\rho_{ij}(t_{n+1})c_{ij}(t_{n+1}) - \rho_{ij}(t_n)c_{ij}(t_n)] = \Delta t \sum_{g=-1}^1 \sum_{h=-1}^1 \Gamma_{ij}^{gh}(t_{n+1/2})c_{ij}^{gh}(t_{n+1/2}) \\ + \Delta t \sum_{k=1}^s Q_{ij}^k(t_{n+1/2})\tilde{c}_{ij}(t_{n+1/2}).$$

Upstream weighting based upon the sign of the appropriate mass transfer rate is applied to the average concentration  $c_{ij}^{gh}(t_{n+1/2})$  appearing in (19). Following the IMPES technique, this term is approximated explicitly in time. Let

$$(20) \quad c_{ij}^{gh}(t_{n+1/2}) = \begin{cases} c(t_n)_{ij} & \text{if } \Gamma_{ij}^{gh}(t_{n+1/2}) < 0, \\ c(t_n)_{gh} & \text{if } \Gamma_{ij}^{gh}(t_{n+1/2}) > 0. \end{cases}$$

Also following the IMPES analysis, the concentration at the source terms is evaluated explicitly in time. Let

$$(21) \quad \tilde{c}_p(t_{n+1/2}) = \begin{cases} 1 & \text{if } Q_p^k > 0, \\ c_p(t_n) & \text{if } Q_p^k < 0. \end{cases}$$

Define the total mass mobility  $\xi$  as

$$(22) \quad \xi(P, c) = \frac{k\rho(P, c)}{\mu(c)}.$$

The coefficient  $k$  appearing in (22) is known as the permeability of the rock. Although the permeability is in general both a spatially dependent function and a directional quantity, assume that it is an isotropic constant for the sake of brevity. Following Darlow, Ewing, and Wheeler [14], the viscosity  $\mu$  is assumed to be a function of the concentration of the invading component of the form

$$(23) \quad \mu(c) = \{\mu_i^{1/4}c + \mu_r^{1/4}[1 - c]\}^4.$$

The mobility ratio  $M$  of the flood is defined to be the ratio of the viscosity  $\mu_r$  of the resident fluid to the viscosity  $\mu_i$  of the invading fluid. Let

$$(24) \quad M = \frac{\mu_r}{\mu_i}.$$

The mass flux distribution is related to the primary unknowns by Darcy's law and by the definition of the mass mobility. Let

$$(25) \quad u(x, y, t_{n+1/2}) = -\xi(P(x, y, t_{n+1/2}), c(x, y, t_{n+1/2}))\nabla P(x, y, t_{n+1/2}).$$

The distribution of the mass flux at each point in the domain  $\Omega$  required by (17) is determined from a pointwise approximation to the primary unknowns and (25). A pointwise approximation  $\bar{P}$  to the pressure distribution is obtained from a linear combination of the functions  $W_{ij}$  and the average values of the pressure  $p_{ij}$  in the computation molecules  $\Omega_{ij}$ . Let

$$(26) \quad \bar{P}(x, y, t_n) = \sum_{i=0}^{n_x} \sum_{j=0}^{n_y} P_{ij}(t_n) W_{ij}(x, y).$$

In a similar fashion, an approximation  $\bar{c}$  to the concentration of the invading fluid is given by

$$(27) \quad \bar{c}(x, y, t_n) = \sum_{i=0}^{n_x} \sum_{j=0}^{n_y} c_{ij}(t_n) W_{ij}(x, y).$$

The approximation to the mass flux distribution used in the implementation of this procedure is further simplified to permit an efficient calculation of the mass transfer rates. The total mass mobility  $\xi$  is evaluated explicitly in time and as a piecewise constant over each grid block. Let

$$(28) \quad \begin{aligned} \bar{\xi}(P(x, y, t_{n+1/2}), c(x, y, t_{n+1/2})) &= \xi(\bar{P}(x_{i+1/2}, y_{j+1/2}, t_n), \bar{c}(x_{i+1/2}, y_{j+1/2}, t_n)), \\ (x, y) &\in (x_i, x_{i+1}) \times (y_j, y_{j+1}). \end{aligned}$$

Using (25) and (28), the mass flux distribution is approximated as

$$(29) \quad \bar{u}(x, y, t_{n+1/2}) = -\bar{\xi}(P(x, y, t_{n+1/2}), c(x, y, t_{n+1/2})) \nabla \bar{P}(x, y, t_{n+1}).$$

The vector  $\bar{u}$  calculated by (29) is used computationally as an approximation to the mass flux distribution in (17).

Although the evaluation of (17) appears formidable, only the functions  $X_i$ ,  $Y_j$ , and  $W_{ij}$  remain under the integral sign if it is expanded through the definitions given above. These integrations are performed separately on each grid block. Recall that the total mass mobility is approximated as a constant over each grid block and is thus removed from under the integral sign. The remaining functions appearing in (17) are constant with respect to time and are polynomials in space. The integrals are evaluated exactly for every grid block only once by using an appropriate master grid block and a mapping. The mass transfer rates between adjacent computational molecules are thus written as easy to compute linear combinations of either four or six nodal values of the pressure. The zero flux boundary conditions are imposed by reflection.

In finite difference procedures, the approximate mass transfer rates are directly proportional to the pressure difference. In contrast, the procedure described above determines approximate mass transfer rates by the integration of a mass flux vector which is the weighted gradient of a piecewise differentiable representation of the pressure distribution. By using an appropriate numerical quadrature scheme, these integrals are evaluated as a linear combination of nodal values of the pressure.

Other approximations to the mass transfer rates can be obtained by using different quadrature rules. If the mass flux vector is assumed constant over each individual grid block and every grid block is a square, the formula of Yanosik and McCracken [6] can be obtained from (17). Evaluation of (17) using the Lobatto points, as in Young [11], results in a five point finite difference formulation. If the components of the mass flux  $u_x$  and  $u_y$  are known exactly, these values can be used in (17). Numerical experiments of Potempa [21] indicate that use of a higher order approximation to the mass flux in (17) further reduces the amount of grid orientation.

Injection rates are assumed to be specified at the injection wells. A well model is used for calculating the injection rates at the production wells. The form of the well model is that used by Peaceman [22]. Assume that uniform, square grid blocks are used, i.e., that  $\Delta x_i = \Delta y_j = \Delta$  for all pairs of indices  $i$  and  $j$ . Let the  $k$ th well be located at  $(x_k, y_k)$ , the bottom hole pressure of the  $k$ th well be denoted by  $P_{bh}^k$ , and the wellbore radii of the  $k$ th well be represented by  $r_k$ . The injection rate  $Q^k$  at the  $k$ th production well is calculated from

$$(30) \quad Q^k(t_{n+1/2}) = 2\pi \bar{\xi}(P(x_k, y_k, t_n) c(x_k, y_k, t_n)) \frac{P_{bh} - \bar{P}(x_k, y_k, t_{n+1})}{\ln(.1168\Delta) - \ln(r_k)}.$$

The constant .1168 is calculated directly from the simulator in the manner described by Peaceman [22] and Coats and Ramesh [7]. All coefficients appearing in (30), except the local reservoir pressure, are evaluated explicitly in time. This definition is modified in the manner described by Peaceman [22] if the grid is not uniform.

Given the approximation to the injection rates, the injection rate into a computational molecule  $\Omega_{ij}$  due to a source of strength  $Q^k$  is given by

$$(31) \quad Q_{ij}^k(t_{n+1/2}) = Q^k(t_{n+1/2}) W_{ij}(x_k, y_k).$$

The wells are not restricted to lie at the nodes of the discretization. This is an improvement over finite difference formulations, where the uncertainty in the location of a well is the size of the grid block in which it is located. Included in the computational results is a simulation of an inverted seven spot pattern in which some of the wells do not lie at the nodes of the discretization.

**3. Theoretical results.** In this section, the fluid is assumed to be incompressible, i.e., that  $\rho = \rho_i = \rho_r$  and  $\alpha_i = \alpha_r = 0$  in (8). Given Darcy's law for the mass flux (25), the following system of partial differential equations govern the incompressible miscible displacement process. An equation known in the literature as the pressure equation is given by

$$(32) \quad \nabla \cdot \mathbf{u} = \sum_{k=1}^s Q_k \delta(x - x_k) \delta(y - y_k).$$

Let  $D$  denote a molecular diffusivity. The distribution of the invading fluid is governed by the concentration equation given by

$$(33) \quad \phi \rho \frac{\partial c}{\partial t} + \nabla \cdot c \mathbf{u} - \nabla \cdot D \nabla c = \sum_{k=1}^s Q_k \tilde{c} \delta(x - x_k) \delta(y - y_k).$$

Let  $N$  denote the outward pointing normal on the boundary  $\partial\Omega$  of the region  $\Omega$ . The following boundary conditions are imposed. Let

$$(34) \quad \mathbf{u} \cdot N = 0 \quad \text{on } \partial\Omega.$$

Bell, Shubin, and Wheeler [18] present a detailed proof for the procedure detailed above of convergence to the true solution of (32)–(34) under the assumptions of both smoothly distributed sources and sinks and a positive molecular diffusivity  $D$ . The rate convergence is shown to be of order of the norm of the mesh  $\Delta$ . They also prove that the procedure outlined above is consistent with (32)–(34) if  $D$  is equal to zero. The numerical dispersion tensor that is added by the procedure is demonstrated to be nearly rotationally invariant.

Although it is a nontrivial matter to demonstrate the consistency of this procedure with the concentration equation, it is relatively straightforward to show consistency with the pressure equation. In the incompressible case, the injection rates at the production wells are specified by the condition of incompressibility. Dropping the indexing denoting time, substitution of (17) and (31) into (18) yields

$$(35) \quad - \int_{x_{i-1}}^{x_{i+1}} \int_{y_{j-1}}^{y_{j+1}} \mathbf{u}(x, y) \cdot \nabla W_{ij}(x, y) \, dy \, dx = \sum_{k=1}^s Q^k W_{ij}(x_k, y_k).$$

This equation is weak formulation of the pressure equation (32) subject to the zero flux boundary conditions (34).

**4. The discretized system of nonlinear equations.** A symbolic form of the total material balance which is enforced for each computational molecule  $\Omega_{ij}$  is

$$(36) \quad \Phi_{ij}(c_{ij}(t_{n+1}), \mathbf{P}(t_{n+1})) = 0.$$

The vector  $\mathbf{P}$  indicates that a material balance depends upon the value of the pressure

at the new time level in not only the computational molecule  $\Omega_{ij}$  but also in the ten surrounding molecules. In a similar manner, the material balance governing the approximate distribution of the concentration of the invading component is

$$(37) \quad \Xi_{ij}(c_{ij}(t_{n+1}), \mathbf{P}(t_{n+1})) = 0.$$

The material balances represented by (36) and (37) are solved simultaneously for the composition and the pressure with respect to every computational molecule.

The resulting system of nonlinear equations is solved in the following fashion. The initial guess for the average pressure values  $P_{ij}^0(t_{n+1})$  at the new time level are the average pressure values  $P_{ij}(t_n)$  computed at the old time level. A sequence  $\{P_{ij}^0(t_{n+1}), P_{ij}^1(t_{n+1}), \dots\}$  is computed until convergence as follows. Given the  $k$ th pressure iterate at time  $t_{n+1}$ , the material balance on the invading component represented by (37) is readily solved for the  $k$ th iterate of the concentration  $c_{ij}^k(t_{n+1})$ . The  $k$ th residual  $\Phi_{ij}^k$  of the total material balance is computed from (36). The norm of  $\Phi_{ij}^k$  is computed in units of density. If the norm of the residual is less than a specified tolerance,  $P_{ij}(t_{n+1})$  and  $c_{ij}(t_{n+1})$  are set equal to  $P_{ij}^k(t_{n+1})$  and  $c_{ij}^k(t_{n+1})$ , respectively. Otherwise, a new pressure iterate is obtained using Newton's method.

The Newton step is calculated by requiring that the differential of the total material balance with respect to the finite pressure distribution for all computational molecules  $\Omega_{ij}$  be equal to the negative of the residual in the total material balance. The elements  $J_{ij}^{gh}$  of the associated Jacobian matrix are determined from

$$(38) \quad J_{ij}^{gh} = \frac{\partial \Phi_{ij}}{\partial P_{gh}} + \frac{\partial \Phi_{ij}}{\partial c_{ij}} \frac{\partial \Xi_{ij}^{-1}}{\partial c_{ij}} \frac{\partial \Xi_{ij}}{\partial P_{gh}}.$$

The Newton step satisfies

$$(39) \quad \sum_{g=0}^{n_x} \sum_{h=0}^{n_y} J_{ij}^{gh} (P_{gh}^{k+1} - P_{gh}^k) = -\Phi_{ij}^k.$$

Application of (39) to every computational molecule generates a banded system of linear equations, which is then solved exactly using a banded matrix routine.

**5. Computational results.** The numerical simulations using this new procedure as a model of single phase miscible displacement are presented. Both the recovery curves and contour plots showing the computed distribution of the concentration of the invading fluid are presented. The contours correspond to the locations where the concentration of the invading fluid is .1, .3, .5, .7, and .9, respectively. The injection wells are labeled by the symbol  $\emptyset$ , the production wells are labeled as  $\circ$ , and the wells that have been shut in are labeled as  $\oplus$ . For the purpose of comparison, simulations employing a parallel grid are pictorially represented on a diagonal grid.

The measure of time used in presenting these results is in terms of the number of pore volumes of fluid injected. For compressible problems, a pore volume of fluid is defined to be the amount of the fluid that occupies the amount of pore space available in the simulated pattern at standard conditions. The number of pore volumes of the fluid injected into a reservoir is abbreviated as PVI.

The physical parameters utilized in this study are listed in Table 1. A large fluid compressibility is used to study the sensitivity of the model to the compression effects. Injection rates are given in terms of the total injection rate in the repeated pattern as opposed to the alternative specification of total injection rate into the simulated pattern. No limiting rate is specified at the production wells.

TABLE 1  
Reservoir and fluid properties.

Patterns	4,000,000	sq ft five-spot
	10,392,000	sq ft seven-spot
Permeability	10	md
Fluid viscosity		
Resident fluid	1	cp
Invading fluid	.01	cp
Density		
Resident fluid	60	lbs/cu ft
Invading fluid	60	lbs/cu ft
Compressibility		
Resident fluid	.00001	1/psi
Invading fluid	.00002	1/psi
Porosity	.1	
Injection rate	200	cu ft/day
Well bore radius	.5	ft
Bottom hole pressure	100	psi
Initial pressure	100	psi
Time steps	2.5	days

The basic model problem considered is displacement at an unfavorable mobility ratio of 100. Figure 1 exhibits the compositional profiles generated by this procedure at .5 pore volumes injected for a  $12 \times 12$  diagonal grid and a  $17 \times 17$  parallel grid. The recovery curves for both grids until 1.5 pore volumes of fluid are injected are presented in Fig. 2. This procedure exhibits very little grid orientation at unfavorable mobility ratios in both the recovery curves and the compositional profiles.

Recall that the mass transfer rates between two computational molecules generated by (17) are evaluated inexactly in this implementation of this procedure. The mass mobility is a spatially dependent function which depends upon the local fluid composition. This procedure evaluates the mass mobility at only a single point within each

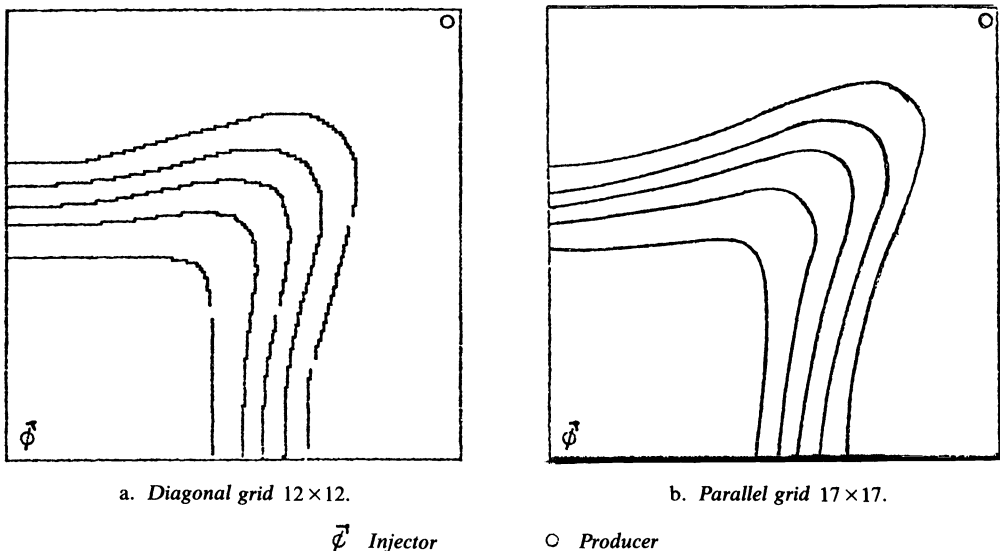


FIG. 1. Compositional profiles at .5 PVI for  $M = 100$ .

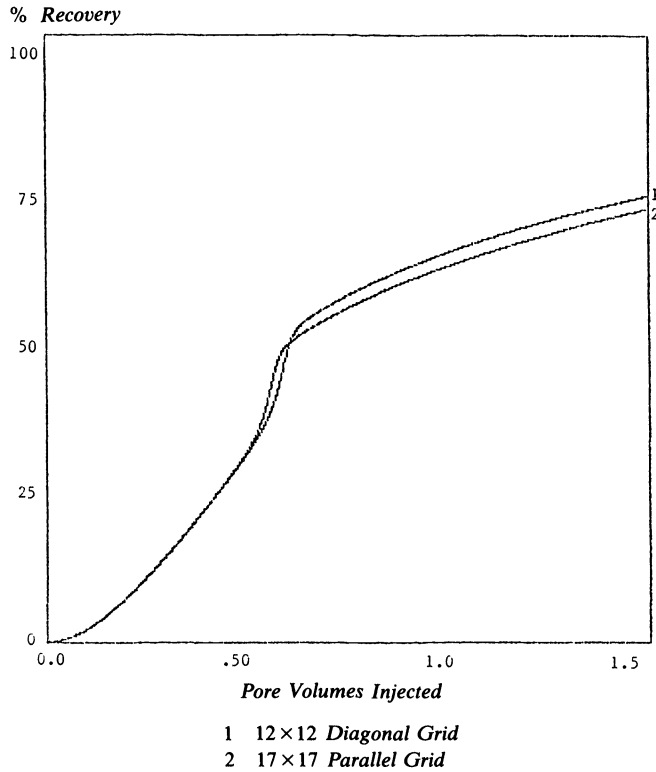


FIG. 2. Recovery curves for the model at  $M = 100$ .

grid block. This integration scheme does not provide enough accuracy to completely control the amount of grid orientation present in the model. The amount of this error can be reduced by using very accurate numerical quadratures. If a two-point Gauss quadrature rule is employed within each grid block to evaluate the integrals governing mass transfer rates, the amount of grid orientation present in this procedure is decreased by about 66% [21].

This procedure is utilized to model an inverted seven-spot pattern. The pattern and a discretization which preserves zero flux boundary conditions are illustrated in Fig. 3. The compositional profiles generated by this procedure for this model problem at .5 and .625 pore volumes injected for an unfavorable mobility ratio displacement

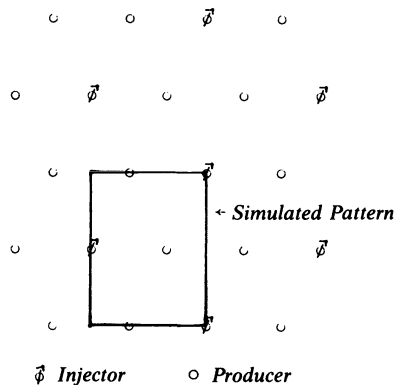


FIG. 3. Inverted seven-spot pattern.



of 100 are illustrated in Fig. 4. The radial flow at early time is evident in Fig. 4. An inverted seven-spot pattern helps prevent the formation of a viscous finger, which allows a higher percentage of the resident fluid to be recovered. The wells located at (0,1732) and at (2,000, 1,732) do not lie at nodes of the discretization. This substantiates the claim that this procedure accurately models the exact location of injection and production wells, unlike finite difference models.

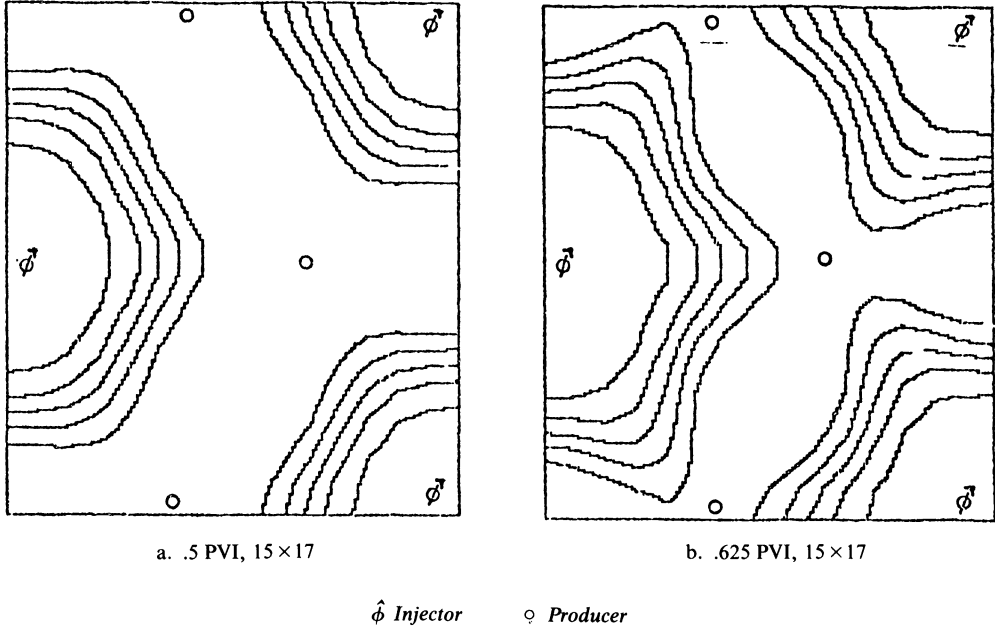


FIG. 4. Compositional profiles for the inverted seven-spot pattern.

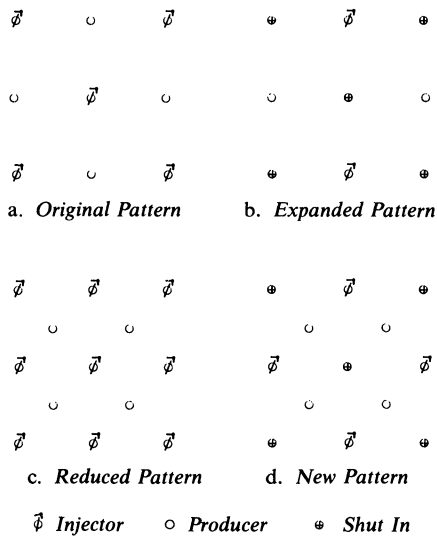
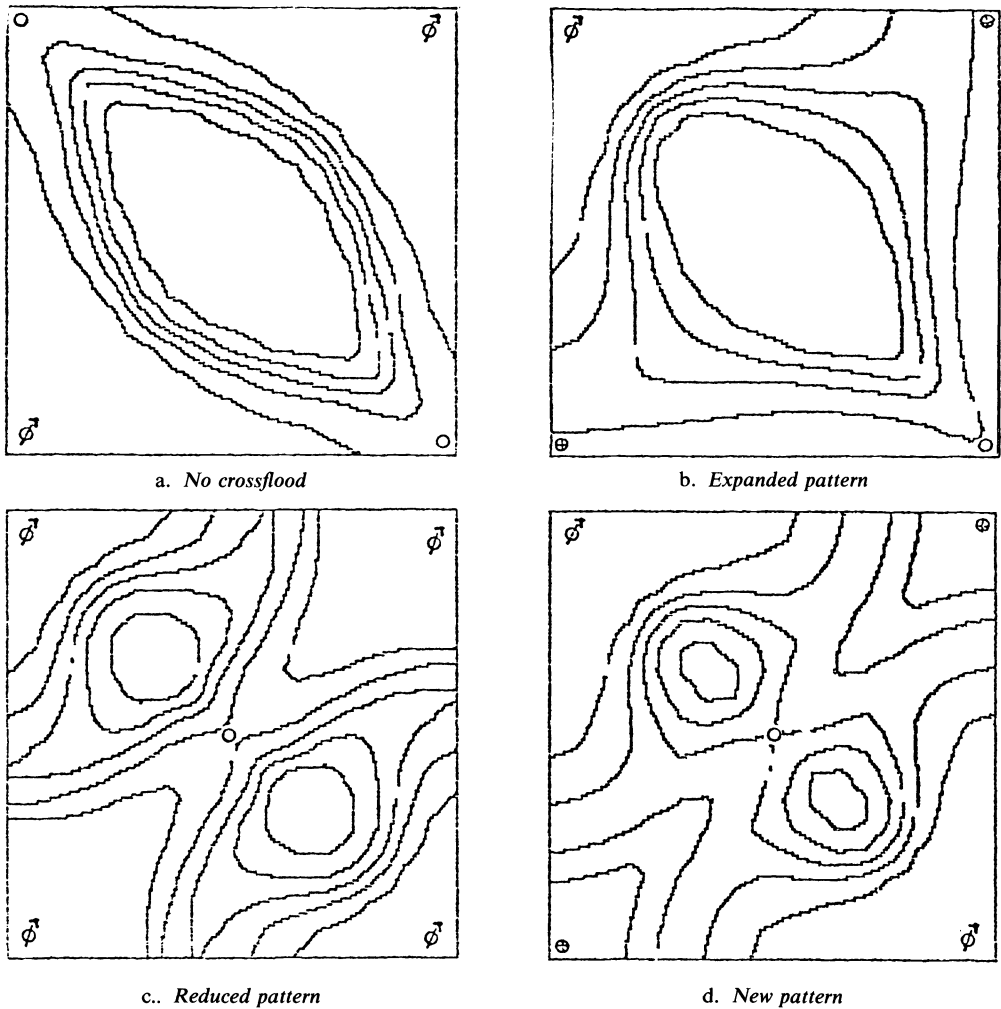


FIG. 5. Crossflooding patterns.

The computational results presented thus far demonstrate the insensitivity of this procedure to the orientation of the grid. This is an important consideration when modeling processes where the physical streamlines do not follow the artificial streamlines induced by the discretization. A physical displacement process where complex stream functions arise is the process of crossflooding a reservoir. Crossflooding is a proven technique for increasing the pattern efficiency of water flooding [23]. There are two methods of crossflooding a reservoir. The old well configuration is expanded to change the streamlines of the fluid flow. This process is an expanded pattern crossflood. Infill drilling is done to produce the blind spot of the old pattern. This process is a reduced pattern crossflood. The original pattern, the expanded pattern, the reduced pattern, and a fourth pattern are illustrated in Fig. 5. The new pattern for crossflooding is a result of considering the simulated compositional profiles for the reduced pattern crossflood using this numerical model.



$\phi$  Injector     $o$  Producer     $\oplus$  Shut In

FIG. 6. Crossflooding profiles at 1 PVI.

The original five-spot pattern is produced until .5 pore volumes of fluid are injected. Crossflooding is initiated at this time, and crossflooding continues until 1.5 pore volumes of fluid are injected. The compositional profiles of the various crossflooding processes are exhibited for 1 pore volume of fluid injected for an unfavorable mobility ratio of 100 in Fig. 6. The recovery curves for crossflooding until 1.5 pore volumes of fluid are injected are illustrated in Fig. 7.

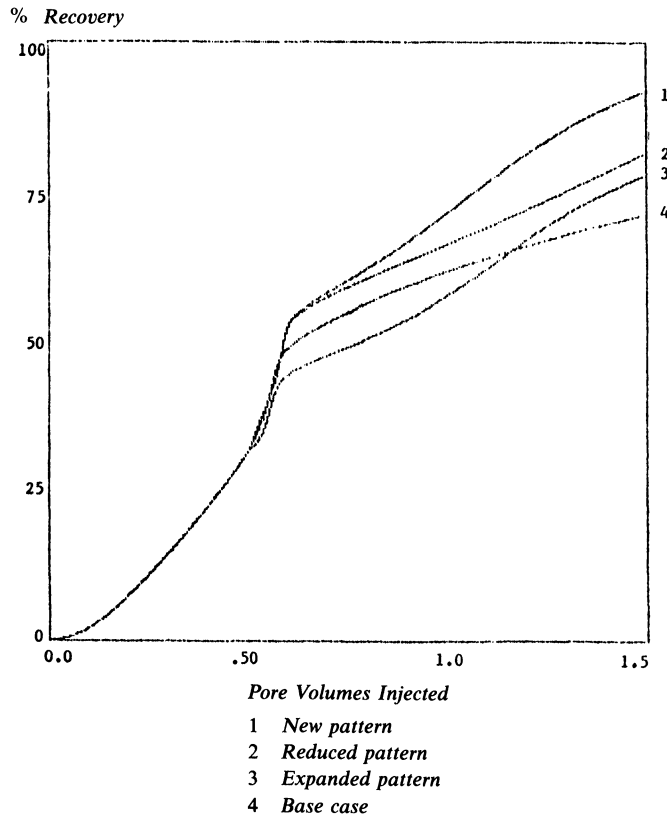


FIG. 7. Recovery curves for crossflooding at unfavorable mobility.

The blind spot to the original pattern is clearly illustrated in Fig. 6. The invading fluid follows the path of least resistance in the expanded pattern crossflood. This path does not displace the agglomeration of resident fluid in the blind spot. Instead, the invading fluid circumvents the blind spot and quickly reaches the production wells. The invading fluid from the original injection wells does not displace any of the resident fluid in the reduced pattern crossflood. Instead, the invading fluid injected into the old injection wells flows directly toward the production wells. By shutting in the old injection wells in the reduced pattern crossflood, the displacement efficiency of the invading fluid is significantly increased. The blind spot in the new pattern crossflood is extremely small compared to the other crossflooding alternatives. The efficient use of the invading fluid in a crossflood is important when the invading fluid is either produced from hydrocarbons, such as polymers or surfactants, or is generated by the combustion of hydrocarbons, such as steam.

**6. Conclusions.** A numerical procedure for modeling two-dimensional, two-component, single phase miscible displacement in a porous media is presented above.

Since this procedure is formulated directly from material balance considerations, it is compatible with many of the finite difference models of miscible displacement. The computational results demonstrate that this procedure adequately models single phase compressible miscible displacement problems. The method shows little grid orientation. The technique produces qualitatively correct results for problems in which the fluid flow does not follow the artificial channels formed by the discretization. Some useful results applicable to the enhanced recovery technique of crossflooding are presented.

**Acknowledgments.** I would like to thank Mary Wheeler, Trond Steihaug, and John Dennis of Rice University for their invaluable suggestions and criticisms regarding this work. I would also like to thank LCSE of Rice for providing the computer time necessary for the numerical experiments and the documentation of this work.

#### Nomenclature.

$c$	Concentration of the invading component.	$X$	Basis function over the partition $\pi_x$ .
$k$	Absolute permeability in .006336 * md.	$Y$	Basis function over the partition $\pi_y$ .
$n$	A node.	$\alpha$	Compressibility, $\text{psi}^{-1}$ .
$t$	Time, days.	$\mu$	Viscosity, cp.
$u$	Mass flux, lb/sq ft/day.	$\pi$	Partition of a coordinate axis.
$x$	X coordinate, ft.	$\rho$	Density, lb/cu ft.
$y$	Y coordinate, ft.	$\phi$	Porosity.
$D$	Diffusivity, lb/ft/day.	$\xi$	Mass mobility, .006336 md lb/cp/cu ft.
$M$	Mobility ratio.	$\Delta$	Distance operator on the partitions $\pi_x$ and $\pi_y$ .
$N$	Outward normal to a region $\Omega$ .	$\Gamma$	Mass transfer rate, lb/day.
$P$	Pressure, psi.	$\Phi$	Material balance on the total mass.
$Q$	Mass injection rate at a source, lb/day.	$\Xi$	Material balance on the invading component.
$V$	Volume, cu ft.	$\Omega$	A domain.
$W$	Basis function over the domain $\Delta$ .	$\partial$	Boundary operator.

#### REFERENCES

- [1] C. CHU, K. H. COATS, W. D. GEORGE AND B. E. MARCUM, *Three dimensional simulation of steamflooding*, Soc. Pet. Eng. J. Trans. AIME, 14 (1974), pp. 573-592.
- [2] G. K. YOUNGREN, *Development and application of an in situ combustion reservoir simulator*, Soc. Pet. Eng. J., Trans. AIME, 20 (1980), pp. 39-51.
- [3] R. C. NELSON AND G. A. POPE, *A chemical flooding compositional simulator*, Soc. Pet. Eng. J., Trans. AIME, 18 (1978), pp. 339-354.
- [4] K. AZIZ, D. K. FONG AND L. X. NGHIE, *Compositional modeling with an equation of state*, Soc. Pet. Eng. J., Trans. AIME, 21 (1981), pp. 687-698.
- [5] M. R. TODD, P. M. O'DELL AND G. J. HIRASAKI, *Methods for increased accuracy in numerical simulation*, Soc. Pet. Eng. J., Trans. AIME, 12 (1972), pp. 515-530.
- [6] T. MCCracken AND J. YANOSIK, *A nine-point, finite difference reservoir simulation for realistic prediction of adverse mobility ratio displacements*, Soc. Pet. Eng. J., Trans. AIME, 19 (1979), pp. 253-262.
- [7] K. H. COATS AND A. B. RAMESH, *Effect of grid type and difference scheme on pattern steamflood simulation results*, SPE paper 11079, 6th SPE Symposium on Reservoir Simulation, New Orleans, LA, January 31-February 3, 1982.
- [8] G. E. ROBERTSON AND P. T. WOO, *Grid orientation effects and the use of orthogonal curvilinear coordinates in reservoir simulation*, Soc. Pet. Eng. J., Trans. AIME, 18 (1978), pp. 13-19.
- [9] T. DUPONT, H. S. PRICE AND A. SETTARI, *Description and analysis of variational methods for solution of miscible displacement problems in porous media*, Soc. Pet. Eng. J., Trans. AIME, 17 (1977), pp. 228-246.
- [10] J. DOUGLAS, Jr., *The numerical simulation of miscible displacement in porous media*, Computation Methods in Nonlinear Mechanics, North-Holland, Amsterdam, 1980, pp. 225-238.
- [11] L. YOUNG, *A finite element method for reservoir simulation*, Soc. Pet. Eng. J., Trans. AIME, 21 (1981), pp. 115-128.

- [12] H. S. PRICE, A. SETTARI AND A. SPIVAK, *Solution of equations for multi-dimensional, two-phase, immiscible flow by variational methods*, SPE paper 5723, 4th Symposium on Numerical Simulations of Reservoir Performance, Los Angeles, CA, February 19-20, 1976.
- [13] B. D. DARLOW, J. DOUGLAS, Jr., R. P. KENDALL AND M. F. WHEELER, *Self adaptive finite elements and finite difference methods for one-dimensional, two-phase, immiscible flow*, this Journal, to appear.
- [14] B. D. DARLOW, R. EWING AND M. F. WHEELER, *Mixed finite element methods for miscible displacement problems in porous media*, SPE paper 10501, 6th SPE Symposium on Reservoir Simulation, New Orleans, LA, January 31-February 3, 1982.
- [15] D. BAVLY, C. D. HARRIS AND J. W. SHELDON, *A method for general reservoir behavior simulation on digital computers*, SPE paper 1521-G, 35th SPE Annual Fall Meeting, Denver, CO, October 2-5, 1960.
- [16] J. GLIMM, E. ISAACSON, D. MARCHESIN AND O. MCBRYAN, *A shock tracking method for hyperbolic systems*, presented at the 1980 Army Numerical Analysis and Computers Conference, Moffet Field, CA, February 20-21, 1980.
- [17] T. F. RUSSELL, *Finite elements with characteristics for two component incompressible miscible displacement*, SPE paper 10500, 6th SPE Symposium on Reservoir Simulation, New Orleans, LA, January 31-February 3, 1982.
- [18] R. EWING, T. F. RUSSELL AND M. F. WHEELER, *Modified method of characteristics and mixed finite element methods for miscible displacement problems in porous media*, in preparation.
- [19] J. BELL, G. SHUBIN AND M. F. WHEELER, *Analysis of a new method for computing the flow of miscible fluids in a porous media*, to appear.
- [20] R. B. BIRD, W. E. LIGHTFOOT AND E. N. STEWART, *Transport Phenomena*, John Wiley, New York, 1960.
- [21] T. C. POTEPA, *Finite element methods for convection dominated transport problems*, Ph.D. Thesis, Rice Univ., Houston, 1982.
- [22] D. W. PEACEMAN, *Interpretation of well-block pressures in numerical reservoir simulation with nonsquare grid blocks and anisotropic permeability*, SPE paper 10528, 6th SPE Symposium on Reservoir Simulation, New Orleans, LA, January 31-February 3, 1982.
- [23] U. N. VADGAMA AND B. B. HINKLE, *Crossflooding to improve waterflooding efficiency in Big Sinking Field, Kentucky*, J. Petrol. Tech., Trans. AIME, 25 (1973), pp. 1021-1024.

## PRECONDITIONING OF TRUNCATED-NEWTON METHODS\*

STEPHEN G. NASH†

**Abstract.** In this paper we discuss the use of truncated-Newton methods, a flexible class of iterative methods, in the solution of large-scale unconstrained minimization problems. At each major iteration, the Newton equations are approximately solved by an inner iterative algorithm. The performance of the inner algorithm, and in addition the total method, can be greatly improved by the addition of preconditioning and scaling strategies. Preconditionings can be developed using either the outer nonlinear algorithm or using information computed during the inner iteration. Several preconditioning schemes are derived and tested.

Numerical tests show that a carefully chosen truncated-Newton method can perform well in comparison with nonlinear conjugate-gradient-type algorithms. This is significant, since the two classes of methods have comparable storage and operation counts, and they are the only practical methods for solving many large-scale problems. In addition, with the Hessian matrix available, the truncated-Newton algorithm performs like Newton's method, usually considered the best general method for this problem.

**Key words.** unconstrained optimization, truncated-Newton algorithm, preconditioning strategies, linear conjugate-gradient algorithm

**1. Introduction.** The problem of minimizing a real-valued function of  $n$  variables

$$(1) \quad \min_x F(x)$$

arises in many contexts and applications. For the purposes of this paper, we assume that  $F$  is bounded and twice continuously differentiable. In particular, we are interested in solving problems for which the number of variables is large, but where the gradient of  $F$  is available.

The most effective general method for solving (1) is Newton's method, which takes full advantage of first- and second-derivative information about the function  $F$ . In its modern, safe-guarded implementations, it provides a standard for measuring the effectiveness of other algorithms. In Newton's method, the direction of search  $p$  is computed from the "Newton equations"

$$(2) \quad G^{(k)}p = -g^{(k)},$$

where  $g^{(k)}$  and  $G^{(k)}$  are, respectively, the gradient vector and Hessian matrix of second derivatives of  $F$  evaluated at the current iterate  $x^{(k)}$ . When the number of variables  $n$  is large, solving (2) can be expensive and can require the storage of an  $n$  by  $n$  matrix, which may be infeasible. Also, it is necessary to compute  $G^{(k)}$  at every iteration. Even for many small problems, this may be very costly. A large-scale problem will often have a sparse Hessian matrix, i.e. the Hessian matrix will have few nonzero entries. This allows Newton's method to be extended to large-scale problems through the use of finite-differencing and sparse-matrix techniques (Powell and Toint (1979)). However, in many contexts (constrained optimization, probability density estimation, etc.) this may not be possible.

Because the Newton equations are based on a Taylor series expansion near the solution  $x^*$  of the minimization problem (1), there is no guarantee that the search direction they compute will be as crucial far away from  $x^*$ . At the beginning of the solution process, a reasonable approximation to the Newton direction may be almost

---

\* Received by the editors November 30, 1982, and in final revised form March 9, 1984. This research was supported by the National Science Foundation under grants MCS-7926009 and ENG77-06761, and by a postgraduate scholarship from the Natural Sciences and Engineering Research Council of Canada.

† Mathematical Sciences Department, Johns Hopkins University, Baltimore, Maryland 21218.

as effective as the Newton direction itself. It is only gradually, as the solution is approached, that the Newton direction takes on more and more meaning.

These comments suggest that, for large-scale problems, it is sensible to use an iterative method to approximately solve the Newton equations (2). Moreover, it should be an iterative method with a variable tolerance, so that far away from the solution, (2) is not solved to undue accuracy. Only when the solution is approached should we consider expending enough effort to compute something like the exact Newton direction. As we approach the solution, the Hessian  $G^{(k)}$  will converge to  $G(x^*)$ . Consequently, by exploiting information from previous iterations, it is possible that a closer approximation to the exact solution can be determined with no increase in effort.

We will refer to any method which uses an iterative algorithm to approximately solve the Newton equations as a truncated-Newton method. Sherman (1978) suggested using Successive-Over-Relaxation (SOR). This is the simplest of a whole class of methods which have been found to be effective for solving linear systems which arise in partial differential equations. However, it can be difficult to get SOR methods to perform well on general problems. Also, they appear to be prohibitively expensive to use in the context of truncated-Newton methods. The number of linear subiterations required to achieve superlinear convergence increases exponentially at each nonlinear iteration. (Notice that a truncated-Newton method is doubly iterative: there is an outer "nonlinear" iteration to minimize the function  $F(x)$ , and an inner "linear" iteration to compute a search direction from the Newton equations (2).)

Various authors (Dembo and Steihaug (1983), Garg and Tapia (1980), O'Leary (1982)) have suggested using variants of the linear conjugate-gradient method. Although it is ideal for problems where the coefficient matrix has only a few distinct eigenvalues, it is guaranteed to converge (in exact arithmetic) in at most  $n$  iterations for any positive-definite symmetric matrix. Thus, the type of exponential growth mentioned above for SOR-type methods is impossible, at least theoretically.

A requirement of the linear conjugate-gradient method is that the coefficient matrix must be positive definite. Unfortunately, the Hessian matrix is only guaranteed to be positive semidefinite at the solution and may be indefinite elsewhere. Thus, whatever iterative method is chosen to solve (2), it must be able to detect and cope with indefinite systems. This is very closely related to the situation with Newton's method.

The definition of the search direction given by (2) is only satisfactory if  $G^{(k)}$  is positive definite. An indefinite  $G^{(k)}$  allows the possibility of  $p$  not being a descent direction and this may result in convergence to a nonoptimal point. In the context of minimization, it is preferable not to solve the system (2) if  $G^{(k)}$  is indefinite. In this case it is better to define  $p$  as the solution of a neighbouring positive definite system

$$\bar{G}^{(k)}p = -g^{(k)}, \quad \bar{G}^{(k)} = G^{(k)} + E.$$

A method for computing  $\bar{G}^{(k)}$  when matrix factorizations are feasible is to compute the modified Cholesky factorization of  $G^{(k)}$  (Gill and Murray (1974)). The idea of the Gill-Murray algorithm is to increase the diagonal elements of  $G^{(k)}$  during the factorization so that the diagonal elements of the factorization are positive and the subdiagonal elements are bounded. An important feature of the Gill-Murray algorithm is the ability to detect that  $G^{(k)}$  is not sufficiently positive definite and to compute a satisfactory descent direction nevertheless.

A straightforward application of the linear conjugate-gradient method would not have this property. Moreover, the linear conjugate-gradient algorithm is numerically unstable when applied to an indefinite system. To overcome these difficulties, Nash (1984) has derived a modified linear conjugate-gradient algorithm via the Lanczos

algorithm (hereafter referred to as the modified-Lanczos algorithm) which has many of the properties of the Gill-Murray algorithm described above. If the matrix  $G^{(k)}$  is sufficiently positive definite, it is identical to the standard linear conjugate-gradient algorithm. A brief description of the modified-Lanczos algorithm appears in § 2.

Our main interest in this paper is the choice of a preconditioning strategy for a truncated-Newton method. In such a method, we solve a sequence of linear systems of the form (2), whose coefficient matrices  $G^{(k)}$  will converge to  $G(x^*)$  as the solution to the minimization problem is approached. Because of this convergence, it is possible to take advantage of earlier computations to make subsequent linear systems easier to solve, and hence to improve the efficiency of the overall algorithm. In large problems where it is expensive to compute information, it is especially important to make as much use as possible of every computed quantity. This is generally accomplished by using current information to precondition future iterations.

Preconditioning is such a powerful and general idea that there exist preconditioned versions of almost every known numerical algorithm, both direct and iterative. Direct algorithms often use preconditioning to reduce the error in the computed solution. One common example of this is the use of column scaling in Gaussian elimination (see, for example, Wilkinson (1965, Chap. IV)). Iterative methods generally use preconditioning to accelerate convergence, although they may also be concerned with the condition of the problem. One of the best known and best understood examples of this is the generalized (i.e. preconditioned) linear conjugate-gradient algorithm of Concus, Golub and O'Leary (1976).

To conclude this section, we give here a description of a truncated-Newton method in algorithmic form. The details of the methods used to iteratively solve the Newton equations and to precondition the algorithm will be given later.

#### TRUNCATED-NEWTON ALGORITHM.

TN0. Given  $x^{(0)}$ , some initial approximation to  $x^*$ . Set  $k=0$ .

TN1. If  $x^{(k)}$  is a sufficiently accurate approximation to the minimizer of  $F$ , terminate the algorithm.

TN2. Approximately solve the Newton equations (2) using some iterative algorithm with preconditioning  $M^{(k)} \approx G^{(k)}$ . (See §§ 2 and 3.)

TN3. With the search direction  $p$  computed in step TN2, find  $\alpha > 0$  such that  $F(x^{(k)} + \alpha p) < F(x^{(k)})$ . (Line search; see below.)

TN4. Set  $x^{(k+1)} = x^{(k)} + \alpha p$ ,  $k = k + 1$ . Go to step TN1.

For the purposes of this paper, we will assume that a modified-Lanczos algorithm (see below) will be used in Step TN2 to approximately solve the Newton equations. In step TN3, the line search,  $F(x)$  must be "sufficiently decreased" in order to guarantee convergence to a local minimum. One approach is to ensure that the search direction  $p$  is a descent direction ( $p^T g^{(k)} < 0$ ), that  $|g(x^{(k)} + \alpha p)^T p| \leq -\eta p^T g^{(k)}$  with  $0 \leq \eta < 1$ , and that  $F(x^{(k)}) - F(x^{(k)} + \alpha p) \geq -\mu p^T g^{(k)}$  where  $0 < \mu \leq \frac{1}{2}$ ,  $\mu < \eta$ . By choosing  $\mu$  small (say  $10^{-4}$ ), almost any  $\alpha$  that satisfies the first condition will also satisfy the second. The step-length  $\alpha$  can be computed using safeguarded polynomial interpolation (Gill and Murray (1979)).

**2. The modified-Lanczos method.** The general form of the modified-Lanczos algorithm is outlined below. If the Hessian matrix  $G$  is positive definite, this method is equivalent to the linear conjugate-gradient algorithm of Hestenes and Stiefel (1952). For more complete details, refer to Nash (1984).



Assume temporarily that  $G$  is positive definite. Recall that we are iteratively solving

$$(3) \quad Gp = -g.$$

We use the Lanczos algorithm (Lanczos (1950)) to compute a tridiagonal matrix that is an orthogonal projection of  $G$ . At stage  $q$  of the algorithm:

$$(4) \quad V_q^T G V_q = T_q, \quad V_q^T V_q = I,$$

where  $V_q$  is an  $n$  by  $q$  orthogonal matrix, and  $T_q$  is a  $q$  by  $q$  tridiagonal matrix. The tridiagonal matrix  $T_q$  is factored into its Cholesky factors:

$$T_q = L_q D_q L_q^T,$$

where  $D_q$  is diagonal with positive diagonal entries, and  $L_q$  is lower bidiagonal with ones on the main diagonal (this factorization is only possible if  $T_q$  is positive definite). This factorization is then used to compute  $p_q$  (the  $q$ th approximation to the solution of (3)):

$$T_q y_q = L_q D_q L_q^T y_q = (-V_q^T g), \quad p_q = V_q y_q.$$

(Paige and Saunders (1975) have derived iterative formulas for  $p_q$  based on this derivation.) At each inner iteration  $q$ , the direction  $p_q$  is tested to see if it "adequately" solves the Newton equations (3); if so, the inner iteration is *truncated*, and the search direction  $p$  is defined as  $p_q$  (see § 4 for details). The sequence of iterates  $p_q$  is the same as that generated by the linear conjugate-gradient algorithm, if we choose  $V_1 = (g/\|g\|_2)$ .

The linear conjugate-gradient algorithm can only be safely used when  $G$  is positive definite, whereas the Lanczos algorithm only requires that  $G$  be symmetric. The above derivation will enable us to adapt the linear conjugate-gradient algorithm to indefinite systems, as we now indicate.

If  $g$  contains a component of the negative eigenspace of  $G$ , then indefiniteness in  $G$  will ultimately show up in  $T_q$  (for  $q = n$ , (4) defines an orthogonal similarity transformation). In fact, due to the properties of the Lanczos algorithm, it will show up fairly early (Parlett (1980)). O'Leary (1982) has suggested applying the modified-Cholesky factorization of Gill and Murray (1974) to  $T_q$  in this case. However, this factorization requires information about the complete matrix  $T_n$  in order to ensure stability, information not available to this iterative algorithm.

Because  $T_q$  is iteratively generated, and  $T_{q-1}$  is a principal submatrix of  $T_q$ , it is possible to determine exactly the stage  $q$  at which  $T_q$  becomes indefinite. If this occurs, Nash (1984) suggests boosting the diagonal elements of the lower  $2 \times 2$  diagonal block so that the resulting matrix  $\bar{T}_q$  is positive definite. Because only this  $2 \times 2$  is modified, the iterative nature and the low storage requirements are unchanged. In addition, the size of any diagonal modification is bounded by  $3(\delta + \gamma_q + \zeta_q)$  where  $\gamma_q$  and  $\zeta_q$  are the largest (in absolute value) diagonal and off-diagonal elements of  $T_q$ , and  $\delta$  is a tolerance for zero (used to bound  $T_q$  away from singularity).

**2.1. Properties of the search direction.** Even if the Hessian is indefinite, the approximate solutions  $p_q$  of the Newton equations will be descent directions for the minimization algorithm, i.e.  $p_q^T g < 0$  for  $q > 0$ . If  $\bar{T}_q$  denotes the (possibly modified) tridiagonal matrix computed above, then

$$g^T p_q = -g^T V_q \bar{T}_q^{-1} V_q^T g < 0$$

if  $V_q^T g \neq 0$ , since  $\bar{T}_q$  is positive definite by construction. Since  $v_1$ , the first column of  $V_q$ , will be chosen as a nonzero multiple of  $M^{-1}g$  for some positive definite matrix  $M$ ,  $V_q^T g$  will be nonzero, and hence  $p_q$  will be a descent direction as desired.

Although necessary to guarantee the convergence of the algorithm, the fact that  $p$  is a descent direction is not enough to ensure that it is an effective search direction. It should also be well-scaled, i.e. a unit step along  $p$  should approximate the minimum of the function in that direction. Near  $x^*$ , this will be true for Newton's method, but cannot be guaranteed for nonlinear conjugate-gradient methods. However, regardless of how many modified-Lanczos iterations are used to compute the search direction  $p$ , a truncated-Newton method will generally give a well-scaled search direction, in the sense described below.

If the line-search procedure described in § 1 is used, the primary test (for an approximate minimum along the direction  $p$ ) is  $|g(x + \alpha p)^T p| \leq -\eta g^T p$ , where  $0 \leq \eta < 1$ . Assuming that  $V_q^T G V_q$  is positive definite, setting  $\alpha = 1$  (in the hope of a well-scaled direction), and using a Taylor series expansion, we obtain

$$\begin{aligned} p_q^T g(x + p_q) &= -g^T V_q (V_q^T G V_q)^{-1} V_q^T g \\ &\quad + g^T V_q (V_q^T G V_q)^{-1} (V_q^T G V_q) (V_q^T G V_q)^{-1} V_q^T g + O(\|g\|^3) \\ &= O(\|g\|^3). \end{aligned}$$

This final expression, representing the cubic remainder term in the Taylor series, will be small when  $x^{(k)}$  is near to  $x^*$ , or when  $F(x)$  is approximated well by a quadratic function. In these cases, we can expect that the search direction from the truncated-Newton method will be well-scaled, even after only one inner iteration. (See also Dembo and Steihaug (1983).)

**2.2. Preconditioning.** If a matrix  $M$  is available such that  $M \approx G$ , then the modified-Lanczos algorithm can take advantage of this information. The algorithm is applied (implicitly) to the equivalent system of linear equations

$$(M^{-1/2} G M^{-1/2}) M^{1/2} p = -M^{-1/2} g.$$

The number of iterations required to solve this transformed system is equal to the number of distinct eigenvalues of  $M^{-1} G$ . In addition, ignoring this finite-termination property, the algorithm converges linearly with rate  $(\kappa^{1/2} - 1)/(\kappa^{1/2} + 1)$ , where  $\kappa$  is the condition number of  $M^{-1} G$  in the 2-norm. We aim to choose  $M$  so that  $\kappa(M^{-1} G)$  is small, and so that  $M^{-1} G$  has fewer distinct eigenvalues than  $G$ , thus making the system of equations easier to solve. In practice, the matrix  $M^{-1/2}$  is not formed; all that is required is that a system of equations of the form

$$M y = c$$

be solved at each step. For details of these results, see Concus, Golub and O'Leary (1976).

**2.3. Matrix/vector products.** At each iteration, the Lanczos algorithm requires the computation of a matrix/vector product  $Gv$  involving the Hessian matrix  $G$ . However, the matrix  $G$  is not required explicitly. If  $G$  is explicitly available, these matrix/vector products can be formed directly. If  $G$  is sparse, a finite-difference approximation to  $G$  could be formed and used to compute them (see Thapa (1980)). Otherwise,  $Gv$  could be approximated by finite-differencing along the gradient  $g$ :

$$G(x)v \approx \frac{g(x + hv) - g(x)}{h}$$

for some suitably chosen small value of  $h$  (see, for example, Gill and Murray (1974), O'Leary (1982)).

**3. Preconditioning strategies.** With truncated-Newton methods, there are two principal ways in which a preconditioning strategy can be selected. A basic preconditioning might be chosen using the formulas for some low-memory nonlinear algorithm; this is the subject of § 3.1. Secondly, this nonlinear algorithm might be further preconditioned by some scaling of the variables. This is the subject of § 3.2. In either case, our goal is to develop a preconditioning operator dynamically, as the problem is being solved, and not to rely on a priori information.

**3.1. Preconditioning based on a nonlinear algorithm.** A truncated-Newton algorithm operates by using some iterative algorithm to approximately solve a sequence of equations of the form

$$G^{(k)}p = -g^{(k)}.$$

As the solution is approached, it might be hoped that information gained solving equation ( $k$ ) might assist in solving equation ( $k+1$ ). This information is generally used by forming, either explicitly or implicitly, a matrix  $M \approx G^{(k+1)}$ . The better  $M$  approximates  $G^{(k+1)}$ , the better the preconditioning strategy will be (see § 2.2). In order to use  $M$  within the modified-Lanczos algorithm,  $M$  must be positive definite, and linear systems involving  $M$  should be "easy" to solve. For example, the matrix  $M$  might be diagonal or in factored form.

It is possible to design preconditioning strategies by exploiting ideas from other minimization methods. Most nonlinear optimization algorithms can be viewed as computing a search direction by solving, possibly implicitly, a system of linear equations

$$Bp = -g$$

with some operator  $B$ , where  $B$  is an approximation to the Hessian  $G$ . By applying the formulas for the nonlinear method to any vector (instead of  $-g$ ), we implicitly define a preconditioning matrix  $M$ .

The optimal choice would be  $M = G^{(k+1)}$  (Newton's method) since the inner iteration would then converge instantly; however, the costs in storage and computation would be prohibitive. Setting  $M = I$ , i.e. using an unpreconditioned algorithm, corresponds to preconditioning with the steepest-descent operator; this is simple to use, but not particularly effective. As a compromise, Nash (1984) has suggested using the operator from a limited-memory quasi-Newton method, which is inexpensive to use, and yet still effective at improving the performance of the inner algorithm.

The class of limited-memory quasi-Newton methods (see Gill and Murray (1979)) define the search direction as a linear combination of the gradient vector and a subset of the previous search directions. They generalize nonlinear conjugate-gradient algorithms, and are suitable for problems in which the Hessian cannot be stored.

These methods derive their name from the class of quasi-Newton methods for unconstrained optimization. The direction of search for a quasi-Newton method can be defined as

$$p = -H_k g^{(k)},$$

where  $H_k$  is an  $n \times n$  matrix which is stored explicitly and is an approximation to the inverse Hessian matrix. After computing the change in  $x$ ,  $s_k \equiv x^{(k+1)} - x^{(k)}$  and the corresponding change in the gradient vector,  $y_k \equiv g^{(k+1)} - g^{(k)}$ , the approximate Hessian is updated to include the new curvature information obtained during the  $k$ -th iteration. For example, the BFGS formula for  $H_{k+1}$  is given by

$$(5) \quad H_{k+1} = H_k + \frac{(s_k - H_k y_k) s_k^T + s_k (s_k - H_k y_k)^T}{y_k^T s_k} - \frac{(s_k - H_k y_k)^T y_k}{(y_k^T s_k)^2} s_k s_k^T$$

(see Dennis and Moré (1977)). If exact linear searches are made and  $F$  is a positive-definite quadratic function, the matrix  $H_{k+1}$  satisfies the so-called quasi-Newton condition for  $k$  pairs of vectors  $\{s_j, y_j\}$ , i.e.,

$$s_j = H_{k+1}y_j, \quad j = 1, 2, \dots, k.$$

In this case, if the Hessian of  $F$  is  $G$ , then  $Gs_j = y_j$  and consequently

$$s_j = HGs_j,$$

and the matrix  $HG$  has  $k$  unit eigenvalues with eigenvectors  $\{s_j\}$ .

Limited-memory quasi-Newton methods define the direction of search as  $-Hg^{(k)}$ ; the matrix  $H$  is never stored explicitly; rather, only the vectors  $\{s_j, y_j\}$  that define the rank-one corrections are retained (see Shanno (1978), Gill and Murray (1979), and Nocedal (1980)).

Different methods can be developed by varying the number of vectors  $\{s_j, y_j\}$  stored and the choice of quasi-Newton updating formula. For example, if we define the matrix  $H$  to be the identity matrix updated by one iteration of the BFGS formula (5), and if exact line searches are performed, the algorithm will be equivalent to the Fletcher-Reeves nonlinear conjugate-gradient method.

When no preconditioning is used, the first linear iterate will be a multiple of the steepest-descent direction, which is often a poor approximation to the Newton direction. Preconditioning with an effective nonlinear algorithm offers the hope that the first iterate will approximate the Newton direction quite well, and that an adequate search direction can be computed using only a few inner iterations.

**3.2. Diagonal scaling of the variables.** Nonlinear minimization algorithms have been found to work more efficiently if the variables are properly scaled. In part, this means that a unit step along the search direction will approximate the minimizer of the function in that direction. It also implies that the tolerances for the algorithm have the correct scaling; this is a factor even for the more scale-invariant algorithms such as Newton's method. One way of achieving this is through a diagonal scaling matrix. In this context, the inverse of this diagonal matrix will be used as the initial approximation to the matrix  $H$  of § 3.1.

There is some theoretical evidence to indicate that, among diagonal scalings, the most effective strategy will be to approximate the diagonal of the Hessian. Forsythe and Straus (1955) have shown that if the Hessian matrix  $G$  is two-cyclic, then the diagonal of  $G$  is the optimal diagonal preconditioning. This assumption is valid for many problems arising in partial differential equations. Also, in the general case, van der Sluis (1969) has proven that preconditioning with the diagonal of  $G$  will be nearly optimal, in the sense that the condition number (in the 2-norm) of  $G$  preconditioned by its diagonal will be at most  $n$  times as large as the condition number of the optimally diagonally preconditioned  $G$ . Thus, estimating the diagonal of  $G$  should be effective for all problems.

The sample scaling strategies derived here will be based on quasi-Newton approximations to the diagonal of the Hessian matrix. It is possible to use the direct form of the BFGS formula (5) to approximate the diagonal of  $G$ . Here we approximate  $G$  by a sequence of matrices  $B_q$ , rather than approximating  $G^{-1}$  by matrices  $H_q$ .

Because the linear conjugate-gradient algorithm is equivalent to the BFGS algorithm (when applied to the same quadratic objective function with  $B_0 = I$ ), it is possible to show that  $B_n = G$ , if  $G$  is positive definite and if the iteration does not terminate prematurely (see Nazareth (1979)). Thus, if we were able to update only the

diagonals of  $B$ , at the end of  $n$  steps we would have the exact values for the diagonal elements of  $G$ .

To develop this diagonal update, we will ignore the nonlinear algorithm for the moment, and concentrate our attention on one instance of the linear conjugate-gradient method. We are attempting to minimize the quadratic function

$$\phi(p) = \frac{1}{2}p^T Gp + p^T c$$

and hence

$$g(p) = \nabla \phi(p) = Gp + c = -r(p),$$

where  $r(p)$  is the residual at  $p$ . The linear conjugate-gradient algorithm is initialized with  $p_0 = 0$ , and at the  $q$ th iteration, the next estimate of the solution is computed as

$$p_{q+1} = p_q + \alpha_q u_q,$$

where  $u_q$  is the search direction and  $\alpha_q$  is the step-length.

The BFGS algorithm computes the (same) search direction using the formula

$$(6) \quad B_q u_q = -g_q,$$

where  $g_q \equiv g(p_q)$ . If an exact line-search is used, the step-length for the BFGS algorithm is that same as that for the linear conjugate-gradient algorithm. Under the assumptions that  $p_0 = 0$ ,  $B_0 = I$ , and that the new approximate Hessian  $B_{q+1}$  is computed using the direct form of the BFGS formula (5)

$$(7) \quad B_{q+1} = B_q - \frac{1}{s_q^T B_q s_q} (B_q s_q)(B_q s_q)^T + \frac{1}{y_q^T s_q} y_q y_q^T,$$

both algorithms compute the same estimates of the solution at every stage.

It is possible to adapt (7) so that only the diagonal of the update need be computed. Using (6) and

$$s_q = p_{q+1} - p_q = \alpha_q u_q,$$

we can conclude that

$$(8) \quad B_q s_q = -\alpha_q g_q.$$

The other important fact is

$$(9) \quad y_q = g_{q+1} - g_q = \alpha_q G u_q.$$

If we incorporate (8) and (9) in (7), we obtain

$$(10) \quad B_{q+1} = B_q - \frac{1}{u_q^T r_q} r_q r_q^T + \frac{1}{u_q^T (G u_q)} (G u_q)(G u_q)^T.$$

(These quantities are all computed within the conjugate-gradient algorithm.) Using (10), any individual element of  $B_q$  can be individually updated. However, when used to compute a scaling matrix, only the diagonal of  $B_q$  will be formed.

When the linear conjugate-gradient algorithm is used in its standard form, (10) is quite adequate. However, using instead the preconditioned modified-Lanczos algorithm (see § 2) creates two further problems. First, in practice, a new scaling matrix will be generated using an iteration preconditioned by some operator  $M$ . In this case, the BFGS algorithm should be initialized with  $B_0 = M$ . To see this, replace  $G$  by  $M^{-1/2} G M^{-1/2}$  in the above derivation.

A second problem arises because the linear conjugate-gradient algorithm is implicitly implemented using the modified-Lanczos algorithm: only constant multiples of the search direction  $u_q$  and the residual  $r_q$  are computed. These multiplicative factors do not affect the final term in (10), since the factors enter equally into the numerator and the denominator. The other rank-one matrix is affected. However, the true residual can be computed using  $r_q = \alpha_q \tilde{v}_q$ , where  $\tilde{v}_q$  is the unnormalized current Lanczos vector (Parlett (1980)). This leaves only the inner product  $u_q^T r_q$ . Using the recurrence relation for the search direction  $u_q$ , and the fact that the residuals are  $M$ -orthogonal, it can be shown that

$$u_q^T r_q = r_q^T M^{-1} r_q = \alpha_q^2 \tilde{v}_q^T M^{-1} \tilde{v}_q.$$

Note that  $M^{-1} \tilde{v}_q$  is computed within the modified-Lanczos algorithm.

Because the Hessian matrix is not always positive definite, the modified-Lanczos algorithm alters the subproblem it is solving when it runs across evidence of indefiniteness. The preconditioning scheme is trying to approximate the diagonal of the actual Hessian matrix, and the preconditioning algorithm described above has the property of hereditary positive definiteness, so there is some question as to what should be done when the Hessian matrix is modified. We have chosen to omit the diagonal update whenever the matrix goes indefinite, in order to ensure that  $B_q$  remains positive definite.

Using (10) it is possible to compute any number of subdiagonals in addition to the main diagonal. Because this extension is so straightforward, the details will be omitted here.

An additional possibility is to use exact information about the diagonal of the Hessian either to precondition the linear algorithm or to initialize the linear preconditioning. Note, however, that even if matrix-vector products of the form  $Gv$  can be found, it may be inconvenient to compute  $G_{ii}$ . Also, away from the solution of the minimization problem, the matrix  $G$  may be indefinite, so that the diagonal of the Hessian may not define a positive definite preconditioning matrix. In that case, some rule for modifying negative diagonal elements would have to be derived.

**4. Numerical results.** In this section we compare the numerical behavior of three truncated-Newton algorithms with that of other methods. The methods tested are:

1. Algorithm PLMA—A two-step BFGS limited-memory quasi-Newton method with a simple diagonal scaling. PLMA is the most successful nonlinear conjugate-gradient-type method tested in the survey of Gill and Murray (1979).
2. Algorithm MNA—A modified Newton method using first and second derivatives (Gill and Murray (1974)).
3. Algorithm QNM—A quasi-Newton method using the full  $n$  by  $n$  BFGS update of the approximate Hessian matrix (Gill and Murray (1972)).
4. Algorithm TN—A truncated-Newton algorithm, implemented via the modified-Lanczos algorithm, and preconditioned with PLMA with the simple diagonal scaling replaced by the diagonal of (10).
5. Algorithm BTN—A (basic) truncated-Newton algorithm, implemented via the linear conjugate-gradient algorithm, and with no preconditioning strategy.
6. Algorithm PBTN—Algorithm BTN, but preconditioned using the diagonal of (10).

Eighteen problems are considered. Of these, 11 problems are of dimension 50 or less, and 7 problems are of dimension 100. The test examples may be separated into two classes. The first class contains problems whose Hessian matrix at the solution has clustered eigenvalues; the second contains problems whose Hessian matrix has an arbitrary eigenvalue distribution.

*Example 1.* Pen 1 (Gill, Murray and Pitfield (1972)).

$$F(x) = a \sum_{i=1}^n (x_i - 1)^2 + b \left( \sum_{i=1}^n x_i^2 - \frac{1}{4} \right)^2.$$

The solution varies with  $n$ , but  $x_i = x_{i+1}$ ,  $i = 1, \dots, n-1$ . All the runs were made with  $a = 1$ ,  $b = 10^{-3}$ . With these values, the Hessian matrix at the solution has  $n-1$  eigenvalues  $O(1)$  and one eigenvalue  $O(10^{-3})$ . The Hessian matrix is full and consequently, for large values of  $n$ , conjugate-gradient type methods are the only techniques available.

*Example 2.* Pen 2 (Gill, Murray and Pitfield (1972)).

$$F(x) = a \sum_{i=2}^n ((e^{x_i/10} + e^{x_{i-1}/10} - c_i)^2 + (e^{x_i/10} - e^{-1/10})^2) \\ + b \left( \left( \sum_{i=1}^n (n-i+1)x_i^2 - 1 \right)^2 + \left( x_1 - \frac{1}{5} \right)^2 \right)^2,$$

where  $c_i = e^{i/10} + e^{(i-1)/10}$  for  $i = 2, \dots, n$ . The solution varies with  $n$ , but  $x_i = x_{i+1}$  for  $i = 1, \dots, n-1$ . This example was also run with  $a = 1$  and  $b = 10^{-3}$ . For these values the Hessian matrix at the solution has  $n-2$  eigenvalues  $O(1)$  and two eigenvalues  $O(10^{-3})$ . The Hessian matrix is full.

*Example 3.* Pen 3 (Gill, Murray and Pitfield (1972)).

$$F(x) = a \left\{ 1 + e^{x_n} \sum_{i=1}^{n-2} (x_i + 2x_{i+1} + 10x_{i+2} - 1)^2 \right. \\ \left. + \left( \sum_{i=1}^{n-2} (x_i + 2x_{i+1} + 10x_{i+2} - 1)^2 \right) \left( \sum_{i=1}^{n-2} (2x_i + x_{i+1} - 3)^2 \right) \right. \\ \left. + e^{x_{n-1}} \sum_{i=1}^{n-2} (2x_i + x_{i+1} - 3)^2 \right\} \\ + \left( \sum_{i=1}^n (x_i^2 - n) \right)^2 + \sum_{i=1}^{n/2} (x_i - 1)^2.$$

At the minimum, this function has  $n/2$  eigenvalues  $O(1)$  and  $n/2$  eigenvalues  $O(10^{-3})$ . The Hessian matrix is full.

The remaining examples have arbitrary distributions of eigenvalues at the solution.

*Example 4.* Chebyquad (Fletcher (1965)).

$$F(x) = \sum_{i=1}^n f_i(x)^2,$$

where

$$f_i(x) = \int_0^1 T_i^*(x) dx - \frac{1}{n} \sum_{j=1}^n T_i^*(x_j), \quad i = 1, \dots, n,$$

and  $T_i^*(x)$  is the  $i$ th-order shifted Chebyshev polynomial. The Hessian matrix is full.

*Example 5.* GenRose. This function is a generalization of the well-known two-dimensional Rosenbrock function (Rosenbrock (1960)). For  $n > 2$ ,

$$F(x) = 1 + \sum_{i=2}^n (100(x_i - x_{i-1}^2)^2 + (1 + x_i)^2).$$

Our implementation of this function differs from most others in that  $F(x)$  is unity at the solution rather than zero. This modification ensures that the function cannot be

computed with unusually high accuracy at the solution and is therefore more typical of practical problems.

The next three examples arise from the discretization of problems in the calculus of variations. Similar problems arise in the numerical solution of optimal control problems. The general continuous problem is to find the minimum of the functional

$$J(x(t)) = \int_0^1 f(t, x(t), x'(t)) dt,$$

over the set of piecewise differentiable curves with the boundary conditions  $x(0) = a$ ,  $x(1) = b$ . If  $x(t)$  is expressed as a linear sum of functions that span the space of piecewise cubic polynomials then minimization of  $J$  becomes a finite-dimensional problem with a block tridiagonal Hessian matrix. The piecewise polynomials are assumed to be in  $C^1$ , and equally spaced knots are used.

*Example 6.* Cal 1 (Gill and Murray (1973)).

$$J(x(t)) = \int_0^1 \{x(t)^2 + x'(t) \tan^{-1} x'(t) - \log(1 + x'(t)^2)^{1/2}\} dt,$$

with the boundary conditions  $x(0) = 1$ ,  $x(1) = 2$ .

*Example 7.* Cal 2 (Gill and Murray (1973)).

$$J(x(t)) = \int_0^1 \{100(x(t) - x'(t)^2)^2 + (1 - x'(t))^2\} dt,$$

with the boundary conditions  $x(0) = x(1) = 0$ .

*Example 8.* Cal 3 (Gill and Murray (1973)).

$$J(x(t)) = \int_0^1 \{e^{-2x(t)^2}(x'(t)^2 - 1)\} dt,$$

with the boundary conditions  $x(0) = 1$ ,  $x(1) = 0$ .

*Example 9.* QOR (Toint (1978)).

$$F(x) = \sum_{i=1}^{50} \alpha_i x_i^2 + \sum_{i=1}^{33} \beta_i \left( d_i - \sum_{j \in A(i)} x_j + \sum_{j \in B(i)} x_j \right)^2,$$

where the constants  $\alpha_i$ ,  $\beta_i$ ,  $d_i$  and sets  $A(i)$  and  $B(i)$  are described in Toint's paper. This function is convex with a sparse Hessian matrix.

*Example 10.* GOR (Toint (1978)).

$$F(x) = \sum_{i=1}^{50} c_i(x_i) + \sum_{i=1}^{33} b_i(y_i),$$

where

$$c_i(x_i) = \begin{cases} \alpha_i x_i \log_e(1 + x_i), & x_i \geq 0, \\ -\alpha_i x_i \log_e(1 + x_i), & x_i < 0, \end{cases}$$

$$y_i = d_i - \sum_{j \in A(i)} x_j + \sum_{j \in B(i)} x_j$$

and

$$b_i(y_i) = \begin{cases} \beta_i y_i^2 \log_e(1 - y_i), & y_i \geq 0, \\ \beta_i y_i^2, & y_i < 0. \end{cases}$$

The constants  $\alpha_i$ ,  $\beta_i$ ,  $d_i$  and sets  $A(i)$  and  $B(i)$  are defined as in Example 9. This function is convex but there are discontinuities in the second derivatives.



*Example 11.* ChnRose (Toint (1978)).

$$F(x) = 1 + \sum_{i=2}^{25} (4\alpha_i(x_{i-1} - x_i^2)^2 + (1 - x_i)^2),$$

where the constants  $\alpha_i$  are those used in Example 9. The value of  $F(x)$  at the solution has been modified as in Example 5. The Hessian matrix is tridiagonal.

The starting points used were the following:

Start 1  $x^{(0)} = (0, 0, \dots, 0)^T$ .

Start 2  $x^{(0)} = \left( \frac{1}{n+1}, \frac{2}{n+1}, \dots, \frac{n}{n+1} \right)^T$ .

Start 3  $x^{(0)} = (1, -1, 1, -1, \dots)^T$ .

Start 4  $x^{(0)} = (-1, -1, \dots, -1)^T$ .

**4.1. Details of the algorithms.** All the routines are coded in double precision FORTRAN IV. The run were made on an IBM 370/168, for which the relative machine precision  $\varepsilon$  is approximately  $10^{-15}$ .

The truncated-Newton routines require the computation of matrix/vector products of the form  $G^{(k)}v$ . For routine TN with Examples 5-11, sparse finite-differencing techniques (Thapa (1980)) were used to approximate  $G^{(k)}$  at the beginning of each major iteration, and this approximation was used to compute the matrix/vector products. The difference parameter used here was  $\varepsilon^{1/2}$ , where  $\varepsilon$  is the machine precision. Elsewhere, the matrix/vector products were computed by differencing the gradient along the vector  $v$  (§ 2.3). Because our interest is in methods that do not require second derivatives, tests were not made using exact second-derivative information.

For all truncated-Newton algorithms, a fairly stringent criterion was used to terminate the modified-Lanczos iterations. Following Dembo and Steihaug (1983), the modified Lanczos iterations are terminated after  $n/2$  iterations, or if

$$\frac{\|r_q\|}{\|g^{(k)}\|} \leq \min \{1/k, \|g^{(k)}\|\},$$

where  $r_q$  is the  $q$ th residual of the linear system. This criterion forces the algorithm to behave like a conjugate-gradient algorithm near the beginning of the iteration and like Newton's method near the solution. We stress, however, that when second derivatives are not available, or the cost of the matrix/vector product  $G^{(k)}v$  is high, a criterion must be used that always leads to a small number of linear iterations. Because the computation of the search direction can be degraded by loss of orthogonality, at most  $n/2$  modified Lanczos iterations were allowed at each major step.

Each problem was solved using three values of  $\eta$ , the step-length accuracy (see § 1); these values were 0.25, 0.1, and 0.001. Each algorithm requires two additional user-specified parameters. The first ( $\lambda$ ) limits the change in  $x$  at each iteration (the quantity  $\|x^{(k+1)} - x^{(k)}\|_2$ ). The value of  $\lambda$  was set at 10 for all problems to avoid overflow during the computation of the objective function. The second parameter is an estimate of the value of the objective function at the solution and is used to compute the initial step for the step-length algorithm. In each case, this parameter was set to the value of  $F(x)$  at the solution.

The results are contained in Tables 1-3. Each table entry refers to the iteration at which

$$F^{(k)} - F(x^*) < 10^{-5}(1 + |F(x^*)|),$$

where  $x^*$  is the solution.

TABLE 1  
(Reprinted from Nash [14].)

Function	$\eta$	PLMA	MNA	QNM	BTN	PBTN	TN	Finite-difference	Lanczos iterations	Totals
Cal 1 Start 1 $n = 50$	.25	194 366	7 9	162 191	16 978	8 227	10 18	60	181	78/199
	.1	204 401	6 11	89 214	17 1151	8 232	9 19	54	139	73/158
	.001	205 456	6 17	88 269	13 787	7 252	9 34	54	132	88/166
Cal 2 Start 1 $n = 50$	.25	64 106	4 4	28 52	8 133	7 104	7 8	42	61	50/69
	.1	61 118	4 4	28 54	8 133	7 104	7 8	42	61	50/69
	.001	60 123	4 6	28 67	8 135	7 107	7 10	42	61	52/71
Cal 3 Start 1 $n = 50$	.25	80 152	6 6	90 114	7 206	7 125	7 8	42	104	50/112
	.1	78 155	5 7	59 135	7 206	7 127	7 8	42	104	50/112
	.001	77 161	5 11	59 161	8 289	7 125	7 17	42	96	59/113
Cal 1 Start 1 $n = 100$	.25	423 819	not run	not run	26 3855	9 828	10 19	60	390	79/409
	.1	429 854	not run	not run	23 3214	8 570	10 25	60	484	85/409
	.001	416 905	not run	not run	26 3948	8 602	10 35	60	337	95/372
Cal 2 Start 1 $n = 100$	.25	112 204	not run	not run	8 256	7 172	8 9	48	108	57/117
	.1	107 206	not run	not run	8 256	7 172	8 9	48	108	57/117
	.001	113 228	not run	not run	8 259	8 199	8 11	48	111	59/122
Cal 3 Start 1 $n = 100$	.25	143 270	not run	not run	9 508	7 196	7 8	42	151	50/159
	.1	142 281	not run	not run	9 636	7 196	7 9	42	159	51/168
	.001	138 284	not run	not run	9 617	7 195	7 18	42	158	60/176
GenRose Start 2 $n = 50$	.25	108 201	62 202	128 287	33 499	42 584	31 75	93	255	168/330
	.1	119 263	66 257	118 323	32 518	36 469	33 99	99	249	198/348
	.001	119 330	88 392	118 412	35 614	37 468	34 143	102	252	245/395
GenRose Start 2 $n = 100$	.25	191 365	not run	not run	60 1150	73 1055	57 164	171	420	335/684
	.1	192 410	not run	not run	62 1153	72 1012	60 190	180	585	370/775
	.001	188 528	not run	not run	60 1197	63 1062	58 248	174	534	422/782
Chebyquad Start 2 $n = 20$	.25	38 75	29 121	32 65	10 104	10 87	7 16	37	37	53/53
	.1	33 71	24 116	28 67	9 96	9 105	8 17	51	51	68/68
	.001	33 90	30 161	28 92	11 164	10 129	9 29	61	61	90/90

For paired entries ( $n_1, n_2$ ),  $n_1$  is the number of major iterations required, and  $n_2$  is the number of function/gradient evaluations used.

TABLE 2

Function	$\eta$	PLMA	MNA	QNM	BTN	PBTN	TN	Finite-difference	Lanczos iterations	Totals
Pen 1 Start 3 $n = 50$	.25	22 53	17 18	27 33	10 46	10 46	7 18	11	11	29/29
	.1	8 27	9 25	8 26	10 48	10 48	7 19	11	11	30/30
	.001	8 32	7 29	8 31	6 40	6 40	3 15	4	4	19/19
Pen 2 Start 3 $n = 50$	.25	52 118	17 17	134 242	14 89	10 60	11 35	29	29	64/64
	.1	28 76	9 31	99 322	13 86	10 64	12 42	36	36	78/78
	.001	15 71	6 26	73 341	12 118	9 67	12 57	38	38	95/95
Pen 3 Start 3 $n = 50$	.25	40 76	40 44	67 135	10 62	11 60	10 14	33	33	47/47
	.1	38 76	12 44	63 150	10 61	9 52	9 16	30	30	46/46
	.001	28 71	11 48	56 155	10 72	10 63	9 24	30	30	54/54
Pen 1 Start 3 $n = 100$	.25	17 40	not run	not run	2 11	2 11	2 9	2	2	11/11
	.1	2 9	not run	not run	2 11	2 11	2 9	2	2	11/11
	.001	2 10	not run	not run	2 12	2 12	2 10	2	2	12/12
Pen 2 Start 3 $n = 100$	.25	14 28	not run	not run	6 25	6 26	6 15	14	14	29/29
	.1	7 18	not run	not run	5 24	5 23	6 18	12	12	30/30
	.001	7 29	not run	not run	5 36	5 33	6 26	15	15	41/41
Pen 3 Start 3 $n = 100$	.25	49 85	not run	not run	13 79	11 75	11 24	41	41	65/65
	.1	48 94	not run	not run	13 89	11 75	11 26	41	41	67/67
	.001	35 83	not run	not run	11 91	11 89	11 34	43	43	77/77
QOR Start 1 $n = 50$	.25	14 29	3 3	23 39	5 29	5 25	6 7	48	18	55/25
	.1	14 29	3 3	13 27	5 29	5 25	6 7	48	18	55/25
	.001	14 29	3 3	13 27	5 29	5 25	6 7	48	18	55/25
GOR Start 1 $n = 50$	.25	14 71	5 5	29 59	7 74	7 77	7 9	56	51	65/60
	.1	14 76	5 5	29 59	7 74	7 77	7 10	56	51	66/61
	.001	42 97	5 7	29 72	6 61	7 85	7 17	56	50	73/67
ChnRose Start 4 $n = 25$	.25	40 82	15 28	48 97	10 88	11 121	11 23	33	54	56/77
	.1	37 76	16 48	46 122	10 94	11 91	11 25	33	50	58/75
	.001	43 119	12 47	47 164	11 104	1 4	14 56	42	70	98/126

For paired entries ( $n_1, n_2$ ),  $n_1$  is the number of major iterations required, and  $n_2$  is the number of function/gradient evaluations used.

TABLE 3

Total	PLMA	MNA	QNM	BTN	PBTN	TN	Finite-difference	Lanczos iterations	Totals
Smaller functions $n \leq 50$	1,944 4,276	541 1,755	1,895 4,604	383 7,217	353 4,275	347 910	1,505	2,446	2,415/3,356
Larger functions $n = 100$	2,775 5,750	not run	not run	367 17,427	331 6,614	307 916	1,147	3,717	2,063/4,633
Grand totals	4,719 10,026	--	--	750 24,644	684 10,889	654 1,826	2,652	6,163	4,478/7,989

For paired entries  $(n_1, n_2)$ ,  $n_1$  is the number of major iterations required, and  $n_2$  is the number of function/gradient evaluations used.

**4.2. Discussion of results.** With the exception of the results for TN, each entry is a pair of numbers: the first is the number of major iterations; the second is the number of function/gradient evaluations required to solve the problem (for BTN and PBTN, this reflects both the line search and the matrix/vector products). For TN, more detailed results are given. The first pair of numbers gives the total number of iterations, and the number of function/gradient evaluations used in the line search. The finite-difference column records the number of gradient evaluations used to compute the matrix/vector products. The next column is the total number of modified Lanczos iterations (each iteration will normally be dominated by the cost of the matrix/vector product, comparable to a gradient evaluation). The final column combines the line-search cost with the inner-iteration cost to give a measure of the total cost of the minimization; two totals are given: the first combines the line-search costs with finite-differencing costs, and the second with the inner-iteration costs.

We first compare the truncated-Newton algorithms among themselves. It is clear that Algorithm TN is superior to the other two. This is not surprising due to the more elaborate preconditioning strategies that it uses. Based on the total number of iterations required, TN is only marginally better than the other two routines. But based on the number of function/gradient evaluations, there is a clear difference. Even without taking advantage of sparsity, PBTN is 36% slower than TN, and BTN is over three times as slow. A comparison of PBTN with BTN indicates the improvement even simple preconditioning strategies can make to a truncated-Newton algorithm. The two routines are identical, except that PBTN has a diagonal scaling as a preconditioner. This addition is inexpensive (three extra vectors are needed), but offers a better than 50% improvement in performance (based on the total number of function/gradient evaluations).

To compare truncated-Newton algorithms with other methods, we will use the results for Algorithm TN. In the following, results for the Newton algorithm MNA and the quasi-Newton method QNM are only available for the smaller functions ( $n \leq 50$ ). Tests with the larger functions ( $n = 100$ ) were not made due to the storage and computational costs. The total number of function/gradient evaluations (Table 3) will be the primary factor for comparison.

A comparison of TN with the limited-memory quasi-Newton algorithm PLMA shows that TN is 50% better if sparsity of the Hessian is taken into account, and 20% better otherwise (i.e., if each Lanczos iteration requires a gradient evaluation to approximate the matrix/vector product). This comparison is important, since these two classes of methods have comparable storage and operation counts, and they are the only practical methods for solving many large-scale problems.

A comparison of TN with the quasi-Newton method QNM on the smaller test functions indicates that TN is 50% better if sparsity is exploited, and 30% better otherwise. QNM, unlike TN, requires matrix manipulation, and hence has higher storage and operation counts than TN. Both algorithms only require first derivative information.

A comparison of TN with the modified Newton method MNA, again on the smaller test functions, shows that MNA is 30%–50% better than TN, depending on whether sparsity is exploited. However, MNA computes, stores and factors the Hessian matrix, and this is not reflected in the scores for MNA. For this reason, a further comparison is suggested, using the “TN” rather than the “totals” column in Table 3. The “TN” column does not reflect the cost of the matrix/vector products, i.e., the “second-derivative costs” of the truncated-Newton algorithm. From this point of view, TN is twice as efficient as MNA. This is surprising, since the truncated-Newton method

is a compromise on Newton's method designed to enable the solution of large-scale problems.

The results of comparisons for individual functions are not always so remarkable. For many of the functions in Table 2, the Hessian has clustered eigenvalues. All of these problems tend to be easy to solve, and there are few striking differences in performance. The remaining problems (Table 1) have more arbitrary eigenvalue distributions, and are considerably harder to solve. Here, the simple truncated-Newton Algorithm BTN has particular difficulties (Cal 1,  $n = 50, 100$ ). Even Newton's method (MNA) appears to struggle with some functions (GenRose,  $n = 50$ ); and performs worse than any other routine in one case (Chebyquad,  $n = 20$ ). For these two functions, the Hessian is frequently indefinite, suggesting that complete modified factorizations are not always an effective treatment for nonconvex functions.

**Acknowledgments.** The author would like to thank his thesis advisors Philip Gill, Gene Golub, and Walter Murray for their many helpful suggestions. Also, thanks to Mukund Thapa for kindly providing the subroutines for computing the sparse Hessian matrices of the test examples.

#### BIBLIOGRAPHY

- [1] P. CONCUS, G. 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. Bunch and D. Rose, eds., Academic Press, New York, 1976, pp. 309-332.
- [2] R. S. DEMBO AND T. STEihaug, *Truncated-Newton algorithms for large-scale unconstrained optimization*, Math. Prog., 26 (1983), pp. 190-212.
- [3] J. E. DENNIS AND J. J. MORÉ, *Quasi-Newton methods, motivation and theory*, SIAM Rev., 19 (1977), pp. 46-89.
- [4] R. FLETCHER, *Function minimization without evaluating derivatives—a review*, Comput. J., 8 (1965), pp. 33-41.
- [5] G. E. FORSYTHE AND E. G. STRAUS, *On best conditioned matrices*, Proc. Amer. Math. Soc., 6 (1965), pp. 340-345.
- [6] N. K. GARG AND R. A. TAPIA, *QDN: A variable storage algorithm for unconstrained optimization*, Department of Mathematical Sciences Report, Rice Univ., Houston, 1980.
- [7] P. E. GILL AND W. MURRAY, *Quasi-Newton methods for unconstrained optimization*, J. Inst. Maths. Applics., 9 (1972), pp. 91-108.
- [8] ———, *The numerical solution of a problem in the calculus of variations*, in Recent Mathematical Developments in Control, D. J. Bell, ed., Academic Press, New York, 1973, pp. 97-122.
- [9] ———, *Newton-type methods of unconstrained and linearly constrained optimization*, Math. Prog., 17 (1974), pp. 311-350.
- [10] ———, *Conjugate-gradient methods for large-scale nonlinear optimization*, Report SOL 79-15, Operations Research Dept., Stanford Univ., Stanford, CA, 1979.
- [11] P. E. GILL, W. MURRAY AND R. A. PITFIELD, *The implementation of two revised quasi-Newton algorithms for unconstrained optimization*, Report NAC 11, National Physical Laboratory, England, 1972.
- [12] M. HESTENES AND E. STIEFEL, *Methods of conjugate-gradients for solving linear systems*, J. Res. NBS, 49 (1952), pp. 409-436.
- [13] C. LANCZOS, *An iteration method for the solution of the eigenvalue problem of linear differential and integral operators*, J. Res. NBS, 45 (1950), pp. 255-282.
- [14] S. G. NASH, *Newton-type minimization via the Lanczos algorithm*, SIAM J. Numer. Anal., 21 (1984), pp. 770-778.
- [15] L. NAZARETH, *A relationship between the BFGS and conjugate-gradient algorithms and its implications for new algorithms*, SIAM J. Numer. Anal., 16 (1979), pp. 794-800.
- [16] D. P. O'LEARY, *A discrete Newton algorithm for minimizing a function of many variables*, Math. Prog., 23 (1983), pp. 20-33.
- [17] C. C. PAIGE AND M. A. SAUNDERS, *Solution of sparse indefinite systems of linear equations*, SIAM J. Numer. Anal., 12 (1975), pp. 617-629.
- [18] B. N. PARLETT, *The Symmetric Eigenvalue Problem*, Prentice-Hall, Englewood Cliffs, NJ, 1980.

- [19] M. J. D. POWELL AND P. L. TOINT, *On the estimation of sparse Hessian matrices*, SIAM J. Numer. Anal., 16 (1979), pp. 1060–1074.
- [20] H. H. ROSENBROCK, *An automatic method for finding the greatest or least value of a function*, Comput. J., 3 (1960), pp. 175–184.
- [21] D. F. SHANNO, *Conjugate gradient methods with inexact searches*, Math. Oper. Res., 3 (1978), pp. 244–256.
- [22] A. H. SHERMAN, *On Newton-iterative methods for the solution of systems of nonlinear equations*, SIAM J. Numer. Anal., 15 (1978), pp. 755–771.
- [23] M. THAPA, *Optimization of unconstrained functions with sparse Hessian matrices*, Ph.D. thesis, Dept. Operations Research, Stanford Univ., Stanford, CA, 1980.
- [24] P. L. TOINT, *Some numerical results using a sparse matrix updating formula in unconstrained optimization*, Math. Comp., 32 (1978), pp. 839–851.
- [25] A. VAN DER SLUIS, *Condition numbers and equilibration of matrices*, Numer. Math., 14 (1979), pp. 14–23.
- [26] J. H. WILKINSON, *The Algebraic Eigenvalue Problem*, Oxford Univ. Press, London, 1965.

## THE SOLUTION OF NONLINEAR ELLIPTIC DIRICHLET PROBLEMS ON RECTANGLES BY ALMOST GLOBALLY CONVERGENT INTERVAL METHODS\*

HARTMUT SCHWANDT†

**Abstract.** The use of interval arithmetic offers the possibility to develop methods for the solution of systems of nonlinear equations that converge to the solution under relatively weak conditions, provided an initial inclusion is known. In the present paper we describe methods for discretizations of nonlinear elliptic equations, in particular for Dirichlet problems of the type  $(au_x)_x + (bu_y)_y = f(x, y, u)$  where  $a \equiv a(x)$  and  $b = b(y)$  or  $a \equiv a(x, y)$ ,  $b \equiv b(x, y)$ . The interval arithmetic enables us to combine a Newton-like interval method with the concept of fast direct solvers. The convergence to the solution can be ensured by auxiliary methods or by monotonicity arguments. We compare some of the possible variants with the generalized CG method and the nonlinear block SOR method.

**Key words.** nonlinear Dirichlet problems, fast direct solvers, interval arithmetics, Newton-like methods

**1. Introduction.** In the last years a number of very efficient methods have been developed for the solution of large linear systems resulting from discretizations of partial differential equations. Their principles of construction can often be carried over to the nonlinear case. But the methods developed in this context are mostly only locally convergent. There are only a few papers describing efficient globally convergent methods. The following two seem to be the most interesting ones. In 1975 Hageman and Porsching developed a numerical strategy to guarantee global convergence for the nonlinear block SOR method [5]. In 1978 Concus, Golub and O'Leary combined the concept of fast direct solvers with a generalized CG method [4]. Global convergence can be ensured by special convergence safeguards.

The possibilities of interval arithmetic lead to another approach for obtaining methods of practical relevance.

In [8] we introduced an interval arithmetic method to solve systems of nonlinear equations resulting from discretizations of the Poisson equation  $\Delta u = f(u)$  on the unit square or rectangles. The use of interval arithmetic enabled us to guarantee convergence to the solution provided an initial inclusion is known and to develop a reduction method that reduces the arithmetic work per step. We combined the ideas of Newton-like interval methods and fast Poisson solvers to define a rapidly converging method. In [2] the following interval Newton method was introduced:

$$(1.1) \quad x^{k+1} := \{m^k - \text{IGA}(f'(x^k), f'(m^k))\} \cap x^k, \quad k := 0(1)\infty$$

to solve a system  $f(x) = \varphi$  where  $f: \mathbb{R}^n \rightarrow \mathbb{R}^n$ ,  $x \in \mathbb{R}^n$  and  $\varphi = (0)_{i=1}^n \cdot f(x^k)$  denotes an interval extension of the Jacobi matrix  $f'(\cdot)$  of  $f$ , i.e.  $f'(x^k)$  is an  $n \times n$  matrix with interval coefficients. Starting with an interval vector  $x^0$  including a solution of the above system, the method yields a sequence of interval vectors  $x^k$ ,  $k \in \mathbb{N}_0$ , with  $n$  interval components. The vectors  $m^k \in \mathbb{R}^n$  are chosen such that  $m^k \in x^k$ . We abbreviate the interval arithmetic Gauss algorithm by IGA. See § 2 and [2], [8] for the notation. In [8] we introduced the interval arithmetic reduction method IBU (see (2.6) of the present paper) which is suitable for a class of block tridiagonal interval matrices related to the numerical solution of the problems discussed in [8] and this paper. For some particular point matrices (i.e. matrices with real numbers as components) IBU

\* Received by the editors August 18, 1983, and in revised form February 21, 1984.

† Technische Universität Berlin, Fachbereich 3-Mathematik, 1 Berlin 12, West Germany.



degenerates into the Buneman algorithm. With the exception of some particular cases, the method IBU cannot be applied to  $f'(x^k)$ . We therefore included  $f'(x^k)$  by an interval matrix  $\hat{\mathcal{F}}(x^k)$  which has the form required by IBU by enlarging suitable coefficients of  $f'(x^k)$ . We then defined the

(1.2) *Newton-like method with reduction method* IBU (INB).

$$x^{k+1} := \{m^k - \text{IBU}(\hat{\mathcal{F}}(x^k), f'(m^k))\} \cap x^k \quad \text{with } m^k \in x^k, \quad k := 0(1)\infty.$$

In the present paper we extend the results of [8] in two directions. First of all, we “split” the matrix  $\hat{\mathcal{F}}(x^k)$  respectively  $f'(x^k)$  such that we obtain a matrix suitable for the application of IBU when solving problems that do not permit the direct use of INB. The second extension is a consequence of the convergence problem. In [8] we used an auxiliary method to force convergence if necessary. Here we test two similar possibilities. We also propose an alternative approach to guarantee convergence by using monotonicity arguments and particular choices of the  $m^k$ . In this case an auxiliary method is not necessary. Monotone convergence has been discussed by many authors, see [11], [7] e.g. By using interval arithmetic we obtain methods that converge under relatively weak conditions (e.g. no convexity conditions are required) and that produce iterates with easily computable matrices.

In this paper we use the notation introduced in [8]. Section 2 explains notation and some auxiliary results. In § 3 we introduce a matrix “splitting” suitable for the above mentioned applications and we formulate the iteration method. A convergence analysis for the version with an auxiliary method and the monotone versions follows. In § 4 we describe the iterative improvement of a given initial inclusion to be used as a starting vector. In § 5 we formulate our method for five-point discretizations of Dirichlet problems of the form  $(a(s)u_s)_s + (b(t)u_t)_t = f(u)$  or  $(a(s, t)u_s)_s + (b(s, t)u_t)_t = f(u)$  on rectangles. The method can be modified for other classes of equations by introducing appropriate matrix splittings and other methods to compute inclusions of the set of solutions of the “linear” systems in each iterative step. This will be the subject of a forthcoming paper.

The last section reports the results of some numerical experiments. We compared our method with the nonlinear generalized CG method as described in [4] and the nonlinear block SOR method as described in [5].

**2. Notation and auxiliary results.** Our notation follows closely that of [2]. Further details have been given in [8]. Instead of repeating them all, we refer to § 1 of [8] by adding an “A”, e.g. (A1.4). We denote again by  $a, \dots, z$  real numbers, by  $A, \dots, Z$  elements of the set  $I(\mathbb{R})$  of real compact intervals, by  $a, \dots, z$   $N$ -dimensional vectors, by  $a, \dots, z$  vectors with  $N$  interval components, by  $\mathcal{A}, \dots, \mathcal{Z}$  real matrices and by  $\mathcal{A}, \dots, \mathcal{Z}$  interval matrices. For the upper and lower bounds of intervals  $A$ , interval vectors  $a = (A_i)_{i=1}^N$  and interval matrices  $\mathcal{A} = (A_{ij})_{i=1, j=1}^{N, k}$  we introduce the notation  $A = [\underline{A}, \bar{A}]$ ,  $a = [\underline{a}, \bar{a}]$ ,  $\underline{a} = (\underline{A}_i)_{i=1}^N$ ,  $\bar{a} = (\bar{A}_i)_{i=1}^N$ ,  $\mathcal{A} = [\underline{\mathcal{A}}, \bar{\mathcal{A}}]$ ,  $\underline{\mathcal{A}} = (\underline{A}_{ij})_{i=1, j=1}^{N, k}$ ,  $\bar{\mathcal{A}} = (\bar{A}_{ij})_{i=1, j=1}^{N, k}$ .

We write block tridiagonal matrices with  $m^2$  blocks of the size  $l \times l$  as

$$(2.1) \quad (\mathcal{A}_i, \mathcal{B}_j, \mathcal{C}_k)_m, \quad \mathcal{A}_i, \mathcal{B}_j, \mathcal{C}_k \in M_l(I(\mathbb{R})), \quad 1 \leq i \leq m-1, \quad 1 \leq j \leq m, \quad 1 \leq k \leq m-1,$$

where  $\mathcal{A}_i$  are the lower subdiagonal blocks,  
 $\mathcal{C}_k$  the upper subdiagonal block,  
 $\mathcal{B}_j$  the diagonal blocks.

A matrix  $\mathcal{A} \in M_{NN}(\mathbb{R})$  with  $a_{ij} \leq 0$  for  $i \neq j$  and  $a_{ii} > 0$  is called an  $L$ -matrix, a matrix with  $a_{ij} \leq 0$  for  $i \neq j$  and  $\mathcal{A}^{-1} \geq \mathcal{O}$  an  $M$ -matrix. An  $M$ -matrix is an  $L$ -matrix (see [12,

§ 3.5]. An interval matrix  $\mathcal{A} \in M_{NN}(I(\mathbb{R}))$  is called an interval  $M$ -( $L$ -)matrix if all  $\mathcal{A} \in \mathcal{A}$  are  $M$ -( $L$ -)matrices.

(2.2)  $\mathcal{M}, \mathcal{N}$   $M$ -matrices,  $\mathcal{M} \leq \mathcal{N} \Rightarrow \mathcal{O} \leq \mathcal{N}^{-1} \leq \mathcal{M}^{-1}$  ([12, § 3.5]).

(2.3) A matrix that is created by setting off-diagonal entries of an  $M$ -matrix to 0, is also an  $M$ -matrix ([12, Thm. 3.12]).

(2.4) If  $\mathcal{M} \in M_{NN}(I(\mathbb{R}))$  is an interval  $L$ -matrix and if there is an  $M$ -matrix  $\mathcal{C} \leq \mathcal{M}$ , then  $\mathcal{M}$  is an interval  $M$ -matrix (A1.2).

(2.5) If  $\mathcal{M}$  is an interval  $M$ -matrix and  $\mathcal{D} = (\delta_{ij}D_j)_{i,j=1}^N$  with  $\underline{D}_j \geq 0$  for all  $j \in \{1, \dots, N\}$ , then  $\mathcal{M} + \mathcal{D}$  is an interval  $M$ -matrix (follows from (2.4)).

In [8] (resp. [10]) we introduced the interval arithmetic reduction method IBU. Algorithm (2.6) is used for IBU in the present paper (see also § 5). By IGA we denote the interval Gauss algorithm, by IGA-EL the application of its elimination part to a matrix, by IGA-BACKS the execution of all operations concerning the “right-hand side” in the elimination part and the backsolving of the resulting triangular system. As our applications in § 5 only require to apply IGA to symmetric tridiagonal systems, its elimination part produces a transformed diagonal  $d$  and a transformed subdiagonal  $f$ . We assume that we want to compute a vector  $x \in V_N(I(\mathbb{R}))$  with

$$x \supseteq \{x \mid \mathcal{M}x = y, \mathcal{M} \in \mathcal{M}, y \in \mathcal{Y}\},$$

where

$$\mathcal{M} = (\mathcal{P}, \mathcal{A}, \mathcal{P})_q, \quad \mathcal{P} \text{ } p \times p\text{-identity matrix,}$$

$$\mathcal{A} = (a_{i-1}, D_i, a_i)_p,$$

$$y = (y_1, \dots, y_q), \quad y_i = (y_{1i}, \dots, y_{pi}), \quad 1 \leq i \leq q, \quad N = pq, \quad q = 2^{k+1} - 1.$$

(2.6) ALGORITHM FOR IBU.

(0)  $(d_0, f_0) := \text{IGA-EL}(\mathcal{A})$

(1)  $j := 2(2)2^{k-1} + 2: \mu_j^1 := \text{IGA-BACKS}(d_0, f_0, y_j)$   
 $q_j^1 := y_{j-1} + y_{j+1} - 2\mu_j^1$

(2)  $r := 2(1)k:$

$$i := 1(1)2^{r-1}: (d_i, f_i) \leftarrow \text{IGA-EL}\left(\mathcal{A} + 2 \cos\left(\frac{2i-1}{2^r} \pi\right) \mathcal{P}\right)$$

$$j := 2^r(2^r)2^{k+1} - 2^r:$$

$$\xi^{2^{r-1}} := \text{IGA-BACKS}(d_{2^{r-1}}, f_{2^{r-1}}, \mu_{j-2^{r-1}}^{r-1} + \mu_{j+2^{r-1}}^{r-1} - q_j^{r-1})$$

$$l := 2^{r-1} - 1(-1)1: \xi^l := \text{IGA-BACKS}(d_l, f_l, \xi^{l+1})$$

$$\mu_j^r := \mu_j^{r-1} + \xi^1$$

$$q_j^r := q_{j-2^{r-1}}^{r-1} + q_{j+2^{r-1}}^{r-1} - 2\mu_j^r$$

$$x_0 := x_{q+1} := \varphi$$

(3)  $\xi^{2^k} := \text{IGA}\left(\mathcal{A} + 2 \cos\left(\frac{2^{k+1}-1}{2^{k+1}} \pi\right) \mathcal{P}, q_j^k\right)$

$$l := 2^k - 1(-1)1: \xi^l := \text{IGA}\left(\mathcal{A} + 2 \cos\left(\frac{2l-1}{2^{k+1}} \pi\right) \mathcal{P}, \xi^{l+1}\right)$$

$$x_{2^k} := \mu_j^r - \xi^1$$

(4)  $r := k - 1(-1)1:$

if  $r < k - 1$  then  $i := 1(1)2^r: (d_i, f_i) := \text{IGA-EL} \left( \mathcal{A} + 2 \cos \left( \frac{2i-1}{2^{r+1}} \pi \right) \mathcal{F} \right)$

$j := 2^r(2^{r+1})2^{k+1} - 2^r:$

$\tilde{z}^{2^r} := \text{IGA-BACKS} (d_{2^r}, f_{2^r}, \varphi_{2^r}^r - (x_{j-2^r} + x_{j+2^r}))$

$l := 2^r - 1(-1): \tilde{z}^l := \text{IGA-BACKS} (d_l, f_l, \tilde{z}^{l+1})$

$x_j := \varphi_j^r - \tilde{z}^l$

(5)  $j := 1(2)2^{k+1} - 1: x_j := \text{IGA-BACKS} (d_0, f_0, y_j - (x_{j-1} + x_{j+1}))$

We need criteria for the applicability of IBU.

**THEOREM 2.1.** *Let the matrix  $\mathcal{B} \in M_{pp}(\mathbb{R})$  defined by*

$$b_{ij} := \begin{cases} \underline{A}_{ii} + 2\{\cos((1 - 2^{-(k+1)})\pi)\}t_{ii}, & i = j, \\ -\max_{1 \leq r \leq k} \max_{1 \leq l \leq 2^r} \left\{ \left| \underline{A}_{ij} + 2 \cos \left( \frac{2l-1}{2^{r+1}} \pi \right) t_{ij} \right| \right\}, & i \neq j, \end{cases}$$

be an *M*-matrix. Then IBU can be applied to  $\mathcal{M} = (\mathcal{T}, \mathcal{A}, \mathcal{T})_q$ ,  $q = 2^{k+1} - 1$ , with  $\mathcal{T} \in M_{pp}(\mathbb{R})$ ,  $\mathcal{A} \in M_{pp}(I(\mathbb{R}))$  if  $\mathcal{T}\mathcal{A} = \mathcal{A}\mathcal{T} \forall \mathcal{A} \in \mathcal{A}$ . No pivoting is necessary to carry out the interval Gauss algorithm (IGA) in IBU [8, Thm. 3.1].

**THEOREM 2.2.** *Let the interval M-matrix*

$$\mathcal{M} = (\mathcal{T}, \mathcal{A}, \mathcal{T})_q$$

satisfy the conditions of Theorem 2.1 and let

$$\mathcal{D} = (\delta_{ij}D_i)_{i,j=1}^p, \quad \underline{D}_i \geq 0, \quad 1 \leq i \leq p.$$

Then IBU is applicable to  $\mathcal{L} = (\mathcal{T}, \mathcal{A} + \mathcal{D}, \mathcal{T})_q$  if  $\mathcal{D}\mathcal{T} = \mathcal{T}\mathcal{D} \forall \mathcal{D} \in \mathcal{D}$ .  $\mathcal{L}$  is an interval *M*-matrix.

*Proof.* From (2.5) it follows that  $\mathcal{L}$  is an interval *M*-matrix. The off-diagonal entries of  $\mathcal{L}$  and  $\mathcal{M}$  are identical.

Let  $\mathcal{B}'$  the matrix defined in Theorem 2.1 for  $\mathcal{L}$ ,  $\mathcal{B}$  that for  $\mathcal{M}$ ; then

$$b'_{ij} = b_{ij}, \quad i \neq j,$$

$$b'_{ii} = \underline{A}_{ii} + \underline{D}_i + 2\{\cos((1 - 2^{-(k+1)})\pi)\}t_{ii}$$

$$\geq \underline{A}_{ii} + 2\{\cos((1 - 2^{-(k+1)})\pi)\}t_{ii} = b_{ii}, \quad i = j.$$

Hence  $\mathcal{B}'$  is an interval *L*-matrix with  $\mathcal{B}' \geq \mathcal{B}$ . Therefore  $\mathcal{B}'$  is an *M*-matrix by (2.4). From Theorem 2.1 we deduce the applicability of IBU to  $\mathcal{L}$ .  $\square$

An important property of IBU is given by

**LEMMA 2.3.** *Let IBU be applicable to the M-matrix  $\mathcal{M}$ . Then*

(a)  $\text{IBU}(\mathcal{M}, y) = [\underline{\mathcal{M}}^{-1}y, \overline{\mathcal{M}}^{-1}y]$  for  $\varphi \leq y$ ;

(b)  $\text{IBU}(\mathcal{M}, y) = [\underline{\mathcal{M}}^{-1}y, \overline{\mathcal{M}}^{-1}y]$  for  $\varphi \geq y$ ;

(c)  $\text{IBU}(\mathcal{M}, y) = [\underline{\mathcal{M}}^{-1}y, \overline{\mathcal{M}}^{-1}y]$  for  $\varphi \in y$ .

*Proof.* [10, Thm. 2.2.3].  $\square$

From the continuity of the interval operations (+, -, \*, /), it follows that IGA and IBU are continuous.

**LEMMA 2.4.** *If*

$$(\mathcal{M}^{(k)})_{k \in \mathbb{N}_0} \in M_{NN}(I(\mathbb{R})), \quad \mathcal{M}^{(k)} \rightarrow \mathcal{M}^* \in M_{NN}(I(\mathbb{R})),$$

$$(y^k)_{k \in \mathbb{N}_0} \subset V_N(I(\mathbb{R})), \quad y^k \rightarrow y^* \in V_N(I(\mathbb{R})) \quad (k \rightarrow \infty),$$

then

$$\begin{aligned} \text{IBU}(\mathcal{M}^{(k)}, \mathbf{y}^k) &\rightarrow \text{IBU}(\mathcal{M}^*, \mathbf{y}^*), \quad \text{resp.} \\ \text{IGA}(\mathcal{M}^{(k)}, \mathbf{y}^k) &\rightarrow \text{IGA}(\mathcal{M}^*, \mathbf{y}^*) \quad (k \rightarrow \infty), \end{aligned}$$

provided IBU and IGA can be carried out with all matrices.

*Proof.* See [8, Remarks 3.3, 4.3].  $\square$

**Abbreviations.**

GCG	generalized CG method	§ 6, (4) and [4]
GCG+S	GCG with "shifted" diagonal	§ 6, (4)
GCG+R5	GCG with restarts every 5 steps	§ 6, (5)
GCG+R10	GCG with restarts every 10 steps	§ 6, (5)
IBU	reduction method for interval matrices	(2.6)
IGA	interval Gauss algorithm	[8, § 2], [2, § 15]
IMNAB	modified interval Newton-like method with IBU	(3.3)
IMNABHIN	IMNAB with a modification of IGS as auxiliary method	(A5.14)
IMNABM	monotone variants of IMNAB	after (3.8)
IMNAB+IGS	IMNAB with auxiliary method	(3.8) (a)
IMNAB+IS2	IMNAB with auxiliary method	(3.8) (b)
NBSOR	nonlinear block SOR method	[5]

**3. An iterative method for systems of nonlinear equations.** We assume the following general situation:

(3.1) Given  $f: \mathcal{D} \rightarrow V_N(\mathbb{R})$ ,  $\mathcal{D} \subseteq V_N(\mathbb{R})$ ,  $f$  continuously differentiable in  $\mathcal{D}$  and an interval vector  $x^0 \in V_N(I(\mathbb{R}))$  contained in  $\mathcal{D}$  and including exactly one zero  $y$  of  $f$ :

$$y \in x^0 \subseteq \mathcal{D},$$

we try to determine  $y$ .

In order to extend the applicability to classes of equations other than the Poisson equation we split the Jacobi matrix:

$$f'(\cdot) = \mathcal{M}(\cdot) - \mathcal{N}(\cdot).$$

Assuming that  $\mathcal{M}(x)$  is invertible for all  $x \in x^0$  we obtain

$$(3.2) \quad \forall x \subseteq x^0 \quad \forall \varphi \in x \quad \exists u \in x \quad y = \varphi - \mathcal{M}(u)^{-1} \{ \mathcal{N}(u)(\varphi - y) + f(\varphi) \}.$$

We further require the existence of interval extensions of  $\mathcal{M}(\cdot)$  and  $\mathcal{N}(\cdot)$  in  $x^0$ . Now we construct interval matrices  $\hat{\mathcal{M}}(x)$  for  $x \subseteq x^0$  such that  $\mathcal{M}(x) \subseteq \hat{\mathcal{M}}(x)$  and such that IBU can be carried out. The subset property (A1.3) enables us to write for all  $x \subseteq x^0$  with  $y \in x$  and all  $\varphi \in x$

$$y \in \varphi - \text{IBU}(\hat{\mathcal{M}}(x), \mathcal{N}(x)(\varphi - x) + f(\varphi)).$$

We then define the

(3.3) *modified interval Newton-like method with IBU (IMNAB)*

$$k := 0(1)\infty:$$

$$m^k \in x^k; \quad x^{k+1} := \{ m^k - \text{IBU}(\hat{\mathcal{M}}(x^k), \mathcal{N}(x^k)(m^k - x^k) + f(m^k)) \} \cap x^k.$$

The method INB described in [8] is a special case of IMNAB with  $\mathcal{N}(\cdot) = \emptyset$ .

Before we discuss the properties of IMNAB, we define  $\hat{M}(x)$  and  $\mathcal{N}(x)$  for a class of problems suitable for our applications. We assume that  $f$  can be written as follows:

$$\begin{aligned}
 (3.4) \quad & f(x) = \mathcal{B}x + \Phi(x), \quad \mathcal{B} = (\mathcal{T}_{j-1}, \mathcal{A}_j, \mathcal{T}_j)_q, \quad \mathcal{A}_j, \mathcal{T}_j \in M_{pp}(\mathbb{R}), \\
 & \Phi(x) = (\phi_i(x_i))_{i=1}^N, \quad N = pq, \quad q = 2^{k+1} - 1, \quad \text{hence} \\
 & f'(x) = \mathcal{B} + \Phi'(x) = \mathcal{B} + (\delta_{ij}\phi'_i(x_i))_{i,j=1}^N.
 \end{aligned}$$

Let

$$\begin{aligned}
 (3.5) \quad & \hat{\mathcal{B}} := (\mathcal{T}, \mathcal{A}, \mathcal{T})_q, \quad \mathcal{A}, \mathcal{T} \in M_{pp}(\mathbb{R}), \\
 & \hat{\Phi}(x) = (\delta_{ij}\hat{\phi}_i(x))_{i,j=1}^q, \quad \hat{\Phi}(x) := (\delta_{ij}\hat{\phi}_i(x))_{i,j=1}^p, \\
 & \hat{\phi}_i(x) := [\min \{ \phi'_{i+(k-1)p}(X_{i+(k-1)p}) \mid 1 \leq k \leq q \}, \\
 & \quad \max \{ \phi'_{i+(k-1)p}(X_{i+(k-1)p}) \mid 1 \leq k \leq q \}], \\
 & \hat{M}(x) := \hat{\mathcal{B}} + \hat{\Phi}(x) = (\mathcal{T}, \mathcal{A} + \hat{\Phi}(x), \mathcal{T})_q, \quad \mathcal{M}(x) := \hat{\mathcal{B}} + \Phi'(x), \\
 & \mathcal{N} := \hat{\mathcal{B}} - \mathcal{B} = (\mathcal{T} - \mathcal{T}_{j-1}, \mathcal{A} - \mathcal{A}_j, \mathcal{T} - \mathcal{T}_j)_q,
 \end{aligned}$$

with an appropriate choice of  $\mathcal{A}$  and  $\mathcal{T}$ . At least we require

$$(3.6) \quad \mathcal{A}\mathcal{T} = \mathcal{T}\mathcal{A}.$$

The “right-hand side” of IBU in IMNAB then reduces to

$$(3.7) \quad \mathcal{N}(x^k)(m^k - x^k) + f(m^k) = -\mathcal{N}x^k + \hat{\mathcal{B}}m^k + \Phi(m^k).$$

We now discuss the properties of (3.3). By applying the subset property (A1.3) we can show by induction:

LEMMA 3.1. *If IMNAB as given by (3.3) can be carried out, all components of the sequence  $(x^k)_{k \in \mathbb{N}_0}$  computed by IMNAB include  $y$ :  $y \in x^k \forall k \in \mathbb{N}_0$ .*

Due to the intersection in each step there exists  $x^* \in V_N(I(\mathbb{R}))$  such that

$$y \in x^* = \lim_{k \rightarrow \infty} x^k = \bigcap_{k \in \mathbb{N}_0} x^k.$$

In order to generate  $d(x^*) = \emptyset$ , i.e.  $x^* = y$ , additional precautions have to be taken. We propose two ways:

1. We combine IMNAB with an auxiliary method. By IMNABHIN we denote the variant described in [8] (resp. [10]) where a certain number of auxiliary steps is carried out to “restart” the iteration, but only if the latter is likely to stop on a nondegenerate interval vector.

In the present paper we test two other variants where exactly one auxiliary step is carried out in each step of IMNAB. A similar method has been proposed in [2, § 19], for the interval Newton method. Here we propose a modification of the interval Newton–Gauss–Seidel type method (denoted by IGS)—see [2, § 22, (4)]—and a modification of the symmetric method IS2 presented in [9]. We present these methods in a form suitable for systems (3.4). A more general formulation is naturally possible.

(3.8) *IMNAB with auxiliary methods*

$$k := 0(1)\infty:$$

$$m^k \in x^k; \quad \tilde{x}^{k+1} := \{m^k - \text{IBU}(\hat{\mathcal{B}} + \hat{\Phi}(x^k), -\mathcal{N}x^k + \hat{\mathcal{B}}m^k + \Phi(m^k))\} \cap x^k;$$

(a) *variant IMNAB+IGS*

$$i := 1(1)N:$$

$$m_i^{(k+1)} \in \tilde{X}_i^{(k+1)};$$

$$X_i^{(k+1)} := \left\{ m_i^{(k+1)} - \left( - \sum_{j=1}^{i-1} b_{ij} X_j^{(k+1)} - \sum_{j=i+1}^N b_{ij} \tilde{X}_j^{(k+1)} + b_{ii} m_i^{(k+1)} + \phi_i(m_i^{(k+1)}) \right) / (b_{ii} + \hat{\phi}_{i \bmod p}(x^k)) \right\} \cap \tilde{X}_i^{(k+1)}$$

or (b) *variant* IMNAB + IS2

$$i := 1(1)N:$$

$$m_i^{(k+1)} \in \tilde{X}_i^{(k+1)}; Z := - \sum_{j=1}^{i-1} b_{ij} X_j^{(k+1/2)} + b_{ii} m_i^{(k+1)} + \phi_i(m_i^{(k+1)})$$

$$X_i^{(k+1/2)} := \left\{ m_i^{(k+1)} - \left( - \sum_{j=i+1}^N b_{ij} \tilde{X}_j^{(k+1)} + Z \right) / (b_{ii} + \phi'_i(X_i^{(k)})) \right\} \cap \tilde{X}_i^{(k+1)}$$

$$i := N - 1(-1)1:$$

$$X_i^{(k+1)} := \left\{ m_i^{(k+1)} - \left( - \sum_{j=i+1}^N b_{ij} X_j^{(k+1)} + Z \right) / (b_{ii} + \phi'_i(X_i^{(k)})) \right\} \cap X_i^{(k+1/2)}.$$

An auxiliary step of variant (a) results in larger intervals than a step of variant (b). On the other hand a step of variant (b) requires less arithmetic work and less storage as only those extensions of partial derivatives have to be stored that are needed by IMNAB, i.e. the  $m$  intervals needed for  $\hat{\Phi}$  instead of the whole diagonal of  $\Phi'$  of size  $N$ . We assume that the information about the  $b_{ii}$  can be easily recovered without storing them (variant (a)). This will be explained in detail when discussing the applications in § 5.

2. We restrict the choice of the  $m^k$  such that monotonicity arguments ensure the convergence to  $y$ . This version will be denoted by IMNABM.

First of all we reformulate the convergence [8, Thm. 5.3] for IMNABHIN. Due to the more general character of the method and (3.4) we need additional conditions.

**THEOREM 3.2.** *Given  $f(x) = \varphi$  where  $f$  is defined by (3.4), let*

- 1)  $\mathcal{B}$  be an  $M$ -matrix,
- 2)  $\hat{\Phi}'(x^0) \cong \mathcal{O}$ ,
- 3)  $\overline{\text{IBU}}$  be applicable to  $\hat{M}(x^0)$ .

*Then the sequence  $(x^k)_{k \in \mathbb{N}_0}$  computed by IMNABHIN converges to  $y$ .*

*Proof.* From condition 1) we deduce  $b_{ii} > 0$  for all  $i \in \{1, \dots, N\}$  and together with condition 2) we know

$$\underline{D_{ii}(x^0)} = b_{ii} + \underline{\phi'_i(X_i^0)} > 0, \quad \text{i.e. } 0 \notin D_{ii}(x^0),$$

where  $\mathcal{D}(x^0)$  denotes the diagonal part of  $f'(x^0)$ . Splitting  $\mathcal{B}$  into  $\mathcal{B} = \mathcal{D} - \mathcal{L} - \mathcal{U}$  we have  $\mathcal{D}, \mathcal{L}, \mathcal{U} \cong \mathcal{O}$ , hence  $\rho(\mathcal{D}^{-1}(\mathcal{L} + \mathcal{U})) < 1$ . Now the proof can be continued exactly like that of [8, Thm. 5.3].  $\square$

The convergence of IMNAB + IGS and IMNAB + IS2 to  $y$  can be proved analogously to that of IGS and IS2 (see [2, Chapter 22], respectively [9]) because of

$$x^{k+1} \subseteq \tilde{x}^{k+1} \subseteq x^k \subseteq \tilde{x}^k \subseteq x^{k-1} \subseteq \dots \subseteq x^0$$

and the subset property. For IMNAB + IGS we further need the relation  $\rho(|\hat{\mathcal{D}}(x)|\{|\mathcal{L}| + |\mathcal{U}|\}) < 1$ , where  $\hat{\mathcal{D}}(x) = (\delta_{ij} / (b_{ii} + \hat{\phi}_{i \bmod p}(x)))_{i,j=1}^N$  and  $\mathcal{L}, \mathcal{U}$  represent the off-diagonal entries of  $\mathcal{B}$ .

We now analyse the convergence of IMNABM. By choosing  $m^k := \overline{x^k}$  or  $m^k := \underline{x^k}$  and assuming  $f(x^0) \cong \varphi$  or  $f(\underline{x}^0) \cong \varphi$ , we can operate with known signs of interval bounds in important steps of the convergence analysis. It is then possible to apply monotonicity arguments by which the convergence to  $y$  can be proved without

any auxiliary step. In addition we give estimates for the asymptotic convergence rate when discussing the “monotone” versions.

**THEOREM 3.3.** *If*

- (1)  $\hat{M}(x^0)$  is an interval  $M$ -matrix,
- (2)  $\mathcal{N}(x^0) \cong \mathcal{O}$ ,
- (3)  $(\hat{M}(x^0) - \mathcal{N}(x^0))^{-1} \cong \mathcal{O}$ ,
- (4) IBU can be carried out with  $\hat{M}(x^0)$ ,
- (5)  $m^k := \overline{x^k}$  for all  $k \in \mathbb{N}_0$ ,
- (6)  $f(x^0) \cong \varrho$ ,

then

- (1) the sequence  $(x^k)_{k \in \mathbb{N}_0}$  computed by IMNABM converges to  $y$ ;
- (2)  $\overline{x^{k+1}} = \overline{x^k} = \hat{M}(x^k)^{-1} f(x^k) \leq \overline{x^k}$  for all  $k \in \mathbb{N}_0$ .

*Proof.* Since  $\overline{x^0} - x^0 = [\varrho, d(x^0)]$  we conclude from the conditions (2) and (5) that

$$(3.9) \quad \mathcal{N}(x^0)(m^0 - x^0) = [\varrho, \overline{\mathcal{N}(x^0)}d(x^0)].$$

We then obtain

$$(3.10) \quad \begin{aligned} x^1 &= \{\overline{x^0} - \text{IBU}(\hat{M}(x^0), \mathcal{N}(x^0)(\overline{x^0} - x^0) + f(\overline{x^0}))\} \cap x^0 \\ &\subseteq \overline{x^0} - \text{IBU}(\hat{M}(x^0), \mathcal{N}(x^0)(\overline{x^0} - x^0) + f(\overline{x^0})) \\ &= \overline{x^0} - [\hat{M}(x^0)^{-1} f(\overline{x^0}), \hat{M}(x^0)^{-1}(\overline{\mathcal{N}(x^0)}(\overline{x^0} - x^0) + f(\overline{x^0}))] \\ &= [\overline{x^0} - \hat{M}(x^0)^{-1}(\overline{\mathcal{N}(x^0)}d(x^0) + f(\overline{x^0})), \overline{x^0} - \hat{M}(x^0)^{-1}f(\overline{x^0})]. \end{aligned}$$

In the third line we have applied Lemma 3.3(a) in view of (3.9) and condition (5). Because of condition (1), we also know that

$$(3.11) \quad \overline{x^0} - \hat{M}(x^0)^{-1} f(\overline{x^0}) \leq \overline{x^0}$$

and therefore

$$(3.12) \quad \overline{x^1} = \overline{x^0} - \hat{M}(x^0)^{-1} f(\overline{x^0}), \text{ thus } \overline{x^1} \leq \overline{x^0}.$$

This enables us—together with the conditions (1), (2) and (6)—to conclude

$$(3.13) \quad \begin{aligned} f(x^1) &= f(\overline{x^0}) + f'(u)(\overline{x^1} - \overline{x^0}) \\ &= f(\overline{x^0}) + f'(u)(-\hat{M}(x^0)^{-1} f(\overline{x^0})) \\ &= \{\varphi - f'(u)\hat{M}(x^0)^{-1}\} f(\overline{x^0}) \\ &= \{\hat{M}(x^0) - f'(u)\}\hat{M}(x^0)^{-1} f(\overline{x^0}) \\ &= \{\hat{M}(x^0) - \mathcal{M}(u) + \mathcal{N}(u)\}\hat{M}(x^0)^{-1} f(\overline{x^0}) \geq \varrho \end{aligned}$$

with a vector  $u \in [\overline{x^1}, \overline{x^0}]$ . Note that  $x^1 \subseteq x^0$ , thus  $u \in x^0$  and therefore  $\mathcal{M}(u) \in \mathcal{M}(x^0) \subseteq \hat{M}(x^0)$ , thus  $\mathcal{M}(u) \leq \hat{M}(x^0)$ . Analogously we show by induction for all  $k \in \mathbb{N}_0$

$$(3.14) \quad \begin{aligned} (1) \quad &x^{k+1} \subseteq [\overline{x^k} - \hat{M}(x^k)^{-1}\{\overline{\mathcal{N}(x^k)}d(x^k) + f(\overline{x^k})\}, \overline{x^k} - \hat{M}(x^k)^{-1}f(\overline{x^k})] \\ (2) \quad &\overline{x^{k+1}} = \overline{x^k} - \hat{M}(x^k)^{-1} f(\overline{x^k}) \leq \overline{x^k}, \\ (3) \quad &f(\overline{x^{k+1}}) \geq \varrho. \end{aligned}$$

The intersection in each step yields a nested sequence with

$$(3.15) \quad \overline{x^0} \leq \underline{x^k} \leq \overline{x^{k+1}} \leq \overline{x^{k+1}} \leq \overline{x^k} \leq \overline{x^0} \quad \text{for all } k \in \mathbb{N}_0.$$

Therefore there is a vector  $x^* \in V_N(\mathbb{R})$  such that

$$x^* = \lim_{k \rightarrow \infty} \overline{x^k}.$$

Since  $f(\overline{x^{k+1}}) \geq \varphi$ ,  $x^k \subseteq x^0$  (thus  $\hat{M}(x^k) \subseteq \hat{M}(x^0)$  and also  $\mathcal{O} \subseteq \overline{\hat{M}(x^0)^{-1}} \subseteq \overline{\hat{M}(x^k)^{-1}}$ ) we now get

$$(3.16) \quad \varphi \leq \overline{\hat{M}(x^0)^{-1} f(x^k)} \leq \overline{\hat{M}(x^k)^{-1} f(x^k)} = \overline{x^k - x^{k+1}} \rightarrow \varphi \quad (k \rightarrow \infty).$$

This means  $f(\overline{x^k}) \rightarrow \varphi$ , hence  $\overline{x^k} \rightarrow y$  ( $k \rightarrow \infty$ ) as  $y$  is the only zero of  $f$  in  $x^0$ . Now we can show that  $x^k \rightarrow y$ . We have for all  $k \in \mathbb{N}_0$

$$(3.17) \quad d(x^{k+1}) = \overline{x^{k+1} - \underline{x}^{k+1}} \leq (\underline{\hat{M}(x^k)^{-1}} - \overline{\hat{M}(x^k)^{-1}}) f(\overline{x^k}) + \underline{\hat{M}(x^k)^{-1} \mathcal{N}(x^k)} d(x^k).$$

Due to the intersection there exists

$$x^* = \bigcap_{k \in \mathbb{N}_0} x^k = \lim_{k \rightarrow \infty} x^k.$$

Applying continuity arguments to (3.17) we get with  $\overline{x^k} \rightarrow y$

$$(3.18) \quad d(x^*) \leq \underline{\hat{M}(x^*)^{-1} \mathcal{N}(x^*)} d(x^*).$$

With  $x^* \subseteq x^0$ , hence  $\hat{M}(x^*) \subseteq \hat{M}(x^0)$ ,  $\mathcal{N}(x^*) \subseteq \mathcal{N}(x^0)$ , and the conditions (1) and (2) we note

$$(3.19) \quad \mathcal{O} \subseteq \underline{\hat{M}(x^*)^{-1} \mathcal{N}(x^*)} \subseteq \underline{\hat{M}(x^0)^{-1} \mathcal{N}(x^0)},$$

and therefore with the conditions (1)-(3) and [12, Thms. 2.8, 3.13]

$$(3.20) \quad \rho(\underline{\hat{M}(x^*)^{-1} \mathcal{N}(x^*)}) \leq \rho(\underline{\hat{M}(x^0)^{-1} \mathcal{N}(x^0)}) < 1.$$

We rearrange (3.18):

$$(3.21) \quad (\mathcal{I} - \underline{\hat{M}(x^*)^{-1} \mathcal{N}(x^*)}) d(x^*) \leq \varphi.$$

Multiplying (3.20) with the positive inverse of that matrix (see [12, Thm. 3.8] together with (3.20)), we obtain  $d(x^*) \leq \varphi$ , i.e.  $x^* \equiv x^* = y$ .

A similar theorem is valid for

$$f(\underline{x}) \leq \varphi \quad \text{and} \quad m^k := \underline{x}^k \quad \text{for all } k \in \mathbb{N}_0.$$

Then we have

$$\underline{x^{k+1}} = \underline{x^k - \hat{M}(x^k)^{-1} f(x^k)} \geq \underline{x^k}.$$

For the method IMNABHIN we can estimate the asymptotic convergence speed as in [8, (5.24)]. The monotone version leads to a more satisfactory estimate.

**THEOREM 3.4.** *Under the conditions of Theorem 3.3 the following estimate is valid:*

$$R_1(\text{IMNABM}, y) \leq \rho(\underline{\hat{M}(y)^{-1} \{ \mathcal{N}(y) + d(\mathcal{M}(y)) \}}).$$

*Proof.* From (3.17) we know

$$(3.22) \quad d(x^{k+1}) \leq \underline{\hat{M}(x^k)^{-1} \mathcal{N}(x^k)} d(x^k) + (\underline{\hat{M}(x^k)^{-1}} - \overline{\hat{M}(x^k)^{-1}}) f(\overline{x^k}).$$

We further note that for all  $k \in \mathbb{N}_0$ ,  $y \leq \overline{x^k}$ ; hence

$$\begin{aligned} \exists u \in [y, \overline{x^k}]: \varphi \leq f(\overline{x^k}) = f'(u)(\overline{x^k} - y) \quad \text{and} \\ f'(u) \leq f'(x^k) \leq \underline{\hat{M}(x^k)^{-1} \mathcal{N}(x^k)}; \end{aligned}$$



hence  $f'(y)(\overline{x^k - y}) \leq (\overline{\hat{M}(x^k)} - \overline{\mathcal{N}(x^k)})(\overline{x^k - y})$ .

We examine the second term of the right-hand side of (3.22).

$$\begin{aligned}
 \varphi &\leq (\overline{\hat{M}(x^k)}^{-1} - \overline{\hat{M}(x^k)}^{-1})f(\overline{x^k}) \\
 &\leq (\overline{\hat{M}(x^k)}^{-1} - \overline{\hat{M}(x^k)}^{-1})(\overline{\hat{M}(x^k)} - \overline{\mathcal{N}(x^k)})(\overline{x^k - y}) \\
 (3.23) \quad &\leq (\overline{\hat{M}(x^k)}^{-1} - \overline{\hat{M}(x^k)}^{-1})\overline{\hat{M}(x^k)}(\overline{x^k - y}) \\
 &= \overline{\hat{M}(x^k)}^{-1}d(\overline{\hat{M}(x^k)})\overline{\hat{M}(x^k)}^{-1}\overline{\hat{M}(x^k)}(\overline{x^k - y}) \\
 &\leq \overline{\hat{M}(x^k)}^{-1}d(\overline{\hat{M}(x^k)})d(\overline{x^k}).
 \end{aligned}$$

(3.22) and (3.23) yield the assertion together with  $x^k \rightarrow y$  (see also [10, Lemma 1.9]).

For the sequence  $(x^k)_{k \in \mathbb{N}_0}$  of upper bounds we can prove an even better estimate for  $R_1(\text{IMNABM}, y)$ : for all  $k \in \mathbb{N}_0$  there exists an  $u \in [y, x^k]$  such that

$$\begin{aligned}
 \varphi &\leq \overline{x^{k+1} - y} = \overline{x^k - y} - \overline{\hat{M}(x^k)}^{-1}f(\overline{x^k}) \\
 &= \overline{x^k - y} - \overline{\hat{M}(x^k)}^{-1}(f(\overline{x^k}) - f(y)) \\
 &= \overline{x^k - y} - \overline{\hat{M}(x^k)}^{-1}f'(u)(\overline{x^k - y}) \\
 &= \{\mathcal{I} - \overline{\hat{M}(x^k)}^{-1}(M'(u) - \mathcal{N}(u))\}(\overline{x^k - y}) \\
 &= \overline{\hat{M}(x^k)}^{-1}\{\overline{\hat{M}(x^k)} - M'(u) + \mathcal{N}(u)\}(\overline{x^k - y}) \\
 &\leq \overline{\hat{M}(x^k)}^{-1}\{d(\overline{\hat{M}(x^k)}) + \overline{\mathcal{N}(x^k)}\}(\overline{x^k - y}).
 \end{aligned}$$

Remember now conditions (2), (3) of Theorem 3.3. Therefore

$$\begin{aligned}
 \overline{\hat{M}(x^k)} - (d(\overline{\hat{M}(x^k)}) + \overline{\mathcal{N}(x^k)}) &= \overline{\hat{M}(x^k)} - \overline{\hat{M}(x^k)} + \overline{\hat{M}(x^k)} - \overline{\mathcal{N}(x^k)} \\
 &= \overline{\hat{M}(x^k)} - \overline{\mathcal{N}(x^k)}
 \end{aligned}$$

is a regular splitting. It can be shown therefore that

$$(3.24) \quad R_1((\overline{x^k})_{k \in \mathbb{N}_0}) \leq \rho(\overline{\hat{M}(x^k)}^{-1}\{d(\overline{\mathcal{M}(x^k)}) + \overline{\mathcal{N}(x^k)}\}) < 1.$$

The above results are also valid for INB with  $\mathcal{N}(\cdot) \equiv \mathcal{O}$ ,  $\hat{M}(\cdot) = \hat{\mathcal{F}}(\cdot)$ . A simple criterion to decide if the right-hand side of the inequality in Theorem 3.4 remains less than 1 follows.

LEMMA 3.5. *If  $\rho(\hat{\mathcal{F}}(x^0)^{-1}) < 1/\max\{d(\hat{F}_{ij}(x^0)) | 1 \leq i, j \leq N\}$  where  $\hat{\mathcal{F}}(x^0)$  is an interval  $M$ -matrix, then  $\rho(\hat{\mathcal{F}}(y)^{-1}d(\hat{\mathcal{F}}(y))) < 1$ .*

*Proof.*

$$\begin{aligned}
 \rho(\hat{\mathcal{F}}(y)^{-1}d(\hat{\mathcal{F}}(y))) &\leq \rho(\hat{\mathcal{F}}(x^0)^{-1}d(\hat{\mathcal{F}}(x^0))) \\
 &\leq \rho(\hat{\mathcal{F}}(x^0)^{-1} \max\{d(\hat{F}_{ij}(x^0)) | 1 \leq i, j \leq N\} \mathcal{I}) \\
 &= \max\{d(\hat{F}_{ij}(x^0)) | 1 \leq i, j \leq N\} \rho(\hat{\mathcal{F}}(x^0)^{-1}). \quad \square
 \end{aligned}$$

**4. Computation of a starting interval vector.** If  $f$  has the properties required in Theorem 3.2 we can compute an initial inclusion of the solution  $y$  with [7, Thm. 13.4.6] as

$$(4.1) \quad x^0 := [-\mathcal{B}^{-1}|\Phi(\varrho)|, \mathcal{B}^{-1}|\Phi(\varrho)|].$$

If the direct solution of (4.1) is too expensive or not possible because of storage restrictions, we can at least define an iterative procedure including IBU applied to  $\hat{\mathcal{B}}$ .

This procedure yields an improved starting vector when started with an initial inclusion of  $\mathcal{Y}$ .

As  $\mathcal{B}$  is an  $M$ -matrix,  $\hat{\mathcal{B}}$  is also an  $M$ -matrix if  $\mathcal{N} \cong \mathcal{O}$ :  $\hat{\mathcal{B}} = \mathcal{B} + \mathcal{N} \cong \mathcal{B}$  is still an invertible  $L$ -matrix with  $\hat{\mathcal{B}}^{-1} \leq \mathcal{B}^{-1}$  (2.2). If we further know (e.g. Theorem 2.1) that IBU is applicable to  $\hat{\mathcal{B}}$ , we use the splitting for the nonlinear system to be solved to define the method

$$(4.2) \quad \mathcal{z}^{k+1} := \text{IBU}(\hat{\mathcal{B}}, \mathcal{N}_{\mathcal{z}^k} + [-|\Phi(\varrho)|, |\Phi(\varrho)|]) \cap \mathcal{z}^k, \quad k := 0(1)\infty.$$

The vector  $\mathcal{z}^0$  can be a perhaps only poor inclusion of  $\mathcal{Y}$ . If we start with a symmetric vector  $\mathcal{z}^0 = [-\overline{\mathcal{z}^0}, \underline{\mathcal{z}^0}]$ , we only deal with symmetric vectors when carrying out (4.2). In fact the computation of (4.2) reduces to

$$(4.3) \quad \overline{\mathcal{z}^{k+1}} := \min[(\hat{\mathcal{B}})^{-1}\{\mathcal{N}_{\overline{\mathcal{z}^k}} + |\Phi(\varrho)|\}, \overline{\mathcal{z}^k}], \quad \underline{\mathcal{z}^{k+1}} := -\overline{\mathcal{z}^{k+1}}.$$

Hence all interval operations necessary for (4.3) can be carried out with special procedures for symmetric intervals computing the correctly rounded upper bounds and setting the lower bounds to the negative value of the upper bounds. In particular we use a version of IBU for symmetric vectors. Compared to the more general interval operations we save the major part of the computation time: we only need half of the number of floating-point operations and do not have to consider different cases. The convergence properties of (4.2) and related methods will be discussed in a subsequent paper. For the special case (4.3) a convergence proof is easy to accomplish:

$\hat{\mathcal{B}}$  is an  $M$ -matrix,  $\mathcal{N} \cong \mathcal{O}$ ,  $\mathcal{z}^k$  and  $[-|\Phi(\varrho)|, |\Phi(\varrho)|]$  are symmetric vectors (a vector  $x$  is symmetric if  $x = -x$ ), hence  $\mathcal{N}_{\mathcal{z}^k} + [-|\Phi(\varrho)|, |\Phi(\varrho)|]$  is also symmetric.

Because of the intersection in each step there exists  $\mathcal{z}^* = \bigcap_{k \in \mathbb{N}_0} \mathcal{z}^k$  and  $\mathcal{N}_{\mathcal{z}^*} + [-|\Phi(\varrho)|, |\Phi(\varrho)|]$  is also symmetric as the interval operations are continuous. From Lemma 2.3 it follows that

$$\begin{aligned} \mathcal{z}^* &= \text{IBU}(\hat{\mathcal{B}}, \mathcal{N}_{\mathcal{z}^*} + [-|\Phi(\varrho)|, |\Phi(\varrho)|]) \cap \mathcal{z}^* \\ &= (\hat{\mathcal{B}})^{-1}(\mathcal{N}_{\mathcal{z}^*} + [-|\Phi(\varrho)|, |\Phi(\varrho)|]) \cap \mathcal{z}^* \\ &\subseteq (\hat{\mathcal{B}})^{-1}(\mathcal{N}_{\mathcal{z}^*} + [-|\Phi(\varrho)|, |\Phi(\varrho)|]); \end{aligned}$$

hence

$$\overline{\mathcal{z}^*} \leq (\hat{\mathcal{B}})^{-1}(\mathcal{N}_{\overline{\mathcal{z}^*}} + |\Phi(\varrho)|), \quad \underline{\mathcal{z}^*} = -\overline{\mathcal{z}^*}$$

or

$$\begin{aligned} \overline{\mathcal{z}^*} &\leq (\mathcal{I} - (\hat{\mathcal{B}})^{-1}\mathcal{N})^{-1}|\Phi(\varrho)|, \\ \underline{\mathcal{z}^*} &\geq -(\mathcal{I} - (\hat{\mathcal{B}})^{-1}\mathcal{N})^{-1}|\Phi(\varrho)|. \end{aligned}$$

The inverse matrix exists and is nonnegative as  $\hat{\mathcal{B}} - \mathcal{N} = \mathcal{B}$  is an  $M$ -matrix. If  $x^0 \subseteq \mathcal{z}^0$  then we also have  $x^0 \subseteq \mathcal{z}^*$ , and analogously if  $\mathcal{Y} \in \mathcal{z}^0 \cap x^0$  then  $\mathcal{Y} \in \mathcal{z}^*$ .

**5. Applications.** For the method under consideration in this paper we use the concept of matrix splitting to create matrices suitable for the application of IBU. We consider the elliptic PDE

$$(5.1) \quad \begin{aligned} (a(s)u_s)_s + (b(t)u_t)_t &= f(s, t, u(s, t)) \quad \text{on } I = [0, 1] \times [0, 1], \\ u(s, t) &= g(s, t) \quad \text{on the boundary of } I, \\ g &\text{ continuous, } a, b \text{ continuously differentiable on } [0, 1], \\ a(s), b(t) &> 0 \quad \text{for } s, t \in [0, 1], \quad \frac{\partial f}{\partial u} \geq 0. \end{aligned}$$

It can be shown that this problem has a unique solution ([3, § 7, Chap. II]). The five point discretization with central differential quotients with a mesh size  $h = 1/(m + 1)$ ,  $m \in \mathbb{N}$ , in both directions produces a nonlinear system (3.4) with

$$\begin{aligned}
 & p = q = m, \\
 & \phi_i(x_i) = h^2 f(jh, kh, x_i) - r_i, \quad i = (k - 1)m + j, \quad 1 \leq j, k \leq m, \\
 (5.2) \quad & \mathcal{T}_j = -\beta_{j+1/2} \mathcal{F}, \quad \mathcal{A}_j = (-\alpha_{i-1/2}, \alpha_{i-1/2} + \alpha_{i+1/2} + \beta_{j-1/2} + \beta_{j+1/2}, -\alpha_{i+1/2})_m, \\
 & \beta_{j\pm 1/2} = b((j \pm 1/2)h) > 0, \quad \alpha_{j\pm 1/2} = a((j \pm 1/2)h) > 0, \\
 & \mathcal{z} \text{ contains the boundary values } \Phi(x) = (\phi_i(x_i))_{i=1}^N, \Phi'(x) \cong \mathcal{O}.
 \end{aligned}$$

With [7, Thm. 4.4.1] it can be shown that there is a unique solution of this system, too.

We now define  $\hat{\mathcal{B}}$  according to (3.5).

Let

$$\begin{aligned}
 \beta & := \min \{ \beta_{j+1/2} | 1 \leq j \leq m - 1 \}, \\
 \beta_{\max} & := \max \{ \beta_{j-1/2} + \beta_{j+1/2} | 1 \leq j \leq m \}, \\
 \mathcal{T} & := -\beta \mathcal{F}, \quad \mathcal{A} := (-\alpha_{i-1/2}, \beta_{\max} + \alpha_{i-1/2} + \alpha_{i+1/2}, -\alpha_{i+1/2})_m, \\
 \mathcal{N}(x) & \equiv \mathcal{N} := ((\beta_{j-1/2} - \beta) \mathcal{F}, (\beta_{\max} - \beta_{j-1/2} - \beta_{j+1/2}) \mathcal{F}, (\beta_{j+1/2} - \beta) \mathcal{F})_m.
 \end{aligned}$$

The relation (3.6) is obviously true.

We can even use the version of IBU matrices of the form  $(\mathcal{F}, \mathcal{A}, \mathcal{F})_m$  by computing

$$\begin{aligned}
 (5.3) \quad & m^k + \text{IBU} (-\hat{M}(x^k)/\beta, (\mathcal{N}(x^k)(m^k - x^k) + \mathcal{f}(m^k)))/\beta \\
 & = m^k + \text{IBU} (-\hat{M}(x^k)/\beta, (\mathcal{T}, \mathcal{A}, \mathcal{T})_m \cdot m^k + \Phi(m^k) - \mathcal{N}x^k)/\beta.
 \end{aligned}$$

$\mathcal{B}$ , hence  $\hat{\mathcal{B}}$  are  $M$ -matrices. Then  $\hat{M}(x^0)$  is an interval  $M$ -matrix (see (2.5)). We further note  $\mathcal{N} \cong \mathcal{O}$ . In addition we state that

$$\hat{M}(x^0) - \overline{\mathcal{N}(x^0)} = \hat{M}(x^0) - \mathcal{N} = \mathcal{B} + \underline{\Phi'(x^0)}.$$

The latter matrix being an  $M$ -matrix (see (2.5)), condition (3) of Theorem 3.3 is satisfied. The matrix  $\mathcal{G}$  with

$$g_{ij} = \begin{cases} \beta_{\max} + \hat{\phi}_i(x^0) + \alpha_{i-1/2} + \alpha_{i+1/2} - 2\beta, & j = i, \\ -\alpha_{i-1/2}, & j = i - 1, \\ -\alpha_{i+1/2}, & j = i + 1, \\ 0, & \text{otherwise} \end{cases} \quad 1 \leq i \leq n,$$

is an irreducibly diagonally dominant  $L$ -matrix, hence an  $M$ -matrix ([12, Cor. 1 to Thm. 3.11]). From Theorem 2.1 we obtain the applicability of IBU to  $\hat{M}(x^0)$ .

We have verified all conditions of the Theorems 3.2 and 3.3 concerning the matrices related to the problem to be solved. A straightforward generalization of (5.1) is given by

$$\begin{aligned}
 (5.4) \quad & (a(s, t)u_s)_s + (b(s, t)u_t)_t = f(u), \\
 & a(s, t), b(s, t) > 0.
 \end{aligned}$$

In (5.2) we have to write

$$\begin{aligned}
 \mathcal{T}_j &= (-\delta_{ik} \cdot \beta_{i,j+1/2})_{i,k=1}^m, \\
 \mathcal{A}_j &= (-\alpha_{i-1/2,j} \beta_{i,j+1/2} + \beta_{i,j-1/2} + \alpha_{i-1/2,j} + \alpha_{i+1/2,j}) - \alpha_{i+1/2,j} \Big)_m, \\
 \beta_{i,j+1/2} &= b(ih, (j+1/2)h), \quad 1 \leq i \leq m, \quad 0 \leq j \leq m, \\
 \alpha_{i+1/2,j} &= a((i+1/2)h, jh), \quad 0 \leq i \leq m-1, \quad 1 \leq j \leq m.
 \end{aligned}
 \tag{5.5}$$

(5.3) has to be changed by replacing the corresponding definitions:

$$\begin{aligned}
 \beta &:= \min \{ \beta_{i,j+1/2} \mid 1 \leq i \leq m, 0 \leq j \leq m-1 \}, \\
 \tilde{\beta}_i &:= \max \{ \beta_{i,j+1/2} + \beta_{i,j-1/2} \mid 1 \leq j \leq m \}, \quad 1 \leq i \leq m, \\
 \tilde{\alpha}_i &:= \max \{ \alpha_{i-1/2,j} + \alpha_{i+1/2,j} \mid 1 \leq j \leq m \}, \quad 1 \leq i \leq m, \\
 \alpha_{i+1/2} &:= \min \{ \alpha_{i+1/2,j} \mid 1 \leq j \leq m \}, \quad 1 \leq i \leq m-1, \\
 \mathcal{T} &:= -\beta \mathcal{I}, \quad \mathcal{A} := (-\alpha_{i-1/2}, \tilde{\beta}_i + \tilde{\alpha}_i, -\alpha_{i+1/2})_m, \\
 \mathcal{N} &:= (\mathcal{T} - \mathcal{T}_j, \mathcal{A} - \mathcal{A}_j, \mathcal{T} - \mathcal{T}_j)_m.
 \end{aligned}
 \tag{5.6}$$

**6. Numerical experiments.** The numerical experiments were carried out on the CYBER 170-835 (48-bit mantissa) of the ZRZ (Zentraleinrichtung Rechenzentrum) of the Technical University of Berlin with the PASCAL compiler of the ETH Zürich/University of Minnesota.

The interval arithmetic and the interval operations had to be simulated by PASCAL procedures. For the representation of intervals we defined a data type INTERVAL = Grecord *u, o*: real end. The computation time can be reduced by a factor of at least four or five if interval operations and a corresponding arithmetic are implemented on the machine or microprogram level (e.g., like the arithmetic developed and implemented together with the PASCAL-SC compiler at the University of Karlsruhe—see [6]). We therefore divided the computation times measured for interval methods by four in order to compare them at least roughly with “ordinary” floating point methods. We described the details concerning our implementation of the interval operations in [8] and [10]. For our numerical experiments we chose some examples for (5.1):

- (1)  $f(u) = \alpha \cdot e^u, \quad a(s) = s, \quad b(t) = 1 + t^4/1024,$
- (2)  $f(u) = \alpha \cdot u^3, \quad a(s) = s/128, \quad b(t) = 1 + t/1024,$
- (3)  $f(u) = \alpha \cdot u^3 \cdot \frac{1+t^2}{\sqrt{1+s^2}}, \quad a(s) = e^s, \quad b(t) = 10 + e^{-(100/(t+1))},$
- (4)  $f(u) = \alpha \cdot u^3, \quad a(s) = s, \quad b(t) = 100000 + t^4/8192,$   
 $g(s, t) = s + 2t \quad \text{for (1)–(4)}$
- (5)  $f(u) = \alpha \cdot \frac{\sin(s^2u/10) + 2 \sin^2(s^2u^3/20)}{\exp(10^{-5}\{\exp(((u^3+1)/(100(u^{10}+7))) \exp(u^2+1)) - u/4\})},$   
 $b(t) = 1000 + e^{-500t^2}/5000, \quad a(s) = (\sin(s) + \cos(3s))/\pi^2,$   
 $g(s, t) = s - 2t.$

These examples are intended to give an idea of the behavior of our test methods depending on *a, b, f* and the shape of *f* varied by different  $\alpha$ .

The examples were discretized with  $h = 1/31, 1/63, 1/127$  and computed with  $\alpha = 10^0, 10^1, 10^2, 10^3, 10^4$ .

For the variants of IMNAB without auxiliary steps we chose the convergence criterion

$$(6.1) \quad \|q(x^{k+1}, x^k)\|_\infty < \varepsilon = 10^{-6}$$

and for IMNAB+IGS and IMNAB+IS2

$$(6.2) \quad \|d(x^{k+1})\|_\infty < \varepsilon = 10^{-6}.$$

For “ordinary” floating-point methods we stopped any iteration when

$$(6.3) \quad \|q(x^{k+1}, x^k)\|_\infty = \|x^{k+1} - x^k\| < \varepsilon = 10^{-6}.$$

We compared some versions of the method IMNAB with the generalized conjugate gradient method (GCG) as described in [4]. We also add some results for the nonlinear block SOR method (NBSOR) as described in [5]. In view of the great number of possible variants of GCG and IMNAB we restricted our attention to the following:

1. IMNAB without auxiliary steps as defined by (3.3), IMNAB+IGS and IMNAB+IS2 as defined by (3.8). We chose  $m^k := m(x^k)$  for all  $k \in \mathbb{N}_0$ . The starting vector was improved by the procedure described in § 4.
2. IMNABM with  $m^k := \underline{x}^k$  for all  $k \in \mathbb{N}_0$  (version *U*) and with  $m^k := \overline{x}^k$  for all  $k \in \mathbb{N}_0$  (version *O*) with improved starting vector.
3. IMNAB with given, not improved starting vector and  $m^k := m(x^k)$  for all  $k \in \mathbb{N}_0$  (denoted by IMNAB  $[a, b]$  if  $x^0 = ([a, b])_{i=1}^N$ ).

All these variants use the algorithm (2.6) for IBU. Storage restrictions prevented us to compute all the elimination parts of IGA already in step (1): in this case we would have to store  $m$  vectors  $d$  of size  $[1 \dots m] \times \text{INTERVAL}$  and  $m$  vectors  $f$  of size  $[1 \dots m - 1] \times \text{INTERVAL}$  occupying about  $4N$  memory locations. By computing the vectors  $d$  and  $f$  separately for each  $r$ , the additional storage reduces to  $(m - 1)/2$  vectors  $d$  and  $(m - 1)/2$  vectors  $f$ , i.e. approximately  $2N$  locations. On the other hand the recomputation of vectors  $d, f$  in step (4) requires only approximately  $3N/2$  arithmetic operations which is not serious in view of a total number of operations of order  $(5k + 7)N$  of IBU (remember that  $k = 5, 6$ ). Further details concerning the programming of IBU can be found in [10]. For the present applications we only have to replace the special variant of IGA for tridiagonal matrices of the form  $(-1, D_i, -1)$  by a version for matrices of the form  $(A_{i-1}, D_i, A_i)$ . The iteration to improve the starting vector was stopped, when  $\|q(\underline{x}^{k+1}, \underline{x}^k)\|_\infty < 10^{-3}$ . This criterion led to the most satisfying results with respect to the minimization of the total computation time in several previous experiments.

The diagonal elements  $b_{ii}$  needed by IGS are recomputed in each auxiliary step as  $b_{ii} := a_j + c_k$ , where  $a_j = \alpha_{j-1/2} + \alpha_{j+1/2}$ ,  $c_k := \beta_{k-1/2} + \beta_{k+1/2}$ ,  $i = (k - 1)m + j$ ,  $1 \leq j, k \leq m$ . The  $[0 \dots m] \times \text{INTERVAL}$  vectors  $a$  and  $c$  are computed only once before starting IMNAB.

We tested the following variants of GCG:

4. GCG without convergence safeguards where  $\mathcal{M}^{(k)} \equiv \mathcal{M} = (-\mathcal{F}, \mathcal{A}, -\mathcal{F})$ ,  $\mathcal{A} = (-\alpha_{i-1/2}, \alpha_{i-1/2} + \alpha_{i+1/2} + \beta_{\max}, -\alpha_{i+1/2})/\beta$  (variant GCG) respectively  $\mathcal{M}^{(k)} = (-\mathcal{F}, \mathcal{A}^{(k)}, -\mathcal{F})$  with  $\mathcal{A}^{(k)} = \mathcal{A} + (h^2/3)(\phi'_1(x_1) + \phi'_{(N+1)/2}(x_{(N+1)/2}) + \phi'_N(x_N))\mathcal{F}$  (variant GCG+S).
5. GCG with the first  $\mathcal{M}$  in 4. and convergence safeguards as described in [4]: with restarts after 5 steps (variant GCG+R5) respectively 10 steps (variant

GCG+R10) and a line search at each restart to determine the stepsize  $a_k$ . We stopped the Newton iteration when  $|a_k^{(l+1)} - a_k^{(l)}| < 10^{-6}$ .

All GCG variants solve the “defect” equation by a variant of the Buneman algorithm constructed similarly to the interval reduction method given by (2.6). We further always used the parameters  $(a_1, b_1)$  (see [4]). The other three possibilities lead to a higher computation time in most cases in a number of tests.

As initial inclusion for the solution  $\mathcal{I}$  dedicated to start the iteration to improve the starting vector (variants mentioned in 1 and 2) respectively to start immediately IMNAB (3) we chose the vectors  $([-4, 4])_{i=1}^N$  for the examples (2)–(5) and  $([-10, 10])_{i=1}^N$  for example (1). A closer inspection of the examples would lead to even better inclusions: we have  $f(x) \subseteq \varphi \subseteq f(\bar{x})$  for  $x = ([0, 3])_{i=1}^N$  in the examples (2)–(5). For example (1) we note the  $f(\bar{x}) \supseteq \varphi$  for  $\bar{x} := (3)_{i=1}^N$ . A correct lowerbound with  $f(x) \subseteq \varphi$  is somewhat more complicated to determine:

We define for example  $x = (x_i)_{i=1}^N$  by  $x_{i+1} := x_i + \alpha_{i+1}$ ,  $1 \leq i \leq N - 1$ , with  $\alpha_{i+1} = \alpha_i + \beta$ ,  $\alpha_0 := 0$ ,  $x_1 := -10 + \beta$  where, e.g., in the case  $m = 127$  we set  $\beta := 10^{-9}$ .

All variants of GCG were started by  $\varphi^0 := (4)_{i=1}^N$  in all five examples. Some tests with  $\varphi^0 = \varphi$  or  $\varphi^0 = (-4)_{i=1}^N$  gave similar results.

In the following tables we report the results of our numerical experiments for  $m = 63, 127$  (Tables 1–5) and  $m = 31$  (Table 6). Each field contains in its left part the number of iterations needed to satisfy (6.3) for the point methods and (6.1) for IMNAB, IMNABM and IMNAB  $[\cdot, \cdot]$  respectively (6.2) for IMNAB+IGS and IMNAB+IS2. The right part contains the total computation time in seconds including the time for the improvement of the starting vector for the corresponding variants of IMNAB. For IMNAB+IGS we note the number of steps necessary to satisfy (6.2) and below that to satisfy (6.1). The computation time refers to (6.2). Fields marked with an  $M$  (for IMNAB+IS2) indicate that storage restrictions prevented the computation.

In view of the great number of experiments and the necessary amount of computation time we did not compute all variants for all examples and all values of  $\alpha$ . The corresponding rows or fields remain empty. Fields marked by “-1” refer to examples where IMNAB satisfied (6.1) while  $\|d(x^{k+1})\|_\infty \approx 0.1$ .

When comparing IMNAB without convergence safeguards with GCG and GCG + S without convergence safeguards we state that in examples (1)–(3), (5) IMNAB is superior for  $\alpha \geq 10^1$  or  $\alpha \geq 10^2$ , in example (4) for  $\alpha \geq 10^3$  ( $m = 63$ ) and  $\alpha = 10^4$  ( $m = 127$ ). Due to the better approximation of the Jacobi matrix GCG + S reduces the number of steps in a sensible manner for increasing  $\alpha$ . The number of steps needed by IMNAB increases rather slowly. Thus the number of steps—and therefore also the computation time—remains largely inferior even to that of GCG + S for these values of  $\alpha$ .

When the improvement of the starting vector is omitted (IMNAB  $[\cdot, \cdot]$ ) the number of steps does not change or increases only slightly in the present examples. On the other hand this is largely compensated by the time needed for the improvement of the starting vector in the other versions. It seems to be preferable to omit the iterative improvement if the initial inclusions are not extremely poor.

There was, however, one test where an arithmetical overflow occurred when calling the exponential function in the first step with  $\varphi^0 = ([-4, 4])_{i=1}^N$  (example (5)). The iterative improvement gave a better inclusion and therefore smaller (sufficiently small) arguments for exp.

Up to now we only compared versions without convergence safeguards. But we have to remember that the convergence of GCG and IMNAB—except the monotone

TABLE 1  
Example (1).<sup>1</sup>

$\alpha$	$10^0$			$10^1$			$10^2$			$10^3$			$10^4$							
	63	127		63	127		63	127		63	127		63	127						
GCG	15	28	15	119	38	71	35	279	68	125	66	523	107	198	96	754	162	300	146	1147
GCG+S	15	29	17	137	30	57	29	238	51	96	52	415	76	143	90	725	128	240	174	1371
IMNAB	16	45	40	515	13	39	31	423	14	38	30	362	19	46	28	305	26	63	31	335
IMNAB+IS2	16	63		M	13	52		M	14	52		M	16	54		M	18	60		M
IMNAB+IGS	16	56	44	696	13	48			14	49			17	53			21	65	44	383
	9		22		8				11				15				18		22	

TABLE 2  
Example (2).<sup>1</sup>

$\alpha$	$10^0$			$10^1$			$10^2$			$10^3$			$10^4$							
	63	127		63	127		63	127		63	127		63	127						
GCG	17	27	16	112	49	80	55	388	71	115	99	696	120	195	103	723	173	282	183	1294
GCG+S	16	27	18	126	31	54	31	217	52	88	53	372	82	143	96	684	132	223	168	1182
IMNAB	18	48	48	567	20	51	48	564	23	57	47	558	52	119	47	504	59	-1	39	-1
IMNAB+IS2	19	60		M	19	60		M	22	68		M	24	74		M	23	71		M
IMNAB+IGS	19	62	55	751	20	66			23	72			26	79			29	88	47	670
	11		26		14				17				20				26		38	
IMNAB[-4,4]	20	43	53	500	21	44			25	52			36	76			8	-1		
																		17		

<sup>1</sup> For examples 1) and 2) GCG+R5 and GCG+R10 lead to an arithmetic overflow for all  $\alpha$  and  $m = 63, 127$ .

TABLE 3  
Example (3).

$\alpha$	$10^0$		$10^1$		$10^2$		$10^3$		$10^4$			
	63	127	63	127	63	127	63	127	63	127		
$m$	7	7	19	36	48	91	85	161	136	257	128	1036
GCG	7	13	7	58	147	383	715	127	127	127	127	127
GCG+R5	7	21	7	86	185	261	373	118	118	118	118	ov
GCG+R10	7	20	7	84	181	283	ov	ov	ov	ov	ov	ov
GCG+S	—	—	7	59	116	256	—	—	—	—	103	852
IMNAB	7	19	7	86	130	184	24	62	39	101	—	-1 >999
IMNAB+IS2	6	24	—	—	M	M	21	73	21	73	—	M
IMNAB+IGS	6	22	6	93	148	217	22	76	24	342	24	30
	4	4	6	6	11	10	18	6	21	24	21	24
IMNAB[-4, 4]	—	—	7	66	118	184	—	—	—	—	—	34
	—	—	—	—	11	17	22	22	22	247	—	34



TABLE 4  
Example (4).

$\alpha$	$10^0$			$10^1$			$10^2$			$10^3$			$10^4$								
	63	127		63	127		63	127		63	127		63	127							
$m$																					
GCG	4	7	4	28	4	7	4	28	5	8	5	35	7	12	6	42	10	16	10	70	
GCG+R5	4	8	4	32	4	8	4	34	5	10	5	41		>250		>120					
GCG+R10	4	8	4	32	4	8	4	34	5	10	5	41	7	13	7	56		>450		>1800	
IMNAB	3	8	3	38	3	8	3	37	4	10	4	46	4	10	4	46	6	14	6	64	
IMNAB[-4, 4]	—	—	3	28	—	—	3	28	—	—	4	37	—	—	5	44	—	—	—	7	62
IMNAB+IS2	2	8		M	2	8		M	3	10		M	4	13		M	5	16		M	

TABLE 5  
Example (5).

$\alpha$	$10^0$			$10^1$			$10^2$			$10^3$			$10^4$							
	63	127		63	127		63	127		63	127		63	127						
$m$																				
GCG	5	18	5	73	6	21	7	101	21	73	59	856	76	264		>2000	39	135		
GCG+S	5	18	7	101	6	22	9	133	16	58	25	364		>299				>299		
GCG+R5	5	23	5	112		ov		ov		ov		ov		ov		ov		ov		ov
GCG+R10	5	24	5	113	6	28	7	156		>1800		>2000								
IMNAB	3	19	4	107	3	19	4	107	3	19	4	106	4	24	4	107	4	24	4	107

TABLE 6  
 Results for the monotone versions IMNABM ( $m = 31$ , version  $O: \overline{m^k} := \overline{x^k}$ , version  $U: \underline{m^k} := \underline{x^k}$ ).

ex	$\alpha$	$10^0$		$10^1$		$10^2$		$10^3$		$10^4$	
		$O$	$U$	$O$	$U$	$O$	$U$	$O$	$U$	$O$	$U$
1	$O$	13	7	42	22	133	65	—	—	—	—
	$U$	16	9	48	25	158	77	—	—	—	—
2	$O$	25	11	67	32	225	100	—	—	—	—
	$U$	20	9	88	42	225	98	—	—	—	—
3	$O$	7	4	24	14	92	50	—	—	—	—
	$U$	10	6	30	17	118	66	—	—	—	—
4	$O$	3	1.7	3	1.7	3	1.8	4	2.2	7	3.5
	$U$	3	1.8	3	1.7	4	2.2	5	2.5	9	4.3
5	$O$	3	4.5	3	4.5	3	4.5	4	6	6	8.5
	$U$	3	4.5	3	4.5	3	4.5	4	6	6	8.5

versions—has only been proved under the condition that convergence safeguards are included.

The convergence preserving property of IGS or IS2 was needed for  $\alpha = 10^4$  in examples 2 and 3: there IMNAB stopped with  $\|q(x^{k+1}, x^k)\|_\infty < 10^{-6}$ , but  $\|d(x^{k+1})\|_\infty > 10^{-1}$ . When testing (6.2) instead of (6.1) the iteration continued for some steps, but without reducing the diameters noticeably. IMNABS+IGS reduced the diameters at least to  $10^{-3}$  or  $10^{-4}$  when (6.1) was true. But IMNAB+IGS and IMNAB+IS2 converge to the necessary precision when an appropriate criterion is set: we therefore computed these two variants always until (6.1) and (6.2) were true.

In all examples except the one just mentioned we obtain at least  $\|d(x^{k+1})\|_\infty < 10^{-5}$  with IMNAB, IMNAB  $[\cdot, \cdot]$ , and IMNABM when (6.1) was satisfied. Remember that  $d(x^k) \rightarrow \varphi (k \rightarrow \infty) \Leftrightarrow q(x^{k+1}, x^k) \rightarrow \varphi (k \rightarrow \infty)$  if  $y \in x^k$  for all  $k$  (see [2, Appendix A]).

For the above mentioned examples we found a similar behavior when applying GCG and GCG+S: we found a number of components with only two or three correct digits. In this context an advantage of interval methods becomes interesting: due to the inclusion of the solution one can immediately determine the number of correct digits.

When IMNAB converges in a few steps without auxiliary steps, IMNAB+IGS or IMNAB+IS2 lead to an increase of computation time. On the other hand IMNAB+IGS and IMNAB+IS2 do not only guarantee the convergence up to a given precision but can also reduce the total number of steps and therefore the computation time in cases where IMNAB converges rather slowly (see example (2)),  $\alpha = 10^3$  and  $\alpha = 10^4$ , and example (3),  $\alpha = 10^4$ ).

As we already mentioned the version IMNAB+IGS was used in order to be able to compute also the case  $m = 127$  in view of the limited storage available at the CYBER 170-835. The variant IMNAB+IS2 contributes in a more sensible way to the reduction of the diameter of the iterates due to the smaller inclusions of the partial derivatives

and the double-steps. A clear advantage of one of the two methods cannot be deduced from the examples computed here.

Table 6 contains some results for the monotone versions of IMNAB. These variants enabled us to a deeper theoretical analysis. With respect to the practical behaviour, however, they gave no improvement over the variants using the midpoint vector, i.e.  $m^k := m(x^k)$ . For those examples, for which IMNAB converged in a few steps, i.e. (4) and (5), the computation time of the monotone versions remained in the same order. In examples (1)–(3) where the number of steps needed by IMNAB was somewhat higher, the number of steps needed by the monotone versions rapidly grew. Remember in this context that the choice of the midpoint as  $m^k$  has been shown to be optimal for the one-dimensional interval Newton method (see [2, Chap. 7]).

The computation times for the variants of IMNAB with improved starting vector are not always proportional to the number of steps as they include the time for the improvement. The latter decreases for increasing  $\alpha$  in example (1), e.g. This is due to the fact that the starting vector  $([-10, 10])_{i=1}^N$  for the iterative improvement includes  $y$  but not necessarily the vector  $x^0$  given by (4.1), i.e. the interval  $[-10, 10]$  or the interval  $[-10, 10] \cap X_i^{(0)}$  is a better inclusion for  $y_i$  than  $X_i^{(0)}$  for some  $i$ . We illustrate this remark in Table 7.

TABLE 7  
*Computation time for the iterative improvement of the starting vector in IMNAB (example (1),  $m = 127$ ).*

$\alpha = 10^0$		$\alpha = 10^1$		$\alpha = 10^2$		$\alpha = 10^3$		$\alpha = 10^4$	
40	515	31	422	30	362	28	305	31	335
27	116	26	111	13	56	4	18	3	14

Each field contains in the first line the number of steps and the total time needed by IMNAB, in the second line the number of steps needed by the iterative improvement and the computation time (included in the total time). In Table 8 we add some values for the other examples. The given values are valid for all  $\alpha$ .

TABLE 8  
*Number of steps and average computation time for the iterative improvement of the starting vector.*

example	2		3		8	
$m = 63$	11	10	2	2	2	2-3
$m = 127$	26-27	110	2	10	3	18

In Table 8 we only note the number of steps and the computation time necessary for the improvement.

The results for the method GCG with convergence safeguards suffer from the fact that in a great number of cases an arithmetic overflow occurred or the method did not converge in a given time. The reason is given by the Newton method used for the line searches: while carrying out the second or third line search the function whose zero

defines the next approximation for  $a_k$  is already very flat. A typical quotient  $h/h'$  in the iteration  $a_k^{(l+1)} = a_k^{(l)} - h(a_k^{(l)})/h'(a_k^{(l)})$ , where  $h$  stands for the function defining the Newton iteration, was given by  $10^{-6}/10^{-10}$ , i.e.  $|a_k^{(l+1)} - a_k^{(l)}| = O(10^4)$  (machine precision  $3.6 \cdot 10^{-15}$ ). This leads to a very slow convergence or divergence (examples (4) and (5)) or an arithmetic overflow (all examples). We tried to control the Newton iteration but the same phenomena occurred only some steps later. On the other hand the bisection method is slow and costly, in particular when the starting interval is not easy to determine. In the cases where the convergence safeguards worked well (example (3),  $m = 63$  e.g.) we observed a behavior similar to that of the auxiliary steps for IMNAB: the computation time was considerably reduced where the convergence of GCG without safeguards took place only after a large number of steps.

Compared to GCG and IMNAB, the nonlinear block SOR method (NBSOR) converged extremely slowly. Table 9 illustrates this fact for  $m = 31$  and  $\alpha = 1$ .

TABLE 9  
Results for NBSOR for  $\alpha = 10^0$ .

example	$m = 31$		$m = 63$	
	1	111	40	231
2	133	86	272	660
3	123	120	247	1002
4	134	83	276	648
5	134	393	—	—

**Conclusion.** The interval method IMNAB is certainly not an all-purpose method and the generalized CG method is suitable for a greater variety of applications. But IMNAB turned out to be an attractive alternative in particular in those cases in which the approximation of the Jacobi matrix by a matrix suitable for a direct solver is rather difficult. Keeping in mind the definition of IMNAB, we note that the “right-hand side” of IBU in IMNAB depends on the function  $b$ . The convergence speed depends on the diameters of this “right-hand side”, i.e. we obtain the best convergence results for “sufficiently” flat functions  $b$ . For suitable functions  $a$  it may be therefore interesting to exchange  $a$  and  $b$ . On the other hand the quality of the approximation of the Jacobi matrix for the present variants of GCG also diminishes when the variation of  $b$  grows. We finally note that reliable software for IMNAB (as well as for GCG) should include suitable measures to guarantee convergence, like in IMNAB+IGS, IMNAB+IS2 or IMNABHIN, e.g.

A further development of the methods presented in [8] and this paper will include for example algorithms replacing IBU by methods dedicated to the solution of more general elliptic equations, 3D problems or other boundary conditions (Neumann, periodical).

REFERENCES

[1] G. ALEFELD, *Über die Durchführbarkeit des Gaußschen Algorithmus bei Gleichungen mit Intervallen als Koeffizienten*, Computing, Suppl. 1 (1977), pp. 15-19.  
 [2] G. ALEFELD AND J. HERZBERGER, *Einführung in die Intervallrechnung*, Bibliographisches Institut, Mannheim-Wien-Zürich, 1974.

- [3] L. BERS, F. JOHN AND M. SCHECHTER, *Partial Differential Equations*, Wiley Interscience, New York-London-Sydney, 1964.
- [4] P. CONCUS, G. GOLUB AND D. O'LEARY, *Numerical solution of nonlinear elliptical partial differential equations by a generalized conjugate gradient method*, *Computing*, 19 (1978), pp. 321-339.
- [5] L. A. HAGEMAN AND T. A. PORSCHING, *Aspects of nonlinear successive overrelaxation*, *SIAM J. Numer. Anal.*, 12 (1975), pp. 316-335.
- [6] U. KULISCH AND CH. ULLRICH, eds., *Wissenschaftliches Rechnen und Programmiersprachen*, German chapter of the ACM/Berichte 10, Teubner-Verlag, Stuttgart 1982.
- [7] J. M. ORTEGA AND W. C. RHEINBOLDT, *Iterative Solution of Nonlinear Equations in Several Variables*, Academic Press, New York and London, 1970.
- [8] H. SCHWANDT, *An interval arithmetic approach for the construction of an almost globally convergent method for the solution of the nonlinear Poisson equation on the unit square*, this Journal, 5 (1984), pp. 427-452.
- [9] ———, *A symmetric iterative interval method for the solution of systems of nonlinear equations*, *Computing*, to appear.
- [10] ———, *Schnelle fast global konvergente Verfahren für die Fünf-Punkt-Diskretisierung der Poissongleichung mit Dirichletschen Randbedingungen auf Rechteckgebieten*, Doctoral dissertation, Technische Universität Berlin, 1981.
- [11] W. TÖRNIG, *Monoton konvergente Iterationsverfahren zur Lösung nichtlinearer Differenzen-Randwertprobleme*, *Beiträge Numer. Math.*, 4 (1975), pp. 245-257.
- [12] R. D. VARGA, *Matrix Iterative Analysis*, Prentice-Hall, Englewood Cliffs, NJ, 1962.

## ACCELERATING AN ITERATIVE PROCESS BY EXPLICIT ANNIHILATION\*

DENNIS C. JESPERSEN† AND PIETER G. BUNING‡

**Abstract.** A slowly convergent stationary iterative process can be accelerated by explicitly annihilating (i.e., eliminating) the dominant eigenvector component of the error. The dominant eigenvalue or complex pair of eigenvalues can be estimated from the solution during the iteration. The corresponding eigenvector or complex pair of eigenvectors can then be annihilated by applying an explicit Richardson process over the basic iterative method. This can be done entirely in real arithmetic by analytically combining the complex conjugate annihilation steps. We illustrate by applying the technique to an implicit algorithm for the calculation of two-dimensional steady transonic flow over a circular cylinder using the equations of compressible inviscid gas dynamics. This demonstrates the use of explicit annihilation on a nonlinear problem.

**Key words.** convergence acceleration, nonstationary iteration, extrapolation to the limit

**AMS (MOS) subject classifications.** primary 65B05; secondary 65F10, 65H10, 65N20

**1. Introduction.** A great deal of work is being done to develop efficient algorithms for steady-state problems in computational fluid dynamics. The most widely used class of algorithms for steady problems consists of time-like methods that are marched to a steady state. Our purpose here is to point out a simple and easily applied technique from linear algebra which can in some cases produce a remarkable speedup in algorithms that march to the steady state. The idea may be briefly summarized as follows. In the later stages of the marching algorithm, the iteration may be behaving linearly. If so, one can estimate the dominant eigenvalue of the underlying iteration matrix and “annihilate” the eigenvector corresponding to this eigenvalue by a special iteration step. Even when the eigenvalue and eigenvector are complex, all the computations can be performed in real arithmetic.

Some notation is developed and some facts from linear algebra are reviewed in the next section. A particular algorithm for the Euler equations is presented in § 4, and an application of annihilation to steady transonic flow over a circular cylinder is described in § 5. We wish to emphasize that the application of annihilation is not restricted to the particular marching algorithm that we use for illustrative purposes.

**2. Review of linear algebra.** In this section we establish our notation and review some elementary facts from numerical linear algebra. To begin, consider the matrix problem  $Ax = b$ , where  $A$  is nonsingular. Let  $A = M - N$  be a splitting of  $A$ , where  $M$  is nonsingular, and consider the iteration

$$\begin{aligned} & Mx^{n+1} = Nx^n + b, \quad n \geq 0, \quad x^0 \text{ given,} \\ (1) \quad & \text{or } M\Delta x^n = -Ax^n + b, \\ & \text{and } x^{n+1} := x^n + \Delta x^n. \end{aligned}$$

(The notation “ $a := b$ ” is used here to mean that  $a$  is defined as  $b$ .)

---

\* Received by the editors November 12, 1982, and in revised form September 15, 1983.

† NASA Ames Research Center, Moffett Field, California 94035. This work performed while this author was a member of the Professional Staff, Informatics General Corporation, Palo Alto, California. The work was supported by NASA Contract NAS2-9891.

‡ Research Scientist, Computational Fluid Dynamics Branch, NASA Ames Research Center, Moffett Field, California 94035.

Let  $x^*$  satisfy  $Ax^* = b$ , and denote the error by  $e^n := x^* - x^n$ . Then  $Me^{n+1} = Ne^n$ , and so the iteration converges if and only if the spectral radius of  $M^{-1}N$  is less than 1,  $\rho(M^{-1}N) < 1$ .

Suppose now that  $M^{-1}N$  is diagonalizable with eigenvalues  $\lambda_i$  and corresponding eigenvectors  $v_i$ . Take  $|\lambda_1| \geq |\lambda_2| \geq \dots \geq |\lambda_m|$ , and write  $e^n = \sum_{i=1}^m \alpha_i v_i$ . Then we clearly have

$$(2) \quad e^{n+k} = \sum_{i=1}^m \alpha_i \lambda_i^k v_i = \lambda_1^k \left( \alpha_1 v_1 + \sum_{i=2}^m \alpha_i \left( \frac{\lambda_i}{\lambda_1} \right)^k v_i \right).$$

Evidently the component of the error in the direction of  $v_1$  is the slowest decreasing component (if  $|\lambda_1| > |\lambda_2|$ ).

The component of the error in any direction  $v_j$  can be annihilated as follows. After computing  $\Delta x^{n+k}$ , define  $x^{n+k+1} := x^{n+k} + \sigma \Delta x^{n+k}$ , where  $\sigma$  is as yet arbitrary. (We call this a Richardson step with parameter  $\sigma$ .) Then we have

$$(3) \quad \begin{aligned} e^{n+k+1} &= e^{n+k} - \sigma \Delta x^{n+k} \\ &= e^{n+k} - \sigma M^{-1}(-Ax^{n+k} + b) \\ &= e^{n+k} - \sigma M^{-1}Ae^{n+k} \\ &= [I - \sigma M^{-1}(M - N)]e^{n+k} \\ &= [(1 - \sigma)I + \sigma M^{-1}N] \sum_i \alpha_i \lambda_i^k v_i \\ &= \sum_i \alpha_i \lambda_i^k [(1 - \sigma) + \sigma \lambda_i] v_i. \end{aligned}$$

Thus if we choose  $\sigma = 1/(1 - \lambda_j)$ , the component of  $e^{n+k+1}$  in the direction of  $v_j$  is zero. Any component  $v_j$  can in theory be annihilated, but in practice the important component is the eigenvector  $v_1$  associated with the dominant eigenvalue  $\lambda_1$ .

Of course we do not know  $\lambda_1$ , but we can use the sequence  $\Delta x^n$  to estimate it, since  $\Delta x^{n+1} = M^{-1}N\Delta x^n$ . Estimating  $\lambda_1$  from the sequence  $\Delta x^n$  is simply using the power method to find the dominant eigenvalue of a matrix. Thus we estimate  $\lambda_1$  by  $\lambda_1 \approx (\Delta x^{n+k})_r / (\Delta x^{n+k-1})_r$  for some appropriate  $r$ th component of the update  $\Delta x$ . (The residual  $r^{n+k} := -Ax^{n+k} + b$  could also be used for the estimation, since  $r^{n+k} = Ae^{n+k}$ .) In summary, we plan to estimate  $\lambda_1$  from the sequence  $\Delta x^n$  and take a Richardson step with parameter  $\sigma = 1/(1 - \lambda_1)$ .

The general condition for stability of the Richardson step with parameter  $\sigma$  is  $|1 - \sigma(1 - \lambda_i)| \leq 1$  for all eigenvalues  $\lambda_i$  of  $M^{-1}N$ . This ensures that no component of the error is amplified by the Richardson step. It may well happen that the Richardson step we propose is “unstable” in the sense that some components of the error are magnified. We note, however, that in the later stages of an iteration the subdominant components of the error are liable to be so small that a certain amplification of them is acceptable when accompanied by a large decrease in the dominant component of the error. Also, further ordinary steps will rapidly reduce these subdominant components again. To be more specific, suppose  $e^{n+k} = \alpha_1 v_1 + \varepsilon v_i$ ,  $i \neq 1$ . Then we easily find

$$(4) \quad \begin{aligned} e^{n+k+1} &= e^{n+k} - \sigma M^{-1}Ae^{n+k} \\ &= \alpha_1(1 - \sigma + \sigma \lambda_1) v_1 + \varepsilon(1 - \sigma + \sigma \lambda_i) v_i \\ &= \varepsilon \frac{\lambda_i - \lambda_1}{1 - \lambda_1} v_i \quad \text{if } \sigma = \frac{1}{1 - \lambda_1}. \end{aligned}$$

Thus, the  $v_i$  error component is multiplied by a factor of  $(\lambda_i - \lambda_1)/(1 - \lambda_1)$ . We expect  $|\lambda_1|$  to be close to 1, so this factor could be large, but the factor of  $\varepsilon$  will compensate if it is small enough.

We remark here that in our application,  $A$  is not constant, but  $A = A(x^n)$ . Nevertheless, the assumption of linear behavior will be approximately satisfied in the later stages of the iteration. We will not attempt annihilation unless the iteration seems to be behaving linearly. An empirical criterion for determining this is described in § 5.

We will now show how annihilation can proceed in real arithmetic even when the dominant eigenvalue is complex. It is quite possible that the dominant eigenvalue could be complex, and thus (since we will assume that  $A$  is real) that there is a dominant pair of complex conjugate eigenvalues, say  $\lambda_2 = \bar{\lambda}_1$ , and  $|\lambda_1| = |\lambda_2| > |\lambda_3| \cong \dots \cong |\lambda_m|$ . Then if we write  $\Delta x^n = \sum_{i=1}^m \beta_i v_i$ , we see

$$\Delta x^{n+k-p} = \lambda_1^{k-p} \beta_1 v_1 + \lambda_2^{k-p} \beta_2 v_2 + O(|\lambda_3|^{k-p})$$

for  $p=0, 1, 2$ . Thus, for any real  $c$  and  $d$ ,

$$(5) \quad \begin{aligned} \Delta x^{n+k} + c\Delta x^{n+k-1} + d\Delta x^{n+k-2} \\ = \lambda_1^{k-2} \beta_1 (\lambda_1^2 + c\lambda_1 + d)v_1 + \lambda_2^{k-2} \beta_2 (\lambda_2^2 + c\lambda_2 + d)v_2 + O(|\lambda_3|^{k-2}). \end{aligned}$$

The coefficients of  $v_1$  and  $v_2$  will vanish if  $c = -2 \operatorname{Re} \lambda_1$  and  $d = |\lambda_1|^2$ , for then  $\lambda^2 + c\lambda + d = (\lambda - \lambda_1)(\lambda - \lambda_2)$ .

In practice, we can pick indices  $i \neq j$  and find  $c$  and  $d$  such that

$$(6) \quad \begin{aligned} (\Delta x^{n+k})_i + c(\Delta x^{n+k-1})_i + d(\Delta x^{n+k-2})_i &= 0, \\ (\Delta x^{n+k})_j + c(\Delta x^{n+k-1})_j + d(\Delta x^{n+k-2})_j &= 0. \end{aligned}$$

This is a linear system for  $c$  and  $d$ . Once  $c$  and  $d$  are known,  $\lambda_1 = -c/2 + i\sqrt{d - c^2/4}$ , and two Richardson steps can be performed by defining  $\sigma := 1/(1 - \lambda_1)$  and putting

$$(7) \quad \begin{aligned} x^{n+k+1} &:= x^{n+k} + \sigma \Delta x^{n+k}, \\ x^{n+k+2} &:= x^{n+k+1} + \bar{\sigma} \Delta x^{n+k+1}, \end{aligned}$$

where  $\Delta x$  is still given by one step of the original iterative scheme, namely,  $\Delta x^{n+k} := M^{-1}(-Ax^{n+k} + b)$  and  $\Delta x^{n+k+1} := M^{-1}(-Ax^{n+k+1} + b)$ . By the calculations done previously, this will eliminate the  $v_1$  and  $v_2$  components of the error. Furthermore, there is no need for complex arithmetic, for the computation can be reorganized as follows:

$$(8) \quad \begin{aligned} M(x^{n+k+2} - x^{n+k}) &= M(\bar{\sigma} \Delta x^{n+k+1} + \sigma \Delta x^{n+k}) \\ &= \bar{\sigma}(-Ax^{n+k+1} + b) + \sigma(-Ax^{n+k} + b) \\ &= \bar{\sigma}[-A(x^{n+k} + \sigma \Delta x^{n+k})] + \bar{\sigma}b - \sigma Ax^{n+k} + \sigma b \\ &= -(\sigma + \bar{\sigma})Ax^{n+k} + (\sigma + \bar{\sigma})b - |\sigma|^2 AM^{-1}(-Ax^{n+k} + b) \\ &= 2 \operatorname{Re} \sigma r^{n+k} - |\sigma|^2 AM^{-1} r^{n+k}. \end{aligned}$$

Thus, two complex Richardson steps can be performed entirely in real arithmetic, at a cost of 1) solving two linear systems with coefficient matrix  $M$  and 2) performing one matrix-vector multiplication with coefficient matrix  $A$ . This is the same as the cost of two steps of the basic iteration process. As for stability, a component of  $e^{n+k}$  in the direction of the vector  $v_j, j > 2$ , is multiplied by the factor

$$(9) \quad \frac{(\lambda_j - \lambda_1)(\lambda_j - \bar{\lambda}_1)}{(1 - \lambda_1)(1 - \bar{\lambda}_1)}.$$



Again, this factor could be large, but we expect to apply the annihilation in the later stage of the iteration when the coefficients of  $v_j$  are very small and some amplification would be tolerable.

It seems to be more robust to solve a linear least squares problem of the form

$$\begin{pmatrix} \vdots & \vdots & \cdots & \vdots \\ P\Delta x^{n+k-1} & P\Delta x^{n+k-2} & \cdots & P\Delta x^{n+k-r} \\ \vdots & \vdots & \cdots & \vdots \\ \vdots & \vdots & \cdots & \vdots \end{pmatrix} \begin{pmatrix} c_{r-1} \\ c_{r-2} \\ \vdots \\ c_0 \end{pmatrix} = \begin{pmatrix} \vdots \\ -P\Delta x^{n+k} \\ \vdots \\ \vdots \end{pmatrix},$$

for coefficients  $c_0, \dots, c_{r-1}$ . Here  $P$  is a projection operator of modest rank. After solving the least squares problem, one finds the roots of the polynomial  $x^r + c_{r-1}x^{r-1} + \dots + c_0$  and chooses the appropriate real root or complex pair of roots for annihilation.

We have found it convenient and useful in practice to reformulate the double annihilation step as follows, where we introduce an arbitrary nonzero parameter  $\alpha$ :

$$\begin{aligned} (10) \quad x^{n+k+1} &:= x^{n+k} + \frac{|\sigma|^2}{\alpha} M^{-1} r^{n+k}, \\ r^{n+k+1} &:= -Ax^{n+k+1} + b, \\ x^{n+k+2} &:= x^{n+k} + M^{-1} [(2 \operatorname{Re} \sigma - \alpha) r^{n+k} + \alpha r^{n+k+1}]. \end{aligned}$$

This results in the same  $x^{n+k+2}$  as (8), but it may have an advantage in the nonlinear case  $A = A(x^n)$ . The reason for this is as follows. In the nonlinear case, if we choose  $\alpha = |\sigma|^2$ , then  $x^{n+k+1}$  is given by one step of the usual algorithm, and since we expect to be in the ‘‘linear’’ regime where  $\|x^{n+k+1} - x^{n+k}\|$  is small, we expect that  $A(x^{n+k})$  and  $A(x^{n+k+1})$  do not differ significantly. On the other hand, if we choose  $\alpha = 2 \operatorname{Re} \sigma$ , then the first step of (10) can involve a large change in  $x$ , and this could invalidate the linearity assumption.

For a general iteration

$$x^{n+1} := x^n + \mathcal{G}(x^n),$$

where a solution  $x^*$  of  $\mathcal{G}(x^*) := 0$  is sought, the preceding remarks suggest that complex annihilation be performed as follows:

$$\begin{aligned} x^{n+1} &:= x^n + \mathcal{G}(x^n), \\ x^{n+2} &:= x^n + [(2 \operatorname{Re} \sigma - |\sigma|^2) \mathcal{G}(x^n) + |\sigma|^2 \mathcal{G}(x^{n+1})]. \end{aligned}$$

This formulation has the nice feature that no linearizations are required; only  $\mathcal{G}$  need be evaluated. This is useful if the evaluation of  $\mathcal{G}$  is done by a complicated algorithm which may be difficult or impossible to linearize correctly. Of course, if  $\mathcal{G}(x) = M^{-1}(-Ax + b)$ , then we recover (8).

We might remark that the presence of a dominant complex conjugate pair of eigenvalues can be signaled by oscillations in the norm of the residual. For example, if  $r^0 = \alpha_1 Av_1 + \alpha_2 Av_2 = 2 \operatorname{Re} (\alpha_1 Av_1)$ , then, writing  $\lambda_1 = \rho e^{i\theta}$ , we see that in any norm

$$\begin{aligned} (11) \quad \rho^{-k} \|r^k\| &= \|\alpha_1 e^{ik\theta} Av_1 + \bar{\alpha}_1 e^{-ik\theta} A\bar{v}_1\| \\ &= \|2 \operatorname{Re} (\alpha_1 e^{ik\theta} Av_1)\|. \end{aligned}$$

Thus, the vector  $w := \alpha_1 Av_1$  is rotated through an angle  $k\theta$  and projected on the real axis, which may give oscillations. The oscillations need not be visible in the  $l_2$  norm, however, for the different components of  $w$  (which will oscillate) may oscillate out of

phase with one another, and the averaging of the  $l_2$  norm may obliterate the phase information. For example, if  $w = (\sigma_1 e^{i\phi_1}, \dots, \sigma_m e^{i\phi_m})^T$ , then

$$\|\operatorname{Re}(e^{ik\theta} w)\|_2^2 = \sum_{j=1}^m \sigma_j^2 \cos(\phi_j + k\theta)^2;$$

so if  $m$  is large and the phases  $\{\phi_j\}$  are independent random variables uniformly distributed in the interval  $[0, 2\pi)$  then the expected value of  $\|\operatorname{Re}(e^{ik\theta} w)\|_2^2$  is  $(\sum \sigma_j^2)/2$  and no oscillations will be visible. However, the projection of the residual  $r^k$  onto a "small"-dimensional subspace should give visible oscillations in the  $l_2$  norm; i.e., if  $P$  is a low-rank projection operator, then  $\|Pr^k\|_2$  should exhibit oscillations. Furthermore, if  $\theta = \pi p/q$  in lowest terms, then the period of the oscillation will be  $q$ . Examples of oscillatory decreasing residuals (which signal the presence of a dominant complex conjugate eigenvalue pair) can be found in [1]–[3].

It is sometimes possible to use the oscillations in  $\|r^k\|_2$  (or  $\|Pr^k\|_2$ ) to estimate the maximum eigenvalue  $\lambda_1 = \rho e^{i\theta}$ . If  $\|r^k\|_2$  has successive maxima at  $k = k_1$  and  $k = k_2$ , then  $\rho$  is given by  $\rho = (\|r^{k_2}\|_2 / \|r^{k_1}\|_2)^{1/(k_2 - k_1)}$  and  $\theta$  is given (approximately) by  $\theta = \pi / (k_2 - k_1)$ .

We close this section with some remarks on the relation of annihilation to previous work. It is easy to show that in the case of a scalar sequence  $(x^0, x^1, \dots, x^n)$ , our "annihilation" step is, exactly, an Aitken  $\delta^2$ -extrapolation step. Wilkinson [4] discusses the application of the Aitken  $\delta^2$  technique to the power method for estimating eigenvalues and eigenvectors, where  $x^n$  is a vector. The formulas there are different from the ones we have used: in Wilkinson's formulation the Aitken  $\delta^2$  acceleration step is applied to each component of the vector separately (in effect, an "eigenvalue" is assumed for each separate component of the sequence of vectors), whereas in our formulation a single eigenvalue is estimated and used for all components of the vector. We are unable to say which strategy is better. Furthermore, in Wilkinson's formulation it is unclear how to handle the case of a dominant complex conjugate pair of eigenvalues (it is shown how to estimate the eigenvalues as roots of a quadratic equation, but the acceleration of the convergence of the vectors is not discussed). The idea of annihilation is elementary but appears to have been neglected. Lyusternik [5] considered a procedure that amounts to annihilation in the case in which the eigenvalues are all real, but he did not discuss the complex case. Hyman and Manteuffel [6] describe an algorithm very similar to annihilation for accelerating slowly convergent iterative methods. They use a Krylov sequence technique to obtain estimates of the eigenvalues of the iteration matrix, then estimate an ellipse in the complex plane which contains these eigenvalues and apply Chebyshev acceleration techniques in the complex plane.

Many authors have been concerned with the harder problem of optimizing relaxation schemes (getting all the eigenvalues of the iteration matrix as far inside the unit circle as possible). An adaptive procedure for minimizing the norm of the iteration matrix was presented by Manteuffel [7]; he estimated complex eigenvalues by the same technique we use above. To reiterate our point of view: we do not ask for the "optimum" method in some class of methods, nor do we insist that our annihilation steps be stable. We simply estimate eigenvalues and perform annihilation. Moreover, we are willing to accept amplification of some error components in the annihilation step; these components will be rapidly reduced anyway in subsequent normal iteration steps.

**3. A test of annihilation.** In this section we will test the application of annihilation to accelerate convergence for a realistic nonlinear problem. The example problem is nonlinear and small enough that the eigenvalues of any frozen coefficient problem can

be computed by standard eigenvalue software. We study the steady quasi-one-dimensional Euler equations

$$\frac{\partial E(Q, x)}{\partial x} + H(Q, x) = 0, \quad 0 < x < 1.$$

Here  $Q = Q(x)$  is the three-vector  $Q = a(\rho, \rho u, e)^T$ , where  $\rho$  is the density,  $u$  the velocity,  $e$  the total energy per unit volume, and  $a = a(x)$  the cross-sectional area of a channel or nozzle. These equations can be derived by averaging the two-dimensional inviscid compressible ideal gas dynamics equations with respect to  $y$  in the channel or nozzle. The nonlinear function  $E$  is given by  $E(Q, x) = a(x)(\rho u, \rho u^2 + p, u(e + p))^T$ , and the forcing term  $H$  is given by  $H(Q, x) = (0, -a'(x)p, 0)^T$ , where the pressure  $p = p(Q)$  is given by  $p = (\gamma - 1)(e - \rho u^2/2)$  and  $\gamma$  is a constant. Appropriate boundary values must be adjoined to complete the specification of the problem. We chose to study the case of a diverging nozzle ( $a'(x) > 0$ ) with supersonic inflow and subsonic outflow conditions, with an inflow Mach number of 1.26; we took  $a(x) = 1.398 + 0.347 \tanh(0.8(x - 5))$ . For this case the exact solution  $Q$  has a shock. We could of course use the large body of work that has been done for one-dimensional boundary value problems, but we are interested in this problem as a stepping-stone for multi-dimensional problems, so we will treat it similarly to two-dimensional problems.

In order to better resolve the shock, an invertible mapping  $\xi = \xi(x)$  is constructed such that a uniform placement of grid points in  $\xi$  space gives a grid with clustering near the shock in  $x$  space. Under this mapping the equations take the form

$$\frac{\partial E(Q)}{\partial \xi} + x'(\xi)H(Q, \xi) = 0.$$

A standard implicit algorithm [9] begins with the time-dependent form of these equations, uses the implicit Euler method for time discretization and then linearizes the nonlinear equations of the implicit Euler method about the solution at the previous time step (see [9] for details). The result is the algorithm

$$\left( I + h_n \delta_\xi \frac{\partial E(Q^n)}{\partial Q} + h_n x'(\xi) \frac{\partial H(Q^n)}{\partial Q} \right) \Delta Q^n = -h_n (\delta_\xi E(Q^n) + x'(\xi)H(Q^n)),$$

$$Q^{n+1} = Q^n + \Delta Q^n.$$

Here the superscript  $n$  denotes the time level,  $h_n$  denotes the time step (which can vary with  $n$ ) and  $\delta_\xi$  denotes a spatial difference operator approximating  $\partial/\partial \xi$ . Here we will consider central differencing,  $\delta_\xi v_j := (v_{j+1} - v_{j-1})/2\Delta \xi$ . With this choice of differencing, “smoothing” operators are usually added to both sides of the equation, giving the algorithm

$$\left( I + h_n \delta_\xi \frac{\partial E(Q^n)}{\partial Q} + h_n x'(\xi) \frac{\partial H(Q^n)}{\partial Q} + h_n \varepsilon_i \Delta_\xi \nabla_\xi \right) \Delta Q^n$$

$$= -h_n (\delta_\xi E(Q^n) + x'(\xi)H(Q^n) + \varepsilon_e (\Delta_\xi \nabla_\xi)^2 Q^n),$$

where  $\Delta_\xi$  and  $\nabla_\xi$  are first-order forward and backward difference operators, respectively, and  $\varepsilon_i \geq 0$  and  $\varepsilon_e \geq 0$  are “smoothing coefficients.” For a given CFL number  $\nu$ , the time step  $h_n$  is defined by  $h_n = \nu x'(\xi_j) / (|u_j^n| + c_j^n)$ , where  $|u_j^n| + c_j^n = \max_i (|u_i^n| + c_i^n)$ ,  $c$  is the speed of sound ( $c^2 = \gamma p/\rho$ ) and  $\Delta \xi$  has been chosen equal to 1. We take the initial condition  $Q^0$  to have each component a linear function of  $\xi$ . The case we will show used 21 mesh points. The computations for this one-dimensional problem were done on a VAX 11/780 in single precision (23 bit mantissa).

For the eigenvalue estimation, we used the least squares idea outlined above with a further refinement (also used by Hyman and Manteuffel [6]). We compute a lambda over each of a number of time steps, use a weighted average of these lambdas as a proposed lambda for annihilation, and perform annihilation if each of the lambdas is within a certain specified tolerance of the proposed lambda. For the results to follow, a lambda was computed for each of 12 time steps.

Our first example had a CFL number of 10 and smoothing coefficients  $(\epsilon_e, \epsilon_i) = (0.5, 1.0)$ . First we ran the implicit algorithm for 300 steps without any annihilation. The resulting  $l_2$ -norm of  $\Delta Q$  is shown by the solid line in Fig. 1a. Next the annihilation strategy was employed, with lambdas estimated over 12 time steps and averaged to determine a proposed lambda for annihilation. The results are shown by the dotted line in Fig. 1a. As shown in the figure, four annihilation steps were performed and the single-precision roundoff level was reached in about 200 iterations.

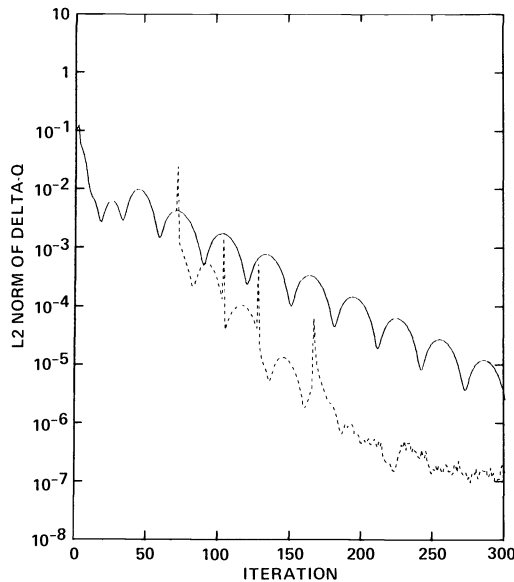


FIG. 1a. Annihilation for steady one-dimensional Euler equations, free-stream Mach number = 1.26, CFL number = 10, smoothing coefficients  $(\epsilon_e, \epsilon_i) = (0.5, 1.0)$ .

For our next example, we used a CFL number of 2 and smoothing coefficients  $(\epsilon_e, \epsilon_i) = (0.1, 0.2)$ . The comparison of the algorithm without and with annihilation is shown in Fig. 1b. As can be seen, seven annihilation steps were performed, and the norm of  $\Delta Q$  after 1200 iterations is about a factor of 50 lower when annihilation was applied. For this problem the dominant eigenvalues at step 600 were found by numerical software to be  $.99497 \pm .05790i$ ,  $.99429 \pm .02187i$ ,  $.96937 \pm .08277i$ , and  $.95585 \pm .01409i$ . This problem, with its eigenvalues very close to 1, is a severe test of annihilation.

**4. An iterative method for the Euler equations.** In this section we will consider a two-dimensional problem. The governing equations are the steady two-dimensional Euler equations of compressible inviscid gas dynamics, which express the conservation of mass, momentum and energy. They can be written in the form

$$(12) \quad \frac{\partial E(Q)}{\partial x} + \frac{\partial F(Q)}{\partial y} = 0,$$

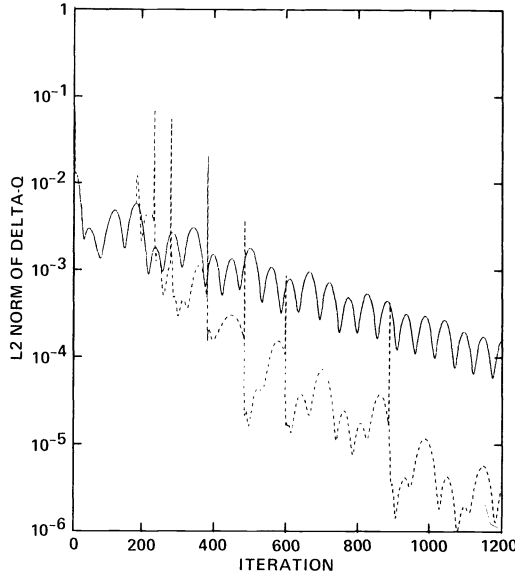


FIG. 1b. Annihilation for steady one-dimensional Euler equations, free-stream Mach number = 1.26, CFL number = 2, smoothing coefficients  $(\epsilon_s, \epsilon_i) = (0.1, 0.2)$ .

where  $Q$  is the four-vector  $Q = (\rho, \rho u, \rho v, e)^T$ . Here  $\rho$  is density,  $u$  and  $v$  are Cartesian velocity components and  $e$  is total energy per unit volume. The functions  $E, F: D \subset R^4 \rightarrow R^4$  are nonlinear functions given by

$$E(Q) = (\rho u, \rho u^2 + p, \rho uv, u(e + p))^T, \quad F(Q) = (\rho v, \rho v^2 + p, v(e + p))^T.$$

The pressure  $p$  is defined by  $p = (\gamma - 1)(e - \frac{1}{2}\rho(u^2 + v^2))$ , where  $\gamma$  is a constant. Appropriate boundary conditions must be adjoined to the differential equation to complete the specification of the problem. It is not important for our purposes what these boundary conditions are, so we will not discuss them further. In order to handle a curved geometry, we map from the physical  $(x, y)$  domain to a “computational”  $(\xi, \eta)$  domain; the computational domain is usually taken to be a rectangle, for ease in differencing. In the  $(\xi, \eta)$  coordinates, the transformed equations retain the strong conservation law form of (2) [8], becoming

$$(13) \quad \frac{\partial \hat{E}(\hat{Q})}{\partial \xi} + \frac{\partial \hat{F}(\hat{Q})}{\partial \eta} = 0.$$

Iterative procedures for solving the steady-state equations (13) often take as their starting point the unsteady equations

$$(14) \quad \frac{\partial \hat{Q}}{\partial t} + \frac{\partial \hat{E}(\hat{Q})}{\partial \xi} + \frac{\partial \hat{F}(\hat{Q})}{\partial \eta} = 0.$$

If we choose Euler implicit differencing in time (hoping for good stability properties) we get the iteration

$$(15) \quad \hat{Q}^{n+1} = \hat{Q}^n - h(\delta_\xi \hat{E}^{n+1} + \delta_\eta \hat{F}^{n+1}),$$

where  $\delta_\xi, \delta_\eta$  are spatial difference operators,  $h := \Delta t$ , and  $n$  denotes a time level. To

avoid iteration on the nonlinear flux terms,  $\hat{E}^{n+1}$  and  $\hat{F}^{n+1}$  on the right-hand side of (15) are expanded about values at level  $n$  using a Taylor series

$$(16) \quad \hat{E}^{n+1} = \hat{E}^n + \left. \frac{\partial \hat{E}}{\partial \hat{Q}} \right|^n \Delta \hat{Q}^n + O((\Delta \hat{Q}^n)^2), \quad \Delta \hat{Q}^n = \hat{Q}^{n+1} - \hat{Q}^n.$$

The locally linearized form of (15) can then be written in ‘‘delta’’ form as

$$(17) \quad [I + h(\delta_\xi \hat{A}^n + \delta_\eta \hat{B}^n)] \Delta \hat{Q}^n = -h(\delta_\xi \hat{E}^n + \delta_\eta \hat{F}^n),$$

where  $\hat{A}^n$  and  $\hat{B}^n$  are the Jacobian matrices  $\partial \hat{E}^n / \partial \hat{Q}^n$  and  $\partial \hat{F}^n / \partial \hat{Q}^n$ , respectively.

Each step of (17) involves the inversion of a large block-banded matrix, with half-bandwidth equal to the number of points in one direction of the mesh. The amount of work required for this is unacceptably large; instead the matrix is approximately factored [9]. For example, if  $\delta_\xi$  and  $\delta_\eta$  were three-point central difference operators, the left-hand side of (16) might be factored via

$$(18) \quad [I + h(\delta_\xi \hat{A}^n + \delta_\eta \hat{B}^n)] = (I + h\delta_\xi \hat{A}^n)(I + h\delta_\eta \hat{B}^n) + O(h^2)$$

as the product of two block-tridiagonal matrices.

The calculations we will show come from an algorithm that uses flux vector splitting ([10], [11]), which is based on separating positive and negative characteristic directions to allow the use of one-sided spatial difference operators. The Euler equations (14) are hyperbolic, and  $\hat{E} = \hat{A}\hat{Q}$  can be decomposed as

$$(19) \quad \begin{aligned} \hat{E} = \hat{A}\hat{Q} &= X\Lambda X^{-1}\hat{Q} = X(\Lambda^+ + \Lambda^-)X^{-1}\hat{Q} \\ &= (\hat{A}^+ + \hat{A}^-)\hat{Q} = \hat{E}^+ + \hat{E}^-, \end{aligned}$$

where  $\Lambda$  is the diagonal matrix of eigenvalues of  $\hat{A}$ , and  $\Lambda^+$  and  $\Lambda^-$  are the positive and negative parts of  $\Lambda$ , respectively.  $\hat{F}$  can be similarly decomposed. Defining  $\tilde{A}^+ := \partial \hat{E}^+ / \partial \hat{Q}$ , and similarly for  $\tilde{A}^-$ ,  $\tilde{B}^+$  and  $\tilde{B}^-$ , the unfactored scheme is

$$[I + h(\delta_\xi^b \tilde{A}^+ + \delta_\xi^f \tilde{A}^- + \delta_\eta^b \tilde{B}^+ + \delta_\eta^f \tilde{B}^-)] \Delta \hat{Q}^n = -h(\delta_\xi^b \hat{E}^+ + \delta_\xi^f \hat{E}^- + \delta_\eta^b \hat{F}^+ + \delta_\eta^f \hat{F}^-)^n,$$

where  $\delta^b$  and  $\delta^f$  are backward and forward spatial difference operators. On the left-hand side, the forward difference operators are separated from the backward, resulting in an approximate factorization into lower and upper block-triangular matrices. First-order spatial differences  $\Delta^b$  and  $\Delta^f$  are used on the left-hand side, and second-order differences  $\delta^b$  and  $\delta^f$  are used on the right-hand side, to give second-order accuracy in the steady state. We have, for instance,

$$(20) \quad \Delta_\xi^b \hat{A}_i := \hat{A}_i - \hat{A}_{i-1}, \quad \delta_\xi^b \hat{E}_i := \frac{3\hat{E}_i - 4\hat{E}_{i-1} + \hat{E}_{i-2}}{2}.$$

The full scheme is thus

$$(21) \quad \begin{aligned} [I + h(\Delta_\xi^b \tilde{A}^+ + \Delta_\eta^b \tilde{B}^+)] [I + h(\Delta_\xi^f \tilde{A}^- + \Delta_\eta^f \tilde{B}^-)] \Delta \hat{Q}^n \\ = -h(\delta_\xi^b \hat{E}^+ + \delta_\xi^f \hat{E}^- + \delta_\eta^b \hat{F}^+ + \delta_\eta^f \hat{F}^-)^n. \end{aligned}$$

In the notation discussed in § 1,

$$(22) \quad \begin{aligned} M &= [I + h(\Delta_\xi^b \tilde{A}^+ + \Delta_\eta^b \tilde{B}^+)] [I + h(\Delta_\xi^f \tilde{A}^- + \Delta_\eta^f \tilde{B}^-)], \\ A &= h(\delta_\xi^b \tilde{A}^+ + \delta_\xi^f \tilde{A}^- + \delta_\eta^b \tilde{B}^+ + \delta_\eta^f \tilde{B}^-)^n. \end{aligned}$$

It is clear that  $A$  is not a constant matrix, but if we wait until the later stages of the iteration,  $A$  will be changing very slowly, and we should be able to apply annihilation with a fair degree of success.

**5. Application to steady transonic flow.** The annihilation idea has been applied to the numerical method given in § 4 for the Euler equations. In this section we will describe the physical problem, show how to apply annihilation, and give some results showing the success of annihilation. We will also mention some questions and problem areas that our experiences have identified.

We performed calculations for flow over a circular cylinder with symmetry imposed between the top and bottom, so that only the top half of the region was calculated. Fig. 2 shows the grid used, including the two points below the symmetry line in front of and behind the cylinder used to impose the symmetry condition. At the far-field boundary, 16 diameters away from the cylinder, all flow variables are fixed at their free-stream values. Boundary conditions at the body consist of setting the normal velocity to zero, taking surface density and tangential velocity from the point above, and calculating pressure from a conservation of momentum relation.

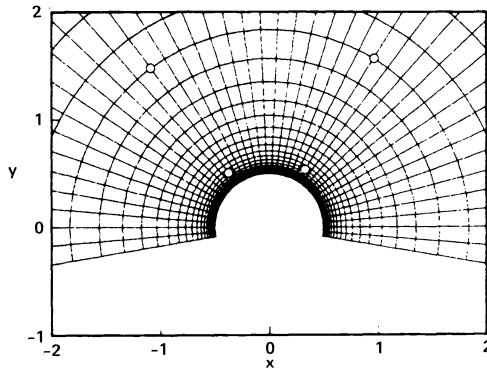


FIG. 2. Exponentially stretched  $42 \times 31$  grid about a circular cylinder, showing points used for eigenvalue estimation.

At a free-stream Mach number of 0.5, a shock forms on the cylinder. A steady solution for this case is shown in Fig. 3. The shock has introduced rotationality into the flow and caused inviscid flow separation on the back side of the cylinder.

The application of eigenvector annihilation to the iterative method (21) requires some ad hoc decisions on how to estimate the dominant eigenvalue pair. The method presented here is one of many strategies that could be envisioned. As described in § 2, one component of  $\Delta \hat{Q}$  is needed at two points in the field, corresponding to subscripts  $i$  and  $j$  in (6), for three consecutive iteration steps. We used the first component (density) of  $\Delta \hat{Q}$ . Four grid points were chosen, each a third of the way in from a corner

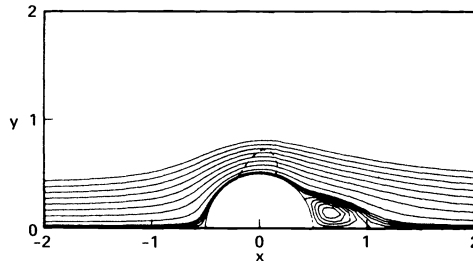


FIG. 3. Streamlines and sonic line for steady flow about a circular cylinder at a free-stream Mach number of 0.5.

of the computational domain (see Fig. 1). The two points closest to the cylinder form one pair, the other two another pair. In this way we can obtain two estimates of the dominant eigenvalue from different regions in the grid. The two points from each pair are separated in an attempt to minimize any local coupling effects. During each iteration, eigenvalue estimates are made from the two pairs of points. If the real and imaginary components of the estimates differ by less than 5%, based on the modulus of the first estimate, the two estimates are averaged to produce a candidate  $\lambda$ . Annihilation is performed if this candidate  $\lambda$  is within 5% of the candidate  $\lambda$  from the previous iteration. Thus, our criterion for linear behavior involves a consistent eigenvalue estimate from widely separated points in the grid over four iterations.

The estimated eigenvalue is in general complex; hence, two Richardson steps are required. An outline of the steps performed is given below, using the definitions of the matrices  $M$  and  $A$  from (22).

1.  $\Delta\hat{Q}^n = M(\hat{Q}^n)^{-1}A(\hat{Q}^n)\hat{Q}^n$ .
2. Decide whether to annihilate. If yes,
3.  $\hat{Q}^{n+1} = \hat{Q}^n + [|\sigma^2|/(2 \operatorname{Re} \sigma)]\Delta\hat{Q}^n$ .
4. Apply boundary conditions to  $\hat{Q}^{n+1}$ .
5.  $\Delta\hat{Q}^{n+1} = M(\hat{Q}^n)^{-1}A(\hat{Q}^n)\hat{Q}^{n+1}$ .
6.  $\hat{Q}^{n+2} = \hat{Q}^n + 2 \operatorname{Re} \sigma \Delta\hat{Q}^{n+1}$ .
7. Apply boundary conditions to  $\hat{Q}^{n+2}$ .

The result of applying annihilation, using this strategy, to the cylinder problem is shown in Fig. 4. In this figure we compare the result of no annihilation with the result of starting the annihilation strategy at  $n = 500$ . In Fig. 4 annihilation steps were performed at  $n = 504, 540, 577, 616, 670, 704, 869, 892$  and  $904$ . The convergence rate over the last 300 iterations for the curve without annihilation was about 0.9958, whereas for the curve with annihilation the convergence rate was about 0.9881. We remark that

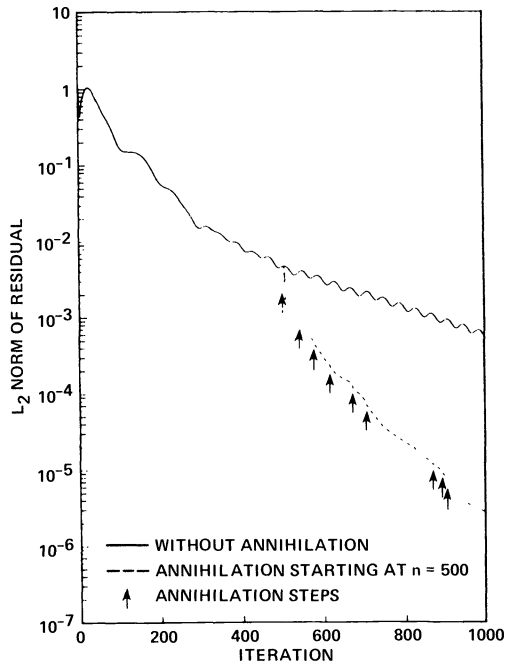


FIG. 4. Comparison of convergence histories without annihilation and with annihilation starting at  $n = 500$ .



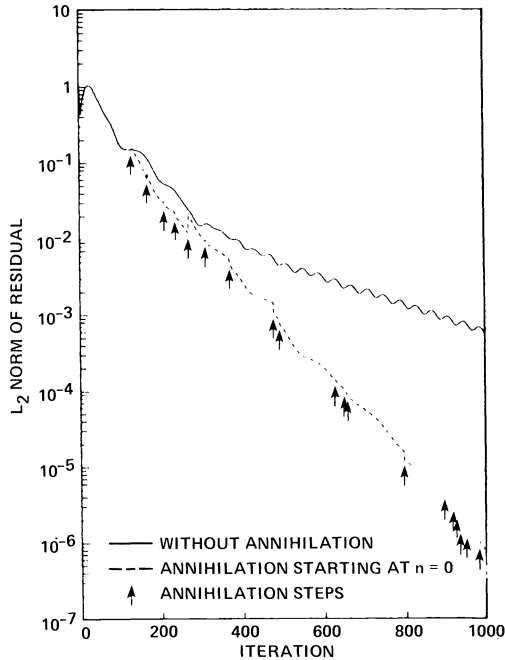


FIG. 5. Convergence history with annihilation starting at  $n = 0$ .

if only one annihilation step is performed, the residual drops sharply but eventually resumes converging at a rate of 0.9958. This may be due either to error in the estimation of the eigenvalue (so that the dominant eigenvector component of the error was not completely annihilated) or to nonlinear feedback effects stemming from the fact that the iteration process is truly nonlinear.

We close by mentioning some questions and problem areas that have arisen during this work. First, the choice of the strategy used to estimate the eigenvalues—ours was heuristic—is important. Can better ones be devised? In this regard we would like to mention the note by Jones [12], who gives a statistical criterion based on a serial correlation coefficient for deciding when to employ the Aitken technique. The application is to a real scalar sequence; it would be useful to have a similar statistical criterion for a trigonometric sequence modulated by a geometrically decaying term, but we are unaware of a serial correlation coefficient in this case.

Second, it is difficult to decide when annihilation should be started. In Fig. 5 we show the result of starting the annihilation strategy at  $n = 0$ . In this case there were 19 annihilation steps: at  $n = 129, 168, 208, 235, 268, 310, 369, 476, 491, 626, 648, 656, 794, 896, 918, 927, 934, 951$  and  $984$ . Evidently this strategy was successful, but not much more so than that of waiting until  $n = 500$  to start annihilating. This may be due to an incorrect estimation of eigenvalues or to the nonlinearity of the process.

Third, stability questions arise when performing annihilation. We see in Fig. 5 some sharp increases in the residual at certain annihilation steps, and annihilation strategies that do not allow these jumps might be preferred. Finally, the annihilation procedure requires an extra (third) level of computer storage beyond that normally needed for the iterative procedure; this may be a problem in cases where extra space is scarce.

**Acknowledgment.** We extend our thanks to Harvard Lomax for introducing us to the idea of eigenvector annihilation.

## REFERENCES

- [1] J. B. BELL, G. R. SHUBIN AND J. M. SOLOMON, *Fully implicit shock tracking*, J. Comp. Phys., 48 (1982), pp. 223-245.
- [2] GARY M. JOHNSON, *Multiple-grid acceleration of Lax-Wendroff algorithms*, NASA TM-82843, 1982.
- [3] A. JAMESON, W. SCHMIDT AND E. TURKEL, *Numerical solutions of the Euler equations by finite volume methods using Runge-Kutta time stepping schemes*, AIAA paper 81-1259, June 1981.
- [4] J. H. WILKINSON, *The Algebraic Eigenvalue Problem*, Cambridge Univ. Press, London, 1966, pp. 578 ff.
- [5] L. A. LYUSTERNIK, *Remarks on the numerical solution of boundary problems for Laplace's equation and the calculation of characteristic values by the method of networks*, Trav. Inst. Math. Stekloff, 20 (1947), pp. 49-64. (In Russian.)
- [6] J. M. HYMAN AND T. A. MANTEUFFEL, *Dynamic acceleration of nonlinear iterations*, preprint, 1983.
- [7] T. A. MANTEUFFEL, *Adaptive procedure for estimating parameters for the nonsymmetric Tchebychev iteration*, Numer. Math., 31 (1978), pp. 183-208.
- [8] H. VIVIAND, *Conservative forms of gas dynamics equations*, La Recherche Aeronautique, 1 (Jan.-Feb. 1974), p. 65.
- [9] R. F. WARMING AND R. M. BEAM, *On the construction and application of implicit factored schemes for conservation laws*, Symposium on Computational Fluid Dynamics, SIAM-AMS Proceedings, 11, 1978.
- [10] J. L. STEGER AND R. F. WARMING, *Flux vector splitting of the inviscid gasdynamic equations with application to finite-difference methods*, J. Comp. Phys., 40 (1981), pp. 263-293.
- [11] P. G. BUNING AND J. L. STEGER, *Solution of the two-dimensional Euler equations with generalized coordinate transformation using flux vector splitting*, AIAA paper 82-0971, June 1982.
- [12] B. JONES, *A note on Aitken's  $\delta^2$  technique*, SIGNUM Newsletter, June 1982, p. 23.

## NUMERICAL AND ASYMPTOTIC STUDY OF A SYSTEM OF COUPLED DIFFUSION AND REACTION EQUATIONS ARISING IN ENZYME KINETICS\*

FRANCIS CONRAD† AND CLAUDINE SCHMIDT-LAINÉ‡

**Abstract.** We consider in this paper a system of coupled equations modelling the steady states in temperature and concentration of a substrate in an enzyme membrane. Two numerical methods are presented for the approximation of the solutions: an optimal control method, and a continuation technique which allows us to follow the solution paths when some parameter is modified. An asymptotic study completes the numerical results.

**Key words.** reaction-diffusion system, enzyme kinetics, optimal control, continuation, asymptotic analysis.

**1. Introduction. The basic system.** A plane enzymatic membrane of thickness  $e$  is immersed in a solution where the substrate has fixed concentration  $S_0$  at a temperature  $T_0$ . The substrate diffuses through the membrane and reacts. If  $S$  and  $T$  denote the concentration and the temperature of the substrate in the membrane, respectively, mass and heat balances lead to the following steady state equations [1], [2], [6], [16]:

$$(1.1) \quad \begin{aligned} -(DS_x)_x + \frac{V_m S}{K_m + |S|} &= 0, & 0 < x < e, & \quad S(0) = S(e) = S_0, \\ -(\Lambda T_x)_x - \Delta H \frac{V_m S}{K_m + |S|} &= 0, & 0 < x < e, & \quad T(0) = T(e) = T_0. \end{aligned}$$

The parameters  $D$ ,  $\Lambda$ ,  $V_m$ ,  $K_m$ ,  $\Delta H$  are, respectively, the species and heat diffusion coefficients, maximal rate of the reaction, kinetic constant and reaction enthalpy;  $\Delta H$  is positive for an exothermic reaction, negative otherwise.

Usually, these parameters are considered as constant. Thus (1.1) is in fact an uncoupled model which has been extensively studied from a theoretical point of view as well as from a numerical one [11], [12].

However, in some cases, especially for highly exothermic reactions, the dependence of some of these parameters on  $T$  cannot be neglected [1]. The functional dependencies in  $T$  are the following:

$$\begin{aligned} D(T) &= D_0 \exp\left(-\frac{E}{RT}\right) && \text{(Arrhénius law),} \\ V_m(T) &= V_0 \exp\left(-\frac{E'}{RT}\right) && \text{(Arrhénius law),} \\ K_m(T) &= K_0 \exp\left(-\frac{\Delta H}{RT}\right) && \text{(Van't Hoff law),} \\ \Lambda(T) &= \Lambda_0(1 + bT) && \text{(experimental curve [18]).} \end{aligned}$$

\* Received by the editors December 22, 1983, and in revised form April 17, 1984.

† Département Informatique, Ecole des Mines de Saint Etienne, 158 Cours Fauriel, 42023 Saint-Etienne, France.

‡ Département Mathématiques - Informatique - Systèmes, Ecole Centrale de Lyon, 36 Route de Dardilly, 69130 Ecully, France.

All the constants introduced are positive,  $E, E'$  are activation energies,  $R$  is the gas constant.

With these laws, (1.1) becomes a genuinely coupled system, whose study is the goal of the present paper.

In § 2 we briefly discuss an existence result for (1.1) and examine the behavior of the system when some of the parameters become large. In §§ 3 and 4 we describe numerical methods used to solve (1.1), mainly when  $\Delta H$  is considered a bifurcation parameter: in § 3 we use an optimal control technique, in § 4 a continuation method. The two methods are tested on a physical model which has been studied in [1], and numerical results are given.

**2. Theoretical study of system (1.1).**

**2.1. Existence results.** Existence proofs for systems more general than (1.1) have been established in [5]. In the special case of (1.1) we can proceed as follows:

By combining the two equations in (1.1) we obtain:

$$(2.1) \quad \Delta HD(T)S_x + \Lambda(T)T_x = \text{const.}, \quad \text{i.e.,} \quad S_x + \frac{1}{\Delta H} \frac{\Lambda(T)}{D(T)} T_x = \frac{\text{const.}}{\Delta HD(T)}.$$

We set  $\alpha(z) = \int_{T_0}^z \Lambda(t)/D(t) dt$  and integrate (2.1)

$$S(x) - S_0 + \frac{\alpha(T(x))}{\Delta H} = \frac{\text{const.}}{\Delta H} \int_0^x \frac{dy}{D(T(y))}.$$

For  $x = e$ ,  $\text{const.} = 0$ ; therefore, setting  $S(x) = S_0 - \alpha(T(x))/\Delta H$  leads to

$$(2.2) \quad (\Lambda(T)T_x)_x + \Delta H \frac{V_m(T)[S_0 - \alpha(T)/\Delta H]}{K_m(T) + |S_0 - \alpha(T)/\Delta H|} = 0.$$

We define also

$$\beta(z) = \int_{T_0}^z \Lambda(t) dt; \quad \text{then (2.2) is equivalent to}$$

$$(2.3) \quad (\beta(T))_{xx} + \Delta H \frac{V_m(T)[S_0 - \alpha(T)/\Delta H]}{K_m(T) + |S_0 - \alpha(T)/\Delta H|} = 0$$

or, with  $u(x) = \beta(T(x))$  and

$$f(z) = \Delta H \frac{V_m(\beta^{-1}(z))[S_0 - \alpha[\beta^{-1}(z)]]/\Delta H}{K_m(\beta^{-1}(z)) + |S_0 - \alpha[\beta^{-1}(z)]]/\Delta H},$$

the final equivalent formulation of (1.1) under the above transformations is:

$$(2.4) \quad u_{xx} + f(u) = 0, \quad 0 < x < e, \quad u(0) = u(e) = 0.$$

**THEOREM 2.1.** *System (1.1) admits at least one solution  $(S, T)$  and in fact a minimal-maximal solution  $(\underline{S}, \bar{T})$  and a maximal-minimal solution  $(\bar{S}, \underline{T})$ .*

*Proof.* We consider first (2.4); let  $z_1 > 0$  be the unique root of  $\Delta HS_0 = \alpha[\beta^{-1}(z_1)]$ . Since  $f(z)$  is positive for  $z < z_1$ , negative for  $z > z_1$ , a standard calculation shows that any solution  $u$  of (2.4) satisfies  $u \leq z_1$  a.e., and, since  $f(u)$  is then positive, also  $u \geq 0$  by the maximum principle. Moreover, since  $f(0) \geq 0, f(z_1) = 0, 0$  is a subsolution and  $z_1$  is a supersolution of (2.4). Therefore the existence of at least one solution of (2.4) is standard [17]. We have in fact the existence of a minimal solution  $\underline{u}$  and a maximal solution  $\bar{u}$ , and every solution  $u$  of (2.4) satisfies:

$$0 \leq \underline{u} \leq u \leq \bar{u} \leq z_1.$$

Using the transformations  $T = \beta^{-1}(u)$ ,  $S = S_0 - \alpha(T)/\Delta H$  and setting  $\bar{T} = \beta^{-1}(\bar{u})$  (resp.  $\underline{T} = \beta^{-1}(\underline{u})$ ),  $\bar{S} = S_0 - \alpha(\bar{T})/\Delta H$  (resp.  $\underline{S} = S_0 - \alpha(\underline{T})/\Delta H$ ) gives the result.  $\square$

*Remark:* Any solution  $(S, T)$  of (1.1) satisfies  $0 \leq S \leq S_0$ ;  $T \geq T_0$ . These inequalities can be obtained by a direct calculation on the weak formulation equivalent to (1.1) or else from (2.4):  $u \geq 0 \Rightarrow T = \beta^{-1}(u) \geq \beta^{-1}(0) = T_0$  and  $u \leq z_1 \Rightarrow S = S_0 - \alpha(T)/\Delta H \geq 0$ .

Henceforth, we consider  $\Delta H$  as a bifurcation parameter, and look for solution branches  $S(\Delta H)$ ,  $T(\Delta H)$ , when all the remaining parameters are kept fixed. First, let us observe that  $f(z) = g(\Delta H, \beta^{-1}(z))$  where

$$g(\Delta H, T) = \Delta H \frac{V_m(T)[S_0 - \alpha(T)/\Delta H]}{K_0 \exp(-\Delta H/RT) + |S_0 - \alpha(T)/\Delta H|}$$

so (2.4) is equivalent to the operating form:

$$(2.5) \quad u_{xx} + g(\Delta H, \beta^{-1}(u)) = 0, \quad u(0) = u(e) = 0.$$

**THEOREM 2.2.** *Let  $\underline{T}(\Delta H)$  (resp.  $\bar{T}(\Delta H)$ ) be the minimal (resp. maximal) solution of (1.1). Then  $\underline{T}$  (resp.  $\bar{T}$ ) considered as a mapping from  $[0, \infty)$  to  $\mathcal{C}[0, e]$  is increasing and left (resp. right) continuous. For small  $\Delta H$ , (1.1) admits a unique solution  $(S, T)$ .*

*Proof.* We consider the equivalent formulation (2.5); let  $\Delta H_1 < \Delta H_2$  and  $\underline{u} = \underline{u}(\Delta H_2)$  be the minimal solution of (2.5) for  $\Delta H = \Delta H_2$ . Since  $g$  is increasing with respect to  $\Delta H$ ,

$$\underline{u}_{xx} + g(\Delta H_1, \underline{u}) \leq \underline{u}_{xx} + g(\Delta H_2, \underline{u}) = 0,$$

thus  $\underline{u}(\Delta H_2)$  is a supersolution of (2.5) for  $\Delta H = \Delta H_1$ , so  $\underline{u}(\Delta H_1) \leq \underline{u}(\Delta H_2)$ . In the same way,  $\bar{u}(\Delta H_1)$  is a subsolution of (2.5) for  $\Delta H = \Delta H_2$  and thus  $\bar{u}(\Delta H_1) \leq \bar{u}(\Delta H_2)$ . The one-sided continuity of the extremal solutions is then a classical fact, see [9]. In order to prove uniqueness, we observe that

$$|g(\Delta H, \beta^{-1}(u))| \leq \Delta H V_m(T) \leq \text{const. } \Delta H.$$

Consequently,  $\lim u(\Delta H) = 0$  in  $\mathcal{C}[0, e]$  as  $\Delta H \rightarrow 0$ , for any solution. Since

$$\frac{\partial f}{\partial z} = \frac{\partial g}{\partial T}(\Delta H, \beta^{-1}(z)) \cdot \frac{\partial \beta^{-1}(z)}{\partial z},$$

$$\lim_{\substack{\Delta H \rightarrow 0 \\ z \rightarrow 0}} \frac{\partial f}{\partial z} = 0 \quad \text{and} \quad \frac{\partial f}{\partial z}(\zeta) \leq \lambda_1$$

for small  $\Delta H$  and any function  $\zeta$  such that  $\underline{u} \leq \zeta \leq \bar{u}$ , where  $\lambda_1 = \pi^2/e^2$  is the first eigenvalue of the problem:

$$-w_{xx} = \lambda w, \quad w(0) = w(e) = 0.$$

This implies uniqueness and completes the proof.  $\square$

*Remark.* 1) No monotonicity in  $\Delta H$  can be proved for  $\bar{S}$  or  $\underline{S}$  and in fact, the numerical results in §§ 3 and 4 show clearly that  $S$  is indeed not monotonic in  $\Delta H$ .

2) For the general case of nonconstant  $V_m$ , i.e.,  $V_m = V_0 e^{-E'/RT}$ , with sufficiently large  $E'$  one can expect nonuniqueness, and so an  $S$ -shape response curve.

**2.2. Asymptotic analysis for large  $S_0$  or  $\Delta H$ .** Numerical tests (see also §§ 3 and 4) have suggested that there is some limit problem for suitable nondimensional functions of  $S$  and  $T$  when  $S_0$  or  $\Delta H$  become large. These facts will be established analytically here for the problem tested numerically, which corresponds to  $V_m(T) = \text{const.}$  ( $E' = 0$ ).

We recall this special form of problem (1.1):

$$\begin{aligned}
 (1.1) \quad & -(D(T)S_x)_x + \frac{V_0 S}{K_m(T) + |S|} = 0, \quad S(0) = S(e) = S_0, \\
 & -(\Lambda(T)T_x)_x - \Delta H \frac{V_0 S}{K_m(T) + |S|} = 0, \quad T(0) = T(e) = T_0, \\
 & D(T) = D_0 \exp(-E/RT), \quad K_m(T) = K_0 \exp(-\Delta H/RT), \\
 & \Lambda(T) = \Lambda_0(1 + bT), \quad D_0, E, \Delta H, \Lambda_0, b, R > 0.
 \end{aligned}$$

First, we observe that in this case, (1.1) admits a unique solution. Indeed with the notation of § 2.1,  $(S, T)$  is a solution of (1.1) iff  $T$  satisfies:

$$(2.3) \quad (\beta(T))_{xx} + g(T) = 0, \quad 0 < x < e, \quad T(0) = T(e) = T_0,$$

but

$$g(T) = \Delta H V_0 \frac{S_0 - \alpha(T)/\Delta H}{K_m(T) + |S_0 - \alpha(T)/\Delta H|}$$

is decreasing in  $T$  since  $\alpha$  and  $K_m$  are increasing in  $T$ . Since  $\beta$  is also increasing in  $T$ , uniqueness in  $T$ , and hence in  $(S, T)$  is a consequence of the monotonicity of the operator  $T \rightarrow -[\beta(T)]_{xx} - g(T)$ .

**2.2.1. The limit system for large  $S_0$ .** We set  $x = ey$ ,  $S(x) = S_0 u(y)$ ,  $T(y) = v(y)/b$ ; with these new variables (1.1) takes the nondimensional form:

$$\begin{aligned}
 (2.6) \quad & -\left(\frac{D(T)K_0}{V_0 e^2} u_y\right)_y + \varepsilon \frac{u}{\varepsilon e^{-\Delta H/RT} + |u|} = 0, \\
 & ((1 + v)v_y)_y + \frac{\Delta H V_0 e^2 b}{\Lambda_0} \frac{u}{\varepsilon e^{-\Delta H/RT} + |u|} = 0, \\
 & u(0) = u(1) = 1, \quad v(0) = v(1) = bT_0 = \alpha,
 \end{aligned}$$

where  $\varepsilon = K_0/S_0 \searrow 0_+$ . Let  $(u, v)$  be the solution of (2.6) for  $\varepsilon > 0$ .

**THEOREM 2.3.** *As  $\varepsilon \searrow 0_+$ ,  $\lim u = 1$ ,  $\lim v = v_\infty$  in  $\mathcal{C}[0, 1]$  where  $v_\infty(y)$  is the unique positive root of the equation*

$$v_\infty(y) + \frac{v_\infty^2(y)}{2} - \left(\alpha + \frac{\alpha^2}{2}\right) = \frac{\Delta H V_0 e^2 b}{2\Lambda_0} y(1 - y).$$

*Proof.* We multiply the first equation by  $1 - u$  and integrate by parts

$$\int_0^1 \frac{D(T)K_0}{V_0 e^2} (1 - u)_y^2 dy = \varepsilon \int_0^1 \frac{u(1 - u)}{\varepsilon e^{-\Delta H/RT} + u} dy.$$

Since  $T \geq T_0$  we get

$$\int_0^1 (1 - u)_y^2 dy \leq \frac{V_0 e^2}{D(T_0)K_0} \varepsilon \int_0^1 (1 - u) dy,$$

which implies  $1 - u \rightarrow 0$  in  $H^1_0(0, 1)$ , hence in  $\mathcal{C}[0, 1]$ .

Similarly, we multiply the second equation by  $v - \alpha$  and integrate by parts:

$$\int_0^1 (1 + v)v_y^2 dy = \frac{\Delta H V_0 e^2 b}{\Lambda_0} \int_0^1 \frac{u}{\varepsilon e^{-\Delta H/RT} + |u|} (v - \alpha) dy.$$

Since  $T \geq T_0$ ,  $v \geq \alpha \geq 0$  and we get

$$\int_0^1 v_y^2 dy \leq \frac{\Delta H V_0 e^2 b}{\Lambda_0} \int_0^1 (v - \alpha) dy$$

which implies that  $v - \alpha$  remains bounded in  $H_0^1(0, 1)$  as  $\varepsilon \searrow 0$ .

Therefore,  $v - \alpha \rightarrow v_\infty - \alpha$  weakly in  $H_0^1(0, 1)$ , hence in  $\mathcal{C}[0, 1]$ . In order to obtain the limit problem concerning  $v$ , we now multiply the second equation by  $\phi \in \mathcal{D}(0, 1)$  and obtain the weak formulation:

$$\int_0^1 (1 + v)v_y \phi_y dy = \frac{\Delta H V_0 e^2 b}{\Lambda_0} \int_0^1 \frac{u}{\varepsilon e^{-\Delta H/RT} + |u|} \phi dy.$$

Since  $u \rightarrow 1$  uniformly, the limit as  $\varepsilon \searrow 0$  of the right-hand side is  $(\Delta H V_0 e^2 b / \Lambda_0) \int_0^1 \phi dy$ . For the left-hand side, we have

$$\begin{aligned} \int_0^1 (1 + v)v_y \phi_y dy &= \int_0^1 (1 + v - 1 - v_\infty)v_y \phi_y dy + \int_0^1 (1 + v_\infty)v_y \phi_y dy \\ \lim_{\varepsilon \rightarrow 0} \int_0^1 (v - v_\infty)v_y \phi_y dy &= 0 \end{aligned}$$

since  $v - v_\infty \rightarrow 0$  uniformly and  $v_y$  is bounded in  $L^2(0, 1)$ .

$$\lim_{\varepsilon \rightarrow 0} \int_0^1 (1 + v_\infty)v_y \phi_y dy = \int_0^1 (1 + v_\infty)v_{\infty y} \phi_y dy$$

since  $v - \alpha \rightarrow v_\infty - \alpha$  weakly in  $H_0^1(0, 1)$ . Thus the limit equation for  $v_\infty$  is:

$$((1 + v_\infty)v_{\infty y})_y + \frac{\Delta H V_0 e^2 b}{\Lambda_0} = 0, \quad v_\infty(0) = v_\infty(1) = \alpha,$$

which gives the desired result.  $\square$

*Remark.* The above behavior appears numerically in Figs. 6 and 7.

**2.2.2. The limit system for large  $\Delta H$ .** We set  $x = \varepsilon y$ ,  $S(x) = S_0 u(y)$  as in § 2.2.1, but  $T(x) = T_0(1 + w(y))$ ; with these new variables (1.1) is transformed into the non-dimensional form:

$$\begin{aligned} & - \left( \exp \left( - \frac{E}{RT_0(1+w)} \right) u_y \right)_y + \frac{V_0 e^2}{D_0 S_0} \frac{u}{\frac{K_0}{S_0} \exp \left( - \frac{\Delta H}{RT_0(1+w)} \right) + |u|} = 0, \\ (2.7) \quad & - ((1 + bT_0(1 + w))w_y)_y - \frac{\Delta H V_0 e^2}{\Lambda_0 T_0} \frac{u}{\frac{K_0}{S_0} \exp \left( - \frac{\Delta H}{RT_0(1+w)} \right) + |u|} = 0, \\ & u(0) = u(1) = 1, \quad w(0) = w(1) = 0. \end{aligned}$$

Finally, we set  $\sigma = V_0 e^2 / D_0 S_0$ ,  $\tau = V_0 e^2 R / \Lambda_0$ ,  $\alpha = E / RT_0$ ,  $\beta = K_0 / S_0$ ,  $\gamma = bT_0$ , all these constants being nondimensional and positive, and  $\lambda = \Delta H / RT_0 \nearrow \infty$ ; then (2.7) is equivalent to

$$\begin{aligned} & - (e^{-\alpha/(1+w)} u_y)_y + \sigma \frac{u}{\beta e^{-\lambda/(1+w)} + |u|} = 0, \\ (2.8) \quad & - [(1 + \gamma(1 + w))w_y]_y - \tau \lambda \frac{u}{\beta e^{-\lambda/(1+w)} + |u|} = 0, \\ & u(0) = u(1) = 1, \quad w(0) = w(1) = 0. \end{aligned}$$

Let us recall that  $0 \leq S \leq S_0$  and  $T \geq T_0$ ; therefore  $0 \leq u \leq 1$  and  $w \geq 0$ .

**THEOREM 2.4.** *Let us assume  $\sigma e^\alpha < 8$ . Then  $\lim_{\lambda \nearrow \infty} u = u^\infty$ ,  $\lim_{\lambda \nearrow \infty} (w/\sqrt{\lambda}) = v^\infty$  in  $\mathcal{C}[0, 1]$  where*

$$u^\infty(y) = 1 - \frac{\sigma}{2}y(1-y) \quad \text{and} \quad v^{\infty 2}(y) = \frac{\tau}{\gamma}y(1-y).$$

*Proof.* We multiply the first equation by  $1-u$  and integrate by parts in order to bound  $1-u$  in  $H^1_0(0, 1)$  as  $\lambda \nearrow \infty$ ; hence,  $u \rightarrow u^\infty$  in  $H^1(0, 1)$  weakly and in  $\mathcal{C}[0, 1]$ .

Next we prove that  $u$  (and consequently  $u^\infty$ )  $\geq 1 - \sigma e^\alpha/8 > 0$ . We proceed as in § 2 for the existence result; by combination of the two equations, we get

$$\begin{aligned} \tau\lambda e^{-\alpha/(1+w)}u_y + \sigma[1 + \gamma(1+w)]w_y &= C = 0 \quad (\text{as in § 1}), \\ \tau\lambda[u(y) - u(0)] + \sigma \int_0^y [1 + \gamma(1+w)]w_y e^{\alpha/(1+w)} dy &= 0, \\ \tau\lambda[1 - u(y)] &\leq \sigma e^\alpha \int_0^y [1 + \gamma(1+w)]w_y dy = \sigma e^\alpha \left[ w + \gamma w + \frac{\gamma w^2}{2} \right]_0^y. \end{aligned}$$

From (2.8) we get

$$\begin{aligned} -\left( w + \gamma w + \frac{\gamma w^2}{2} \right)_{yy} &\leq \tau\lambda \Rightarrow w + \gamma w + \frac{\gamma w^2}{2} \leq \frac{\tau\lambda}{2}y(1-y) \leq \frac{\tau\lambda}{8} \\ &\Rightarrow \tau\lambda[1 - u(y)] \leq \sigma e^\alpha \frac{\tau\lambda}{8} \\ &\Rightarrow u(y) \geq 1 - \frac{\sigma e^\alpha}{8}. \end{aligned}$$

Now let us find a limit for  $v = w/\sqrt{\lambda}$ . If we set  $\psi(y) = w(y) + \gamma w(y) + \gamma w^2(y)/2$ , the second equation becomes:

$$-\psi_{yy} = \tau\lambda \frac{u}{\beta e^{-\lambda/(1+w)} + |u|},$$

therefore  $\psi/\lambda$  is bounded in  $W^{2,p}(0, 1)$  for any  $p > 2$ , and has a limit  $z^\infty$  in  $\mathcal{C}^{1,\mu}[0, 1]$  for any  $\mu \in (0, 1)$ . Consequently

$$\frac{w}{\sqrt{\lambda}} = \frac{-(1+\gamma)/\sqrt{\lambda} + \sqrt{(1+\gamma)^2/\lambda + 2\gamma\psi/\lambda}}{\gamma}$$

has a limit  $v^\infty = \sqrt{2z^\infty/\gamma}$  at least in  $\mathcal{C}[0, 1]$  as  $\lambda \nearrow \infty$ .

The next step consists in giving a lower bound for  $w$ . We have

$$-\psi_{yy} = \tau\lambda \frac{u}{\beta e^{-\lambda/(1+w)} + |u|} \geq \tau\lambda \frac{u}{\beta + |u|} \geq 2\lambda\delta,$$

where

$$2\delta = \frac{\tau(1 - \sigma e^\alpha/8)}{\beta + 1 - \sigma e^\alpha/8} > 0,$$

hence,

$$\psi \geq \lambda\delta y(1-y) \quad \text{and} \quad w(x) \geq \frac{-(1+\gamma) + \sqrt{(1+\gamma)^2 + 2\lambda\delta\gamma y(1-y)}}{\gamma}.$$



This inequality implies  $w \rightarrow \infty$  a.e., since

$$\frac{w}{\sqrt{\lambda}} \geq \frac{(1 + \gamma)/\sqrt{\lambda} + \sqrt{(1 + \gamma)^2/\lambda + 2\delta\gamma y(1 - y)}}{\gamma}.$$

Finally we have to pass to the limit in the equations as  $\lambda \nearrow \infty$ , in a weak sense. Let  $\phi \in \mathcal{D}(0, 1)$ . Since  $w/\sqrt{\lambda} \rightarrow v^\infty$  uniformly,  $w/\lambda \rightarrow 0$  uniformly and

$$\lim_{\lambda \nearrow \infty} \int_0^1 \frac{u}{\beta e^{-\lambda/(1+w)} + |u|} \phi \, dy = \int_0^1 \frac{u^\infty}{|u^\infty|} \phi \, dy = \int_0^1 \phi \, dy$$

because  $u^\infty$  is positive. Concerning the limit of the second equation of (2.8) we have

$$\begin{aligned} \frac{1}{\lambda} \int_0^1 ([1 + \gamma(1 + w)]w_y)_y \phi \, dy &= -\frac{1}{\lambda} \int_0^1 [1 + \gamma(1 + w)]w_y \phi_y \, dy \\ &= \int_0^1 \left( \frac{w}{\lambda} + \gamma \frac{w}{\lambda} + \gamma \frac{w^2}{2\lambda} \right) \phi_{yy} \, dy \rightarrow \int_0^1 \frac{\gamma}{2} v^{\infty 2} \phi_{yy} \, dy. \end{aligned}$$

Therefore, the limit equation in  $\mathcal{D}'(0, 1)$  is the following:

$$\frac{\gamma}{2} (v^{\infty 2})_{yy} + \tau = 0, \quad v^\infty(0) = v^\infty(1) = 0,$$

hence,  $v^{\infty 2}(y) = (\tau/\gamma)y(1 - y)$ .

In the same manner, we write:

$$\begin{aligned} \int_0^1 e^{-\alpha/(1+w)} u_y \phi_y \, dy &= \int_0^1 (e^{-\alpha/(1+w)} - 1) u_y \phi_y \, dy + \int_0^1 u_y \phi_y \, dy, \\ \lim_{\lambda \nearrow \infty} \int_0^1 u_y \phi_y \, dy &= \int_0^1 u_y^\infty \phi_y \, dy \end{aligned}$$

since  $1 - u \rightarrow 1 - u^\infty$  weakly in  $H_0^1(0, 1)$ .

The first integral term goes to 0, as  $\lim e^{-\alpha/(1+w)} - 1 = 0$  a.e. since  $w \rightarrow \infty$  a.e. But  $(e^{-\alpha/(1+w)} u_y)_y$  is bounded in  $L^2(0, 1)$  which implies  $e^{-\alpha/(1+w)} u_y$  is bounded in  $H^1(0, 1)$ , hence in  $\mathcal{C}[0, 1]$ , and  $u_y$  is also bounded in  $\mathcal{C}[0, 1]$ . Then

$$\lim_{\lambda \nearrow \infty} \int_0^1 (e^{-\alpha/(1+w)} - 1) u_y \phi_y \, dy = 0$$

by the Lebesgue dominated convergence theorem, and the limit equation for  $u$  is

$$-(u^\infty)_{yy} + \sigma = 0, \quad u^\infty(0) = u^\infty(1) = 1,$$

i.e.,  $u^\infty(y) = 1 - (\sigma/2)y(1 - y)$ .  $\square$

*Remark.* The bound 8 for  $\sigma e^\alpha$  is not optimal; see the numerical results shown in Fig. 8 for  $S_0 = 5$ , which gives  $\sigma e^\alpha \cong 32$ . However, Theorem 2.4 is certainly not true for any value of  $\sigma e^\alpha$  since if  $\sigma > 8$ , the limit  $u^\infty$  is no longer positive! When  $\sigma$  is considered variable (with  $S_0$ ) we conjecture the existence of a critical bound  $\sigma_c \in [8e^{-\alpha}, 8]$  such that for  $\sigma < \sigma_c$ , Theorem 2.4 is true and for  $\sigma > \sigma_c$  the limit problem is a free boundary problem in  $u$ , i.e.,  $u^\infty = 0$  on a subset of  $(0, 1)$  of positive measure (and  $w$  does not become infinite). Previous results [3] on similar problems have motivated this conjecture.

In the remaining part of the paper, we discuss numerical techniques and give some results concerning (1.1). From this point of view, the formulation (1.1) is more convenient than (2.4) or (2.5).

We develop two methods which have been tested on (1.1). The crucial feature of (1.1) is that the system is strongly nonlinear and strongly coupled, that is, nonlinear and coupled in the second order terms.

**3. Numerical analysis: an optimal control method.** We recall the classical method of Céa and Geymonat [4] for a quasilinear second order problem of the form

$$(3.0) \quad Au + f(x, u) = 0$$

(with boundary conditions). First one has to replace (3.0) by the following control formulation, for some norm in a suitable Hilbert space

$$(3.1) \quad \begin{aligned} Au_\lambda + f(x, \lambda) = 0 & \quad (\lambda \text{ is the control, } u_\lambda \text{ is the state}), \\ \inf_{\lambda \in V} J(\lambda), & \quad \text{where } J(\lambda) = \frac{1}{2} \|u - \lambda\|_V^2. \end{aligned}$$

Then (3.1) is solved using an optimization algorithm. The gradient is obtained via an adjoint problem associated with the state equation (see [4]). This method has been used, for instance, in [13]. However, the classical framework of this method is not applicable to our system, in which the nonlinearity appears also in the second order terms. Therefore, no natural “additive” splitting is available and we have to adapt the method of Céa and Geymonat. First, we are going to present the extension of the method in a general framework, then we apply it to our system, and give numerical results.

**3.1. Principle of the method.** The notations used are the following: let  $V$  be a Hilbert space with inner product  $((\cdot, \cdot))$  and associated norm  $\|\cdot\|_V$ ,  $V'$  its dual with the norm  $\|\cdot\|_{V'}$ ; the duality between  $V$  and  $V'$  is denoted by the brackets  $\langle \cdot, \cdot \rangle$ .

We consider the problem

$$(3.2) \quad Au = f \quad \text{where } f \in V'.$$

$A$  is a nonlinear operator from  $V$  to  $V'$  of the form:  $A(u) = C(u, u) - B(u)$  where  $C: V \times V \rightarrow V'$  and  $B: V \rightarrow V'$  are continuous,  $C(\cdot, \lambda)$  is linear and continuous from  $V$  to  $V'$ .

We now explain the method formally, assuming enough regularity and coerciveness so that the following equations or expressions make sense. For a rigorous statement of the method, see [5].

For a fixed  $\lambda \in V$ , let  $u_\lambda$  be the (unique) solution of the *state equation*:

$$(3.3) \quad C(u, \lambda) = f + B(\lambda),$$

then we replace (3.2) by the following *optimal control problem*

$$(3.4) \quad \inf_{\lambda \in V} J(\lambda) \quad \text{where } J(\lambda) = \frac{1}{2} \|u_\lambda - \lambda\|_V^2.$$

Under some assumptions, it is possible to give an equivalent optimality system characterizing the solution of the control problem (3.4) [5]. In any case, the gradient of the cost function  $J$  can be calculated by solving two successive linear equations [5]. First define the *adjoint state*  $p \in V$  by

$$(3.5) \quad \langle C(\phi, \lambda), p \rangle = ((u_\lambda - \lambda, \phi)) \quad \forall \phi \in V.$$

Consider next the solution  $K \in V$  of

$$(3.6) \quad \langle (K, \phi) \rangle = \langle [C_\lambda(u_\lambda, \lambda) - B_\lambda(\lambda)]\phi, p \rangle \quad \forall \phi \in V,$$

where  $C_\lambda$  and  $B_\lambda$  are the Fréchet derivatives of  $C(u, \cdot)$  and  $B(\cdot)$ . Then  $\nabla J(\lambda) = -K - (u - \lambda)$ .

Therefore, the numerical method used to solve (1.1) is the following:

- (1) Choose a control  $\lambda_0$  ( $k=0$ ).
- (2) For a given control  $\lambda_k$ , compute the state  $u_k$

$$C(u_k, \lambda_k) = f + B(\lambda_k).$$

- (3) Determine the adjoint state  $p_k$ , the solution of

$$\langle C(\phi, \lambda_k), p_k \rangle = \langle (u_k - \lambda_k, \phi) \rangle \quad \forall \phi \in V.$$

- (4) Solve  $\langle (K_k, \phi) \rangle = \langle [C_\lambda(u_k, \lambda_k) - B_\lambda(\lambda_k)]\phi, p_k \rangle \forall \phi \in V$ .

- (5)  $\nabla J(\lambda_k) = -K_k - (u_k - \lambda_k)$ .

- (6) If  $\nabla J(\lambda_k) = 0$  then stop; otherwise determine  $\lambda_{k+1}$  by some minimization technique using  $\nabla J(\lambda_k)$ , set  $k = k + 1$  and return to Step 2.

For a convergence result when a gradient algorithm with constant step is used, see [5].

**3.2. Adaptation to the coupled equations.** We set

$$S = S_0 + u, \quad T = T_0 + v,$$

$$\tilde{D}(v) = D(T), \quad \tilde{\Lambda}(v) = \Lambda(T),$$

$$\tilde{F}(u, v) = \frac{V_m(T)S}{K_m(T) + |S|}, \quad \tilde{G}(u, v) = -\Delta H \tilde{F}(u, v), \quad \tilde{H}(u, v) = \frac{V_m(T)}{K_m(T) + |S|}$$

and rewrite the physical system (1.1) as

$$-(\tilde{D}(v)u_x)_x + \tilde{F}(u, v) = 0,$$

$$-(\tilde{\Lambda}(v)v_x)_x + \tilde{G}(u, v) = 0,$$

$$u(0) = u(e) = 0, \quad v(0) = v(e) = 0.$$

We set  $V = H_0^1(0, e) \times H_0^1(0, e)$ ,  $V' = H^{-1}(0, e) \times H^{-1}(0, e)$  and consider the splitting

$$C(u, v; \lambda, \mu) = \begin{cases} -(\tilde{D}(\mu)u_x)_x + u\tilde{H}(\lambda, \mu), \\ -(\tilde{\Lambda}(\mu)v_x)_x, \end{cases}$$

$$-B(\lambda, \mu) = \begin{cases} \frac{\tilde{V}_m(\mu)S_0}{\tilde{K}_m(\mu) + |S_0 + \lambda|} = S_0\tilde{H}(\lambda, \mu), \\ \tilde{G}(\lambda, \mu). \end{cases}$$

The special feature of the first component of the splitting insures the positivity of the state  $u$ . With this notation, our optimal control method is the following:

*State equation.* For  $(\lambda, \mu) \in V$  solve

$$(3.7) \quad \begin{aligned} &(\tilde{D}(\mu)u_x, \phi_x) + (u\tilde{H}(\lambda, \mu), \phi) + (S_0\tilde{H}(\lambda, \mu), \phi) = 0, \\ &(\tilde{\Lambda}(\mu)v_x, \phi_x) + (\tilde{G}(\lambda, \mu), \phi) = 0, \quad \forall \phi \in H_0^1(0, e), \quad (u, v) \in V, \end{aligned}$$

where  $(\cdot, \cdot)$  denotes the  $L^2(0, e)$  inner product.

*Adjoint problem.* Given  $(\lambda, \mu)$  and the corresponding state  $(u, v) \in V$ , solve

$$(3.8) \quad \begin{aligned} &(\tilde{D}(\mu)\phi_x, p_x) + (\tilde{H}(\lambda, \mu)\phi, p) = ((u - \lambda)_x, \phi_x), \\ &(\tilde{\Lambda}(\mu)\phi_x, q_x) = ((v - \mu)_x, \phi_x), \quad \forall \phi \in H_0^1(0, e), \quad (p, q) \in V. \end{aligned}$$

*Gradient of the cost function.* With  $J(\lambda, \mu) = \frac{1}{2} \int_0^e |u_x - \lambda_x|^2 dx + \frac{1}{2} \int_0^e |v_x - \mu_x|^2 dx$  we have  $\nabla J(\lambda, \mu) = -(K_1, K_2) - (u - \lambda, v - \mu)$ , where  $(K_1, K_2) \in V$  is the solution of

$$(3.9) \quad \begin{aligned} (K_{1x}, \phi_x) &= \left( \frac{\partial}{\partial \lambda} \tilde{H}(\lambda, \mu) u \phi, p \right) + \left( \frac{\partial}{\partial \lambda} \tilde{H}(\lambda, \mu) S_0 \phi, p \right) + \left( \frac{\partial \tilde{G}}{\partial \lambda}(\lambda, \mu) \phi, q \right), \\ (K_{2x}, \phi_x) &= \left( \frac{\partial}{\partial \mu} \tilde{D}(\mu) \phi u_x, p_x \right) + \left( \frac{\partial}{\partial \mu} \tilde{H}(\lambda, \mu) u \phi, p \right) + \left( \frac{\partial}{\partial \mu} \tilde{\Lambda}(\mu) \phi v_x, q_x \right) \\ &\quad + \left( S_0 \frac{\partial \tilde{H}}{\partial \mu}(\lambda, \mu) \phi, p \right) + \left( \frac{\partial \tilde{G}}{\partial \mu}(\lambda, \mu) \phi, q \right), \quad \forall \phi \in H_0^1(0, e). \end{aligned}$$

The numerical approximation of (3.7), (3.8), (3.9) uses  $P_1$  Lagrangian finite elements on a regular partition of  $[0, e]$ . The optimization method is the Polak-Ribière conjugate gradient technique, see [13], [15]. We have tested the physical problem studied in [1], which corresponds to the following values of the constants:

$$\begin{aligned} D_0 &= 1.726 \cdot 10^{15}, & E &= 3.0 \cdot 10^4 \\ \Lambda_0 &= 2.12 \cdot 10^{-3}, & b &= 0.02, \\ V_0 &= 448, & E' &= 0.0, \\ K_0 &= 1.767 \cdot 10^{17}, & \Delta H &= 2.344 \cdot 10^4, \\ e &= 0.004, & R = 2, \quad T_0 &= 294. \end{aligned}$$

The boundary concentration  $S_0$  takes values between 1 and 10.

For  $T = T_0, D(T_0), \Lambda(T_0), V_m, K_m(T_0)$  are the coefficients of the usual uncoupled model with fixed coefficients, studied in [11]:

$$\begin{aligned} -DS_{xx} + \frac{V_m S}{K_m + S} &= 0, & -\Lambda T_{xx} - \frac{\Delta H V_m S}{K_m + S} &= 0 \\ S(0) = S(e) &= S_0, & T(0) = T(e) &= T_0. \end{aligned}$$

First we will compute the solutions  $(S, T)$  with  $S_0$  varying; then for fixed  $S_0$  we will consider  $\Delta H$  as a bifurcation parameter.

**3.3. Numerical results and conclusions.** First, the above method is applied to the coupled system (1.1) and to the corresponding fixed coefficients system, in order to compare their solutions. In the range of physical data, the two solutions are close to each other. Figure 1 shows separated solutions (in terms of temperature in the middle of the membrane) as the coefficient of physical  $\Delta H$  is increasing. In the following, this coefficient is denoted by  $\Delta H$ .

Various boundary concentrations  $S_0$  are tested to note the influence of this parameter upon the solution values.

For small  $\Delta H (\leq 1)$ , the larger  $S_0$  the faster the evolution of  $T = f(\Delta H)$  (Fig. 2); the result for the concentration is similar (Fig. 3).

For large values of  $\Delta H (> 8)$ , the temperature is independent of  $S_0$  (Fig. 4); however, the concentration is dependent on  $S_0$ ; for  $S_0 = 5$  the result is shown in Fig. 5.

Another class of interesting results deals with the asymptotic analysis of § 2.

For a fixed  $\Delta H$ , as  $S_0$  increases,  $S/S_0 \rightarrow 1$  (Fig. 6), when  $T \rightarrow T_\infty = v_\infty/b$ , for  $v_\infty$  the positive solution of the second order equation presented in Theorem 2.3. This numerical result is shown in Figure 7.

For large  $\Delta H$ , the results of Theorem 2.4 are partially illustrated by Figs. 8 and 9.

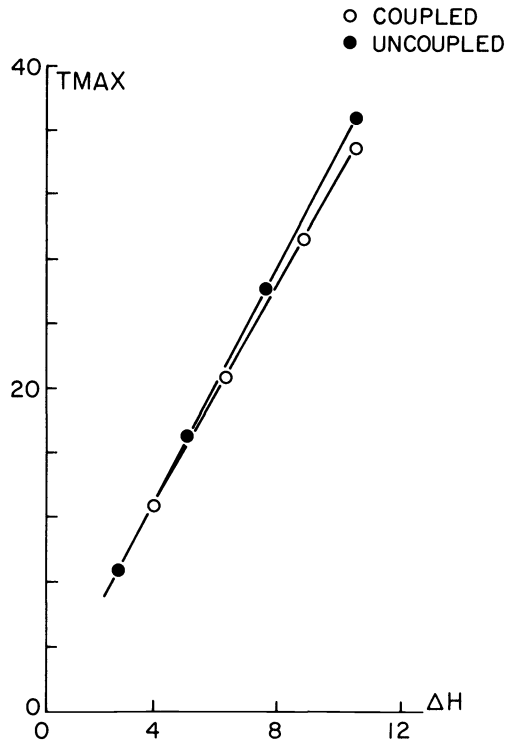


FIG. 1. Temperature at the center of the membrane for the coupled and uncoupled models.

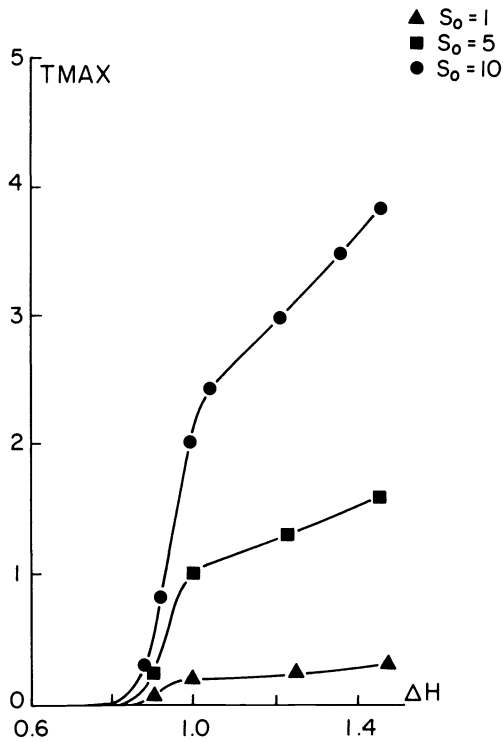


FIG. 2. Evolution of the temperature with respect to  $\Delta H$  for various values of  $S_0$ .

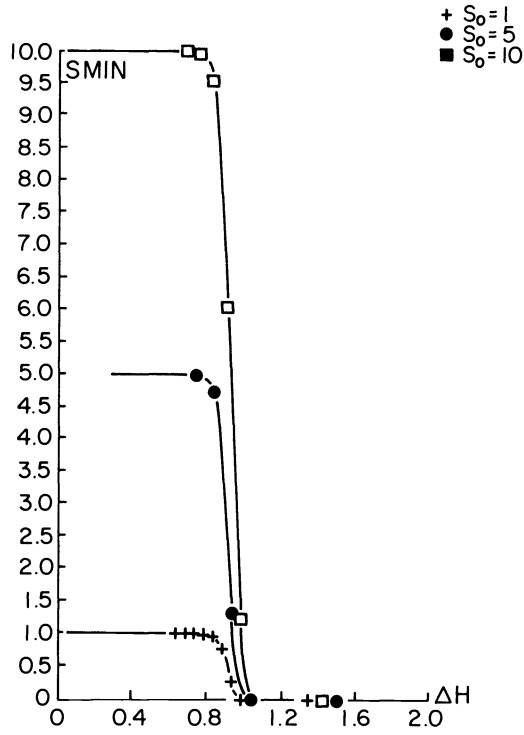


FIG. 3. Evolution of the concentration with respect to  $\Delta H$  for various values of  $S_0$ .

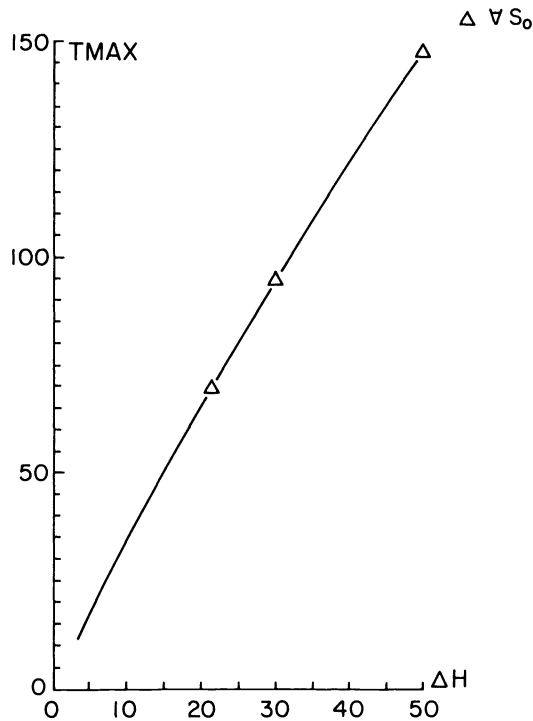


FIG. 4. Evolution of the temperature with respect to large  $\Delta H$  for  $S_0=5$ . For  $S_0=1$  or 10, the curve is unchanged.

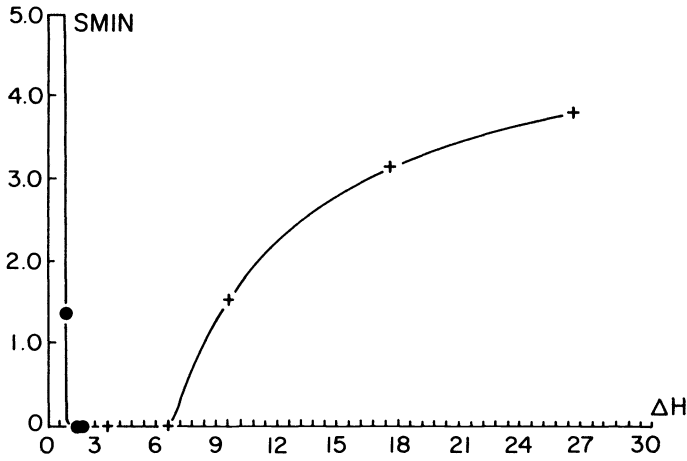


FIG. 5. Evolution of the concentration with respect to large  $\Delta H$ , for  $S_0 = 5$ .

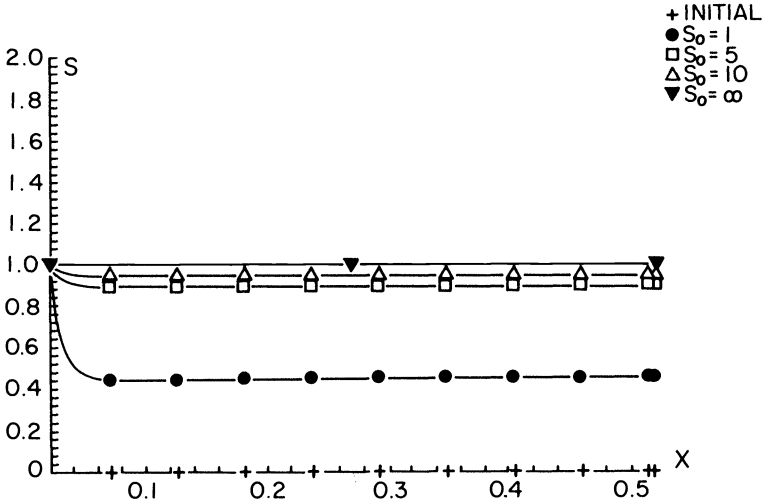


FIG. 6. Normalized concentration profiles, for large  $S_0$ .

The above results, obtained by an optimal control algorithm, are confirmed by a fixed point algorithm whose convergence implies the convergence of the optimal control algorithm. Nevertheless there remains a critical range of values for  $\Delta H$ , for which solutions are not reached numerically ( $1.5 \leq \Delta H \leq 6$ ). For these reasons it is interesting to adapt a continuation process in order to obtain the whole solution arc with respect to  $\Delta H$ . This method is presented in the following section.

**4. Numerical analysis: a continuation method.**

**4.1. Principle of the method.** The previous section discussed a method for finding solutions of the coupled system (1.1) for a fixed  $\Delta H$ . In this section, we describe a continuation method for solving nonlinear problems and an application to the computation of the solution branch  $\{T\}_S = \mathcal{F}(\Delta H)$  for a fixed  $S_0$ .

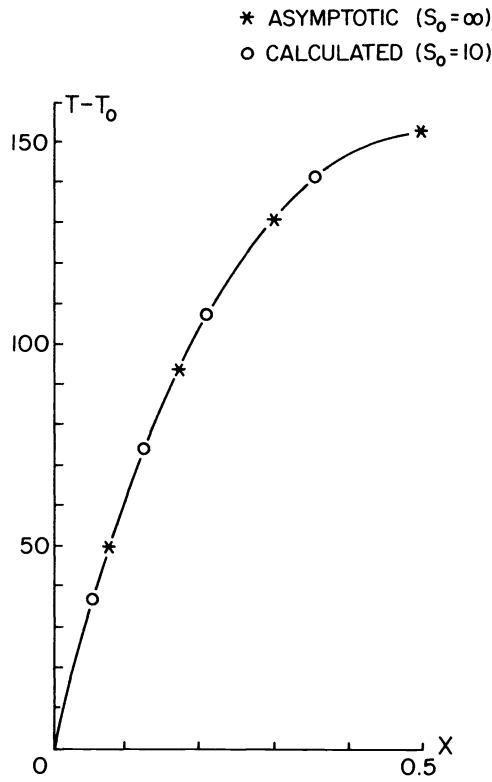


FIG. 7. Temperature profiles for large  $S_0$  (for  $S_0 = 1$  or  $5$ , the curve is the same).

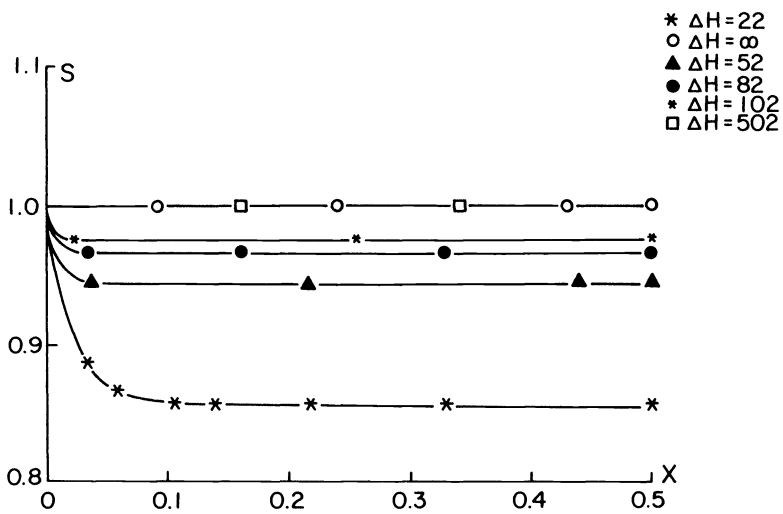


FIG. 8. Normalized concentration profiles as  $\Delta H \rightarrow \infty$  for  $S_0 = 5$ .



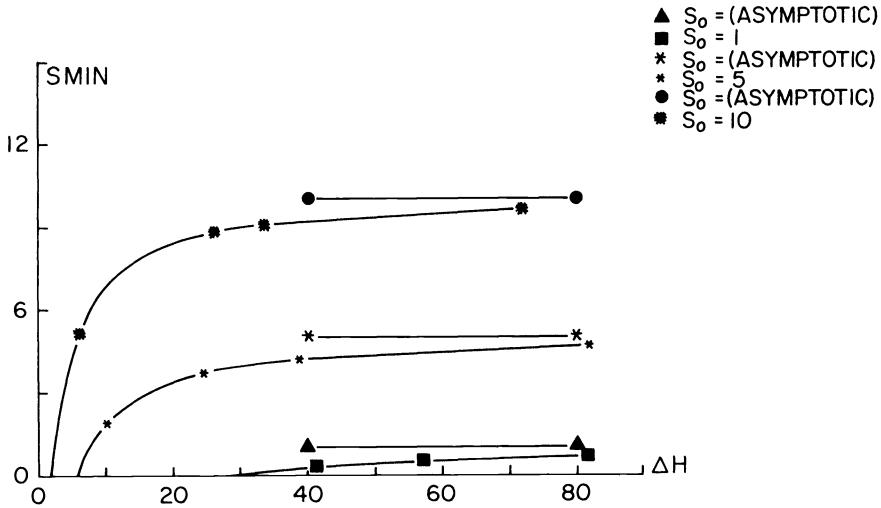


FIG. 9. Evolution of the concentration at the center of the membrane as  $\Delta H \rightarrow \infty$ , for various values of  $S_0$ .

We shall consider a class of nonlinear problems depending on a real parameter  $\lambda$

$$(4.1) \quad G(u, \lambda) = 0,$$

where  $G: B \times \mathbb{R} \rightarrow \mathbb{R}$ , and  $B$  is a Banach space. For the discrete form of (4.1),  $B$  is a finite dimensional space.

DEFINITION 4.1. A regular branch of solutions is a family of solutions of (4.1), depending twice continuously differentially on a parameter  $s$ ; we set

$$(4.2) \quad \Gamma_{ab} = \{(u(s), \lambda(s)), s_a \leq s \leq s_b\}.$$

Our purpose is to compute the regular branches of solutions of problem (4.1).

The standard approach is almost invariably to use  $\lambda$ , one of the naturally occurring parameters of the problem, but this procedure may fail or encounter difficulties when the Fréchet derivative  $G_u(u, \lambda)$  is singular (then  $(u, \lambda)$  is called a singular solution).

The basic idea to circumvent this is due to H.B. Keller [10] and consists in using a normal parametrization

$$u = u(s), \quad \lambda = \lambda(s),$$

which is defined using an auxiliary equation. We obtain the problem

$$(4.3) \quad G(u(s), \lambda(s)) = 0, \quad N(u(s), \lambda(s), s) = 0,$$

where  $N: B \times \mathbb{R}^2 \rightarrow \mathbb{R}$  defines the normal parameter  $s$ , on the arc of solutions.

Introduce then the new unknown  $x \in X = B \times \mathbb{R}$  and the operator  $P: X \times \mathbb{R} \rightarrow X$  defined by

$$(4.4) \quad x(s) = (u(s), \lambda(s))$$

and

$$(4.5) \quad P(x(s), s) = \begin{pmatrix} G(u(s), \lambda(s)) \\ N(u(s), \lambda(s), s) \end{pmatrix}.$$

The new problem is to find the solution  $x(s)$  of

$$(4.6) \quad P(x(s), s) = 0.$$

The main interest of this new formulation is that the ordinary limit points of (4.1) become regular solutions of (4.6) (see H.B. Keller [10] for more details). Let us make precise the concept of a limit point:

**DEFINITION 4.2.** Let  $\{u_0, \lambda_0\} \in B \times \mathbb{R}$  be a solution of problem (4.1). We say that  $\{u_0, \lambda_0\}$  is a *normal limit point* if

$$(4.7) \quad \dim N\left(\frac{\partial G}{\partial u}(u_0, \lambda_0)\right) = \text{codim } R\left(\frac{\partial G}{\partial u}(u_0, \lambda_0)\right) = 1,$$

$$(4.8) \quad \frac{\partial G}{\partial \lambda}(u_0, \lambda_0) \notin R\left(\frac{\partial G}{\partial u}(u_0, \lambda_0)\right).$$

A standard normalization is the so-called arclength parametrization

$$(4.9) \quad N(u(s), \lambda(s), s) = \|\dot{u}^2(s)\| + |\dot{\lambda}^2(s)| - 1,$$

where  $\dot{u}(s), \dot{\lambda}(s)$ , are the derivatives with respect to  $s$ .

The main justification of arclength continuation follows from:

**PROPOSITION 4.1.** *Any normal limit point of problem (4.1) is a regular solution of (4.6).*

For a proof, see Keller [10].

This new formulation enables us to solve problem (1.1) by a Newton method, even in a neighborhood of a normal limit point.

**4.2. Numerical solution of the system (1.1).** We want to find the whole solution arc of the coupled system (1.1) with respect to  $\Delta H$ . Using a normal parametrization, we transform the previous problem into the following:

Find  $\{S, T, \Delta H\}$  in  $B \times B \times \mathbb{R}^+$  satisfying

$$(4.10) \quad S = S(s), \quad T = T(s), \quad \Delta H = \Delta H(s), \quad s \in \mathbb{R}$$

such that

$$(4.11) \quad f_1(S, T, \Delta H) = 0, \quad f_2(S, T, \Delta H) = 0, \quad f_3(S, T, \Delta H, s) = 0$$

with

$$\begin{pmatrix} f_1(S, T, \Delta H) = 0 \\ f_2(S, T, \Delta H) = 0 \end{pmatrix} \Leftrightarrow (1.1)$$

and

$$f_3(S, T, \Delta H, s) = \dot{S}^2(s) + \dot{T}^2(s) + \dot{\Delta H}^2(s) - 1 = N(S, T, \Delta H, s).$$

The last equation is the arclength constraint. In practice, this equation is linearized as follows:

$$(4.12) \quad \begin{aligned} \tilde{f}_3(S, T, \Delta H, s) = & \frac{S(s) - S_2}{\Delta s_{MM_2}} \cdot \frac{S_2 - S_1}{\Delta s_{M_2M_1}} + \frac{T(s) - T_2}{\Delta s_{MM_2}} \cdot \frac{T_2 - T_1}{\Delta s_{M_2M_1}} \\ & + \frac{\Delta H(s) - \Delta H_2}{\Delta s_{MM_2}} \cdot \frac{\Delta H_2 - \Delta H_1}{\Delta s_{M_2M_1}} - 1 = 0, \end{aligned}$$

where  $M_1$  and  $M_2$  denote two solution points on the branch. For arclength approximations see [8], [14]. As shown in (4.12) a new point is computed by means of two previous solutions on the branch. These two points provide the path  $\Delta s$  between two consecutive solutions and an initialization point for the following solution. The coupled system (1.1) is discretized as in § 3. The numerical solution is approximated by a

Newton method. Let us denote by

$$(4.13) \quad F(S, T, \Delta H, s) = 0$$

the approximation of (4.11) with (4.12). The Newton algorithm may be written as follows:

$$M^{(p+1)} = M^{(p)} - (F'(M^{(p)}))^{-1} \cdot F(M^{(p)}), \quad \text{i.e.,} \quad M^{(p+1)} = M^{(p)} + \begin{pmatrix} u \\ v \\ w \end{pmatrix}$$

where  $(u, v, w)$  is the solution of

$$\begin{aligned} \left( \frac{\partial f_1}{\partial S} u + \frac{\partial f_1}{\partial T} v + \frac{\partial f_1}{\partial \Delta H} w \right)_{M^{(p)}} &= -f_1(M^{(p)}), \\ \left( \frac{\partial f_2}{\partial S} u + \frac{\partial f_2}{\partial T} v + \frac{\partial f_2}{\partial \Delta H} w \right)_{M^{(p)}} &= -f_2(M^{(p)}), \\ \left( \frac{\partial \tilde{f}_3}{\partial S} u + \frac{\partial \tilde{f}_3}{\partial T} v + \frac{\partial \tilde{f}_3}{\partial \Delta H} w \right)_{M^{(p)}} &= -\tilde{f}_3(M^{(p)}). \end{aligned}$$

The computation of  $M^{(p+1)}$  entails the inversion of a sparse linear system of order  $2n + 1$ . the algorithm stops when both system and continuation constraint (i.e.,  $\tilde{f}_3(S, T, \Delta H, s) = 0$ ) reach a value less than a given precision parameter.

Numerical results obtained by this process allow us to verify and complete the partial results of the optimal control algorithm. (See Figs. 10 and 11.) We note two “steps” in temperature corresponding to a great sensitivity of  $T$  with respect to  $\Delta H$  for two ranges of this parameter. Correspondingly, for  $S$ , one can see a large domain of free boundary type.

*Remark.* Figures 10 and 11 are obtained with  $\Delta H$  increasing. Experiment with decreasing  $\Delta H$  provides a nonphysical extension ( $S < 0$ ) of the quasi-linear part of the temperature branch. The corresponding phenomenon is shown in Fig. 10.

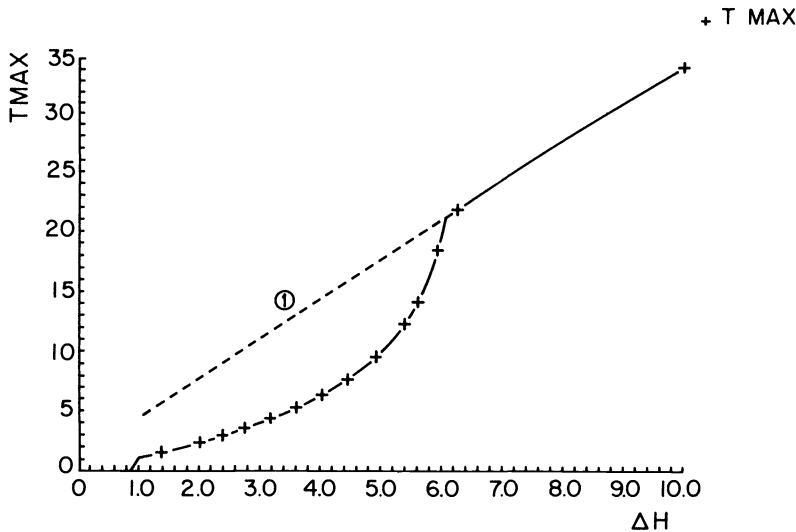
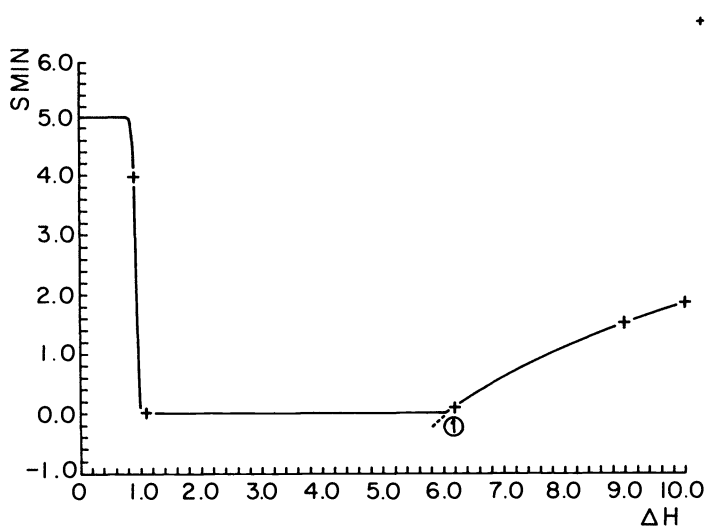


FIG. 10. Bifurcation diagram  $(T, \Delta H)$ . Curve 1 corresponds to a nonphysical solution ( $S < 0$ ).

FIG. 11. Bifurcation diagram ( $S, \Delta H$ ).

## REFERENCES

- [1] V. BILHOU-BOUGNOL, *Cinétique enzymatique d'une réaction exothermique en phase hétérogène et sous pression*, Thèse Doct. Ing., Saint-Etienne, 1976.
- [2] B. H. BRAMFORD AND C. F. TIPPER, *Comprehensive Chemical Kinetics*, vol. 6, Elsevier, Amsterdam, 1972.
- [3] C. M. BRAUNER AND B. NICOLAENKO, *On nonlinear eigenvalue problems which extend into free boundary problems*, Lecture Notes in Mathematics, 782, Springer-Verlag, New York, 1980, pp. 61-100.
- [4] J. CÉA AND G. GEYMONAT, *Une méthode de linéarisation via l'optimisation*, Symp. Math., Inst. Naz. di Alta Matematica, X (1972), pp. 431-451.
- [5] F. CONRAD, *Perturbation de problèmes aux valeurs propres non linéaires et problèmes à frontière libre*, Thèse Univ. Lyon 1, 1983.
- [6] M. DIXON AND E. C. WEBB, *Enzymes*, Longman, London, 1964.
- [7] J. GUYOT, *Etude mathématique et numérique d'un problème elliptique non linéaire avec points de retournement*, Thèse Doct. Ing., Lyon, 1981.
- [8] R. F. HEINEMANN, K. A. OVERHOLSER AND G. W. REDDIEN, *Multiplicity and stability of premixed laminar flames: an application of bifurcation theory*, Chem. Eng. Sci., 34 (1979), pp. 833-840.
- [9] H. B. KELLER, *Some positive problems suggested by nonlinear heat generation*, Bifurcation Theory and Nonlinear Eigenvalue Problems, J.B. Keller and S. Antman, eds., Benjamin, New York, 1969, pp. 217-255.
- [10] H. B. KELLER, *Numerical solution of bifurcation and nonlinear eigenvalue problems*, Applications of Bifurcation Theory, P. H. Rabinowitz, ed., Academic Press, New York, 1977, pp. 359-384.
- [11] J. P. KERNEVEZ, *Evolution et contrôle des systèmes bio-mathématiques*, Thèse Univ. Paris VI, 1972.
- [12] J. P. KERNEVEZ AND D. THOMAS, *Numerical analysis and control of some biochemical systems*, Appl. Math. Optim., 1 (1975), pp. 222-285.
- [13] C. LAINE, *Etude mathématique et numérique de modèles de turbulence en conduite plane*, Thèse Doct. Ing., Lyon, 1980.
- [14] C. LAINE AND L. REINHART, *Further numerical methods of the Falkner-Skan equation: Shooting and continuation techniques*, Tech. Rep. 177, INRIA, 1983.
- [15] E. POLAK, *Computational Methods in Optimization*, Academic Press, New York, 1971.
- [16] J. RICARD, *Cinétique et mécanismes d'actions des enzymes*, Dion, Paris, 1973.
- [17] D. H. SATTINGER, *Topics in Stability and Bifurcation Theory*, Lecture Notes in Mathematics, 309, Springer-Verlag, New York, 1973.
- [18] *Handbook of Chemistry and Physics*, CRC, 1969.

## CONSTRAINED LEAST SQUARES INTERVAL ESTIMATION\*

JANE E. PIERCE† AND BERT W. RUST‡

**Abstract.** We extend the classical least squares method for estimating confidence intervals to the rank deficient case, stabilizing the estimate by means of *a priori* side constraints. In order to avoid quadratic programming, we develop a suboptimal method which is in some ways similar to ridge regression but is quite different in that it provides an unambiguous criterion for the choice of the arbitrary parameter. We develop a method for choosing that parameter value and illustrate the procedure by applying it to an example problem.

**Key words.** confidence intervals, constrained least squares, deconvolution, first kind integral equations, ill-conditioned linear systems, interval estimation, regularization, unfolding

**1. Introduction.** In this paper we shall be concerned with the problem of obtaining confidence interval estimates from linear regression models with rank deficient or nearly rank deficient matrices. We assume that the model has the standard form

$$(1.1) \quad \mathbf{y} = \mathbf{K}\mathbf{x} + \mathbf{e}$$

where  $\mathbf{K}$  is the known  $m \times n$  matrix,  $\mathbf{x}$  is the unknown solution vector,  $\mathbf{y}$  is the vector of observations, and  $\mathbf{e}$  is a stochastic error vector satisfying

$$(1.2) \quad E(\mathbf{e}) = \mathbf{0}, \quad E(\mathbf{e}\mathbf{e}^T) = \mathbf{S}^2,$$

where  $E$  denotes the expectation operator. We assume, without loss of generality, that the covariance matrix  $\mathbf{S}^2$  is diagonal. In most applications  $\mathbf{y}$  is considered to be a sample from a multivariate normal distribution with unknown mean  $\bar{\mathbf{y}}$  which satisfies  $\mathbf{K}\mathbf{x} = \bar{\mathbf{y}}$ . We shall not be overly concerned here with the exact form of the  $\mathbf{y}$ -distribution, assuming only that the equi-probability contours are ellipsoidal and that for any confidence level  $\alpha < 1$  we can find a corresponding constant  $\mu$  so that the expression

$$(1.3) \quad (\bar{\mathbf{y}} - \mathbf{y})^T \mathbf{S}^{-2} (\bar{\mathbf{y}} - \mathbf{y}) \leq \mu^2$$

defines an  $\alpha$ -level confidence ellipsoid for the unknown  $\bar{\mathbf{y}}$ .

The classical linear estimation problem is to find, for a given  $n$ -vector  $\mathbf{w}$ , the best linear, unbiased estimator for the linear function

$$(1.4) \quad \phi(\mathbf{x}) = \mathbf{w}^T \mathbf{x}.$$

Assuming that  $\text{rank}(\mathbf{K}) = n$ , the solution is

$$\hat{\phi} = \mathbf{w}^T \hat{\mathbf{x}}$$

where  $\hat{\mathbf{x}}$  is the least squares solution vector defined by

$$(1.5) \quad \hat{\mathbf{x}} = (\mathbf{K}^T \mathbf{S}^{-2} \mathbf{K})^{-1} \mathbf{K}^T \mathbf{S}^{-2} \mathbf{y}.$$

An  $\alpha$ -level confidence interval  $[\phi_{lo}, \phi^{up}]$  for  $\phi$  is obtained from

$$(1.6) \quad \phi_{lo}^{up} = \mathbf{w}^T \hat{\mathbf{x}} \pm \sqrt{(\mu^2 - r_0) \mathbf{w}^T (\mathbf{K}^T \mathbf{S}^{-2} \mathbf{K})^{-1} \mathbf{w}}$$

---

\* Received by the editors June 11, 1983, and in final form April 11, 1984. Most of this work was done while the authors were at the Computer Sciences Division at Oak Ridge National Laboratory under contract W-7405-eng-26 with the U.S. Department of Energy.

† E. G. & G., ORTEC, Oak Ridge, Tennessee 37830. Present address, SAS Institute, Box 8000, Cary, North Carolina 27511-8000.

‡ Scientific Computing Division, National Bureau of Standards, Washington, DC 20234.

where  $r_0$  is the minimum of the sum of squared residuals, i.e.,

$$r_0 = \min_x \{(\mathbf{y} - \mathbf{Kx})^T \mathbf{S}^{-2}(\mathbf{y} - \mathbf{Kx})\} = (\mathbf{y} - \mathbf{K}\hat{\mathbf{x}})^T \mathbf{S}^{-2}(\mathbf{y} - \mathbf{K}\hat{\mathbf{x}}).$$

Since (1.3) defines a confidence ellipsoid in  $\mathbf{y}$ -space, it follows that

$$(\mathbf{Kx} - \mathbf{y})^T \mathbf{S}^{-2}(\mathbf{Kx} - \mathbf{y}) \leq \mu^2$$

or, equivalently,

$$(1.7) \quad (\mathbf{x} - \hat{\mathbf{x}})^T \mathbf{K}^T \mathbf{S}^{-2} \mathbf{K}(\mathbf{x} - \hat{\mathbf{x}}) \leq \mu^2 - r_0,$$

defines a confidence ellipsoid in  $\mathbf{x}$ -space. The confidence bounds  $\phi_{lo}$ ,  $\phi^{up}$  are just the values attained by  $\phi(\mathbf{x})$  on the two support planes of this latter confidence ellipsoid which are orthogonal to the vector  $\mathbf{w}$  (cf. [14, Appendix III]).

In the case  $\text{rank}(\mathbf{K}) < n$  the ellipsoid (1.7) is unbounded in some directions and the confidence intervals become  $(-\infty, +\infty)$  for any vector  $\mathbf{w}$  having a nonzero component in the null space of  $\mathbf{K}$ . In most applications it is not practical to pick  $\mathbf{w}$  without such a component, and in fact it is not even possible to unambiguously determine  $\text{rank}(\mathbf{K})$ . Therefore it is necessary to add some a priori side constraints to the problem in order to obtain nontrivial interval estimates. In this paper we shall add side constraints of the form

$$(1.8) \quad p_j \leq x_j \leq q_j, \quad j = 1, \dots, n,$$

where the  $p_j$  and  $q_j$  are known bounds obtained from external considerations.

The method that will be described here is basically a generalization and extension of the FERDOR method of radiation spectrum unfolding which was developed at Oak Ridge National Laboratory in the 1960's by Walter R. Burrus and his colleagues. The problem addressed by FERDOR is to give confidence interval estimates of quantities of the form

$$\int_a^b w(\mathcal{E})x(\mathcal{E}) d\mathcal{E},$$

where  $x(\mathcal{E})$  is an unknown radiation energy spectrum which is related to a measured pulse height spectrum  $y_i$  by

$$(1.9) \quad \int_a^b K_i(\mathcal{E})x(\mathcal{E}) d\mathcal{E} = y_i + e_i, \quad i = 1, \dots, m.$$

The  $K_i(\mathcal{E})$  are the response functions of the measuring instrument, and the  $e_i$  are random measuring errors. The functions  $w(\mathcal{E})$ , which are designed to exhibit the various desired aspects of the unknown spectrum, are called window functions, and we shall often refer to the vector  $\mathbf{w}$  as a window vector.

The FERDOR method has enjoyed great success in spite of the lack, until rather recently, of adequate, coherent documentation. A succinct description of the method has been given by Burrus et al. [4] in a paper which also briefly outlines the history of its development and gives references to earlier published descriptions. The method assumes that  $K_i(\mathcal{E}) \geq 0$ ,  $i = 1, \dots, m$  and  $x(\mathcal{E}) \geq 0$  for all energies  $\mathcal{E}$ . The basic discrete problems that must be solved are

$$(1.10) \quad \phi_{lo} = \min_x \{\mathbf{w}^T \mathbf{x} | (\mathbf{Kx} - \mathbf{y})^T \mathbf{S}^{-2}(\mathbf{Kx} - \mathbf{y}) \leq \mu^2, \mathbf{x} \geq \mathbf{0}\},$$

$$(1.11) \quad \phi^{up} = \max_x \{\mathbf{w}^T \mathbf{x} | (\mathbf{Kx} - \mathbf{y})^T \mathbf{S}^{-2}(\mathbf{Kx} - \mathbf{y}) \leq \mu^2, \mathbf{x} \geq \mathbf{0}\}.$$

In [13, Chapt. 5], it is shown that each of these problems can be solved by parametric quadratic programming, but because of the excessive amount of computation required, this is an expensive approach for most applications which require a large number of window vectors. The FERDOR approach is suboptimal in that it gives interval estimates that are wider than the optimally narrow intervals obtained from the quadratic programming procedures. The suboptimal estimates are obtained by an augmented least squares procedure which is similar in approach to ridge regression, and it is necessary to choose the value of an arbitrary parameter. The main contribution of the present work is to provide a procedure for the optimum choice of that parameter value. We also extend the previous work to allow different types of a priori side constraints and by iterating the procedure to obtain improved suboptimal bounds. In the next section we develop the new procedure without considering the statistical details which are described at length in [13, Chapt. 6].

**2. Development of the method.** Given an  $n$ -vector  $\mathbf{w}$  and an  $m \times n$  matrix  $\mathbf{K}$ , we wish to find bounds for  $\phi(\mathbf{x}) = \mathbf{w}^T \mathbf{x}$ , where

$$\mathbf{y} = \mathbf{K}\mathbf{x} + \mathbf{e}$$

and  $\mathbf{y}$  lies in the error ellipsoid

$$(2.1) \quad (\mathbf{y} - \mathbf{K}\mathbf{x})^T \mathbf{S}^{-2} (\mathbf{y} - \mathbf{K}\mathbf{x}) \leq \mu^2,$$

with  $\mathbf{S}^{-2}$  a positive definite diagonal matrix, and  $\mu$  any constant such that

$$\mu^2 \geq r_0 = \min_{\mathbf{x}} (\mathbf{y} - \mathbf{K}\mathbf{x})^T \mathbf{S}^{-2} (\mathbf{y} - \mathbf{K}\mathbf{x}).$$

The problems are then

$$(2.2) \quad \text{Find } \begin{matrix} \max \\ \min \end{matrix} \left\{ \mathbf{w}^T \mathbf{x} \mid (\mathbf{K}\mathbf{x} - \mathbf{y})^T \mathbf{S}^{-2} (\mathbf{K}\mathbf{x} - \mathbf{y}) \leq \mu^2 \right\}.$$

Often  $\mathbf{w}$  is taken successively as  $(1, 0, \dots, 0)^T$ ,  $(0, 1, \dots, 0)^T$ ,  $\dots$ ,  $(0, \dots, 1)^T$ , so that the quantities  $\mathbf{w}^T \mathbf{x}$  are estimates of the components of  $\mathbf{x}$ . If  $\mathbf{K}$  is an ill-conditioned matrix, the error ellipsoid (2.1), which we will call the  $S$ -ellipsoid, is greatly elongated in the directions of the eigenvectors corresponding to the small eigenvalues of  $\mathbf{K}^T \mathbf{S}^{-2} \mathbf{K}$ . (See Fig. 2.1.) The principal axes have the same directions as the eigenvectors of  $\mathbf{K}^T \mathbf{S}^{-2} \mathbf{K}$ , and their lengths are inversely proportional to the corresponding eigenvalues.

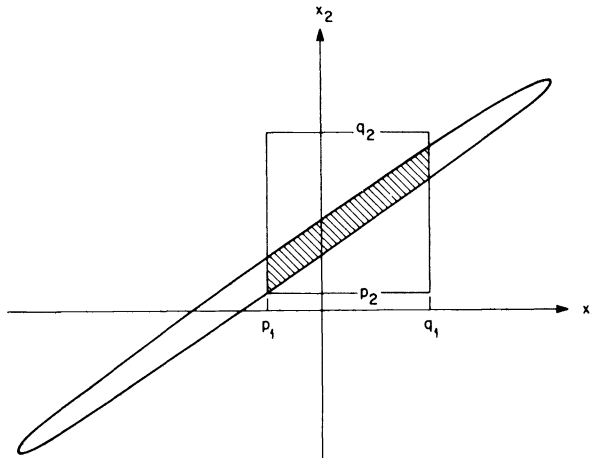


FIG. 2.1. The  $S$ -ellipsoid and the  $Q$ -box.

If  $\mathbf{w}$  has a component in the direction of one of the long axes of the  $S$ -ellipsoid, the bounds on  $\mathbf{w}^T \mathbf{x}$  will be very wide. We seek to improve these bounds by incorporating into the problem a priori knowledge of bounds on the components of  $\mathbf{x}$ ,  $p_j \leq x_j \leq q_j$ ,  $j = 1, \dots, n$  as shown in Fig. 2.1. The problems may now be stated as follows:

$$(2.3) \quad \text{Find } \phi_{lo}^{up} = \max_{\min} \left\{ \mathbf{w}^T \mathbf{x} \mid (\mathbf{K}\mathbf{x} - \mathbf{y})^T \mathbf{S}^{-2} (\mathbf{K}\mathbf{x} - \mathbf{y}) \leq \mu^2, p_j \leq x_j \leq q_j, j = 1, \dots, n \right\}.$$

Geometrically, the constraint region is the intersection of the  $S$ -ellipsoid and an interval in  $R^n$ , which we call the  $Q$ -box. Since the calculation of a solution cannot easily be done using the intersection of an ellipsoid and a box, we replace the  $Q$ -box by an ellipsoid which circumscribes it, namely,

$$\frac{1}{n} (\mathbf{x} - \mathbf{d})^T \mathbf{Q}^{-2} (\mathbf{x} - \mathbf{d}) \leq 1,$$

where

$$\mathbf{d} = \left( \frac{p_1 + q_1}{2}, \frac{p_2 + q_2}{2}, \dots, \frac{p_n + q_n}{2} \right)^T, \quad \text{and} \quad \mathbf{Q} = \text{diag} \left( \frac{q_1 - p_1}{2}, \dots, \frac{q_n - p_n}{2} \right).$$

We call this ellipsoid the  $Q$ -ellipsoid and remark here that, unless a mistake has been made in the analysis of the problem, the  $S$ - and  $Q$ -ellipsoids have a nonempty intersection.

The intersection of two ellipsoids is no easier to handle computationally than the intersection of a box and an ellipsoid. One strategy is to find another ellipsoid which contains the intersection of the  $S$  and  $Q$  ellipsoids. W. Kahan [9] has defined a "tight" circumscribing ellipsoid about the intersection of two ellipsoids with common centers, but there is no guarantee that the  $S$ - and  $Q$ -ellipsoids have that property. Consequently, we make one more suboptimizing step and take a convex linear combination of the  $S$ - and  $Q$ -ellipsoids. The problems now become:

$$(2.4) \quad \phi_{lo}^{up} = \max_{\min} \left\{ \mathbf{w}^T \mathbf{x} \mid \eta \cdot \frac{1}{\mu^2} (\mathbf{K}\mathbf{x} - \mathbf{y})^T \mathbf{S}^{-2} (\mathbf{K}\mathbf{x} - \mathbf{y}) + (1 - \eta) \cdot \frac{1}{n} (\mathbf{x} - \mathbf{d})^T \mathbf{Q}^{-2} (\mathbf{x} - \mathbf{d}) \leq 1 \right\},$$

$$0 \leq \eta \leq 1,$$

where  $\eta$  determines how much of each ellipsoid is taken. It can be shown that every such convex combination of the  $S$ - and  $Q$ -ellipsoids is an ellipsoid which contains the original constraint region, i.e., the intersection of the  $S$ -ellipsoid and the  $Q$ -box. The constraint in (2.4) can be written

$$(\mathbf{A}\mathbf{x} - \mathbf{p})^T \begin{pmatrix} \frac{\eta}{\mu^2} \mathbf{S}^{-2} & 0 \\ 0 & \frac{1-\eta}{n} \mathbf{Q}^{-2} \end{pmatrix} (\mathbf{A}\mathbf{x} - \mathbf{p}) \leq 1,$$

where

$$\mathbf{A} = \begin{pmatrix} \mathbf{K} \\ \mathbf{I} \end{pmatrix}, \quad \mathbf{p} = \begin{pmatrix} \mathbf{y} \\ \mathbf{d} \end{pmatrix}.$$

Now define new parameters

$$\tau = \left( \frac{1-\eta}{\eta} \right) \mu^2 \quad \text{and} \quad \mathbf{V}^{-2}(\tau) = \begin{pmatrix} \mathbf{S}^{-2} & 0 \\ 0 & \frac{\tau}{n} \mathbf{Q}^{-2} \end{pmatrix}.$$



The constraint can then be rewritten as

$$(\mathbf{Ax} - \mathbf{p})^T \mathbf{V}^{-2}(\tau)(\mathbf{Ax} - \mathbf{p}) \leq \tau + \mu^2,$$

with  $\tau$  chosen from the interval  $[0, +\infty)$ . Note that the matrix  $\mathbf{V}^{-2}(\tau)$  would be rank deficient only if one or more of the  $x_j$  were completely determined by a priori information. We assume in the following that any such  $x_j$  have been removed from the problem. By conventional least squares,

$$(2.5) \quad \rho_0 = \min_{\mathbf{x}} (\mathbf{Ax} - \mathbf{p})^T \mathbf{V}^{-2}(\tau)(\mathbf{Ax} - \mathbf{p})$$

is attained at

$$\tilde{\mathbf{x}} = [\mathbf{A}^T \mathbf{V}^{-2}(\tau) \mathbf{A}]^{-1} \mathbf{A}^T \mathbf{V}^{-2}(\tau) \mathbf{p},$$

or

$$\tilde{\mathbf{x}} = \left[ \mathbf{K}^T \mathbf{S}^{-2} \mathbf{K} + \frac{\tau}{n} \mathbf{Q}^{-2} \right]^{-1} \left[ \mathbf{K}^T \mathbf{S}^{-2} \mathbf{y} + \frac{\tau}{n} \mathbf{Q}^{-2} \mathbf{d} \right].$$

Notice here the similarity to ridge regression (cf. [7], [8], [10]), where

$$\mathbf{x}^* = [\mathbf{K}^T \mathbf{S}^{-2} \mathbf{K} + \lambda \mathbf{I}]^{-1} \mathbf{K}^T \mathbf{S}^{-2} \mathbf{y}$$

would be the ridge estimate of  $\mathbf{x}$ .

For a discussion of the technique of ridge regression, see Hoerl and Kennard [7] who point out that the variance of the estimate of  $\mathbf{x}$  is reduced, at the cost of some bias, the bias squared being a continuous monotonically increasing function of  $\lambda$ . Unfortunately, as Brown and Beattie show in [2], the bias produced by ridge regression can be large, and since the expression for squared bias involves  $\mathbf{x}$ , bias cannot be accurately estimated. One advantage of the interval estimation technique is that for any  $\tau \in [0, \infty)$  the interval  $[\phi_{lo}, \phi^{up}]$  contains  $\mathbf{w}^T \mathbf{x}$  with at least the same confidence level as that associated with the original error ellipsoid. Also, there is no universally accepted way to choose the  $\lambda$  of ridge regression, but the interval estimation strategy provides a criterion for choosing  $\tau$ . *Since all values of  $\tau$  produce valid confidence intervals, one should choose that value which yields the narrowest interval.*

The problems now are: Find

$$\begin{aligned} & \max_{\min} \left\{ \mathbf{w}^T \mathbf{x} \mid (\mathbf{Ax} - \mathbf{p})^T \mathbf{V}^{-2}(\tau)(\mathbf{Ax} - \mathbf{p}) \leq \tau + \mu^2 \right\} \\ & = \max_{\min} \left\{ \mathbf{w}^T \mathbf{x} \mid (\mathbf{x} - \tilde{\mathbf{x}})^T (\mathbf{A}^T \mathbf{V}^{-2}(\tau) \mathbf{A})(\mathbf{x} - \tilde{\mathbf{x}}) \leq \tau + \mu^2 - \rho_0 \right\}. \end{aligned}$$

The solutions, using Lagrange multipliers, are

$$(2.6) \quad \phi_{lo}^{up} = \mathbf{w}^T \tilde{\mathbf{x}} \pm \sqrt{\tau + \mu^2 - \rho_0} \sqrt{\mathbf{w}^T (\mathbf{A}^T \mathbf{V}^{-2}(\tau) \mathbf{A})^{-1} \mathbf{w}}.$$

Note that  $\tilde{\mathbf{x}}$ ,  $\rho_0$  and  $\mathbf{V}^{-2}$ , and hence  $\phi^{up}$  and  $\phi_{lo}$  are dependent on the choice of  $\tau$ . We wish to choose a  $\tau$  (if one exists) giving the minimum interval width. Let

$$\mathcal{L}^2 = (\tau + \mu^2 - \rho_0) \mathbf{w}^T (\mathbf{A}^T \mathbf{V}^{-2}(\tau) \mathbf{A})^{-1} \mathbf{w}.$$

We take  $\partial \mathcal{L}^2 / \partial \tau$  and solve the equation

$$\frac{\partial \mathcal{L}^2}{\partial \tau} = 0 \quad \text{for } \tau.$$

Recall that  $\mathbf{A}^T \mathbf{V}^{-2} \mathbf{A} = \mathbf{K}^T \mathbf{S}^{-2} \mathbf{K} + (\tau/n) \mathbf{Q}^{-2}$ , so that when  $\tau = 0$ , the problem reduces to the one corresponding to the  $S$ -ellipsoid, while as  $\tau$  increases, the  $S$ -ellipsoid becomes insignificant compared to the  $Q$ -ellipsoid. For a typical ill-conditioned problem, we might expect graphs of  $\mathcal{L}^2$  versus  $\tau$  and  $\partial \mathcal{L}^2 / \partial \tau$  versus  $\tau$  to look somewhat like the ones in Fig. 2.2. The shapes of the graphs have been verified by trial examples.

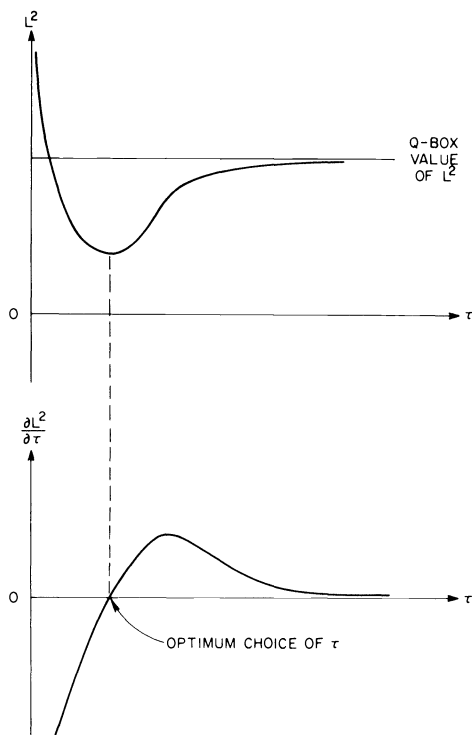


FIG. 2.2. Choice of  $\tau$ .

To solve  $\partial \mathcal{L}^2 / \partial \tau = 0$ , we need to invert  $\mathbf{A}^T \mathbf{V}^{-2} \mathbf{A} = \mathbf{K}^T \mathbf{S}^{-2} \mathbf{K} + (\tau/n) \mathbf{Q}^{-2}$ . It is convenient to make the following change of variables. Let

$$\mathbf{x}' = \mathbf{Q}^{-1} \mathbf{x}.$$

We then have

$$\mathbf{A} \mathbf{x} = \begin{pmatrix} \mathbf{K} \\ \mathbf{I} \end{pmatrix} \mathbf{Q} \mathbf{x}' = \begin{pmatrix} \mathbf{K} \mathbf{Q} \\ \mathbf{Q} \end{pmatrix} \mathbf{x}'$$

and the least squares solution of (2.5) is

$$\tilde{\mathbf{x}}' = \left( \mathbf{Q} \mathbf{K}^T \mathbf{S}^{-2} \mathbf{K} \mathbf{Q} + \frac{\tau}{n} \mathbf{I} \right)^{-1} \left( \mathbf{Q} \mathbf{K}^T \mathbf{S}^{-2} \mathbf{y} + \frac{\tau}{n} \mathbf{Q}^{-1} \mathbf{d} \right).$$

As in (2.6), we have

$$(2.6') \quad \phi_{lo}^{up} = \mathbf{w}^T (\mathbf{Q} \tilde{\mathbf{x}}') \pm \sqrt{\tau + \mu^2 - \rho_0} \sqrt{\mathbf{w}^T \mathbf{Q} \left( \mathbf{Q} \mathbf{K}^T \mathbf{S}^{-2} \mathbf{K} \mathbf{Q} + \frac{\tau}{n} \mathbf{I} \right)^{-1} \mathbf{Q} \mathbf{w}}.$$

Now it can be seen how the change of variables helps in writing the inverse. Consider

the singular value decomposition of  $\mathbf{S}^{-1}\mathbf{K}\mathbf{Q}$ ,

$$\mathbf{S}^{-1}\mathbf{K}\mathbf{Q} = \mathbf{L}\mathbf{\Sigma}\mathbf{R}^T,$$

where  $\mathbf{L}$  is an  $m \times m$  orthogonal matrix,  $\mathbf{R}$  is an  $n \times n$  orthogonal matrix, and  $\mathbf{\Sigma}$  is the  $m \times n$  singular value matrix. Using the singular value decomposition,

$$\left( \mathbf{Q}\mathbf{K}^T\mathbf{S}^{-2}\mathbf{K}\mathbf{Q} + \frac{\tau}{n}\mathbf{I} \right) = \mathbf{R} \left( \mathbf{\Sigma}^T\mathbf{\Sigma} + \frac{\tau}{n}\mathbf{I} \right) \mathbf{R}^T.$$

Notice that  $\mathbf{\Sigma}^T\mathbf{\Sigma} = \text{diag}(\sigma_1^2, \dots, \sigma_n^2)$ , where some of the  $\sigma_j$  may be 0. Now the inverse can easily be written:

$$\left[ \mathbf{Q}\mathbf{K}^T\mathbf{S}^{-2}\mathbf{K}\mathbf{Q} + \frac{\tau}{n}\mathbf{I} \right]^{-1} = \mathbf{R} \text{diag} \left( \frac{n}{n\sigma_1^2 + \tau}, \dots, \frac{n}{n\sigma_n^2 + \tau} \right) \mathbf{R}^T.$$

It is now straightforward, but tedious, to calculate the partial derivative  $\partial\mathcal{L}^2/\partial\tau$ , where

$$\mathcal{L}^2 = (\tau + \mu^2 - \rho_0) \left[ \mathbf{w}^T \mathbf{Q} \left( \mathbf{Q}\mathbf{K}^T\mathbf{S}^{-2}\mathbf{K}\mathbf{Q} + \frac{\tau}{n}\mathbf{I} \right)^{-1} \mathbf{Q}\mathbf{w} \right].$$

The result is

$$\begin{aligned} \frac{\partial\mathcal{L}^2}{\partial\tau} = & -\frac{(\tau + \mu^2 - \rho_0)}{n} \sum_{j=1}^n \frac{v_j^2}{(\sigma_j^2 + \tau/n)^2} \\ & + \left\{ 1 - \frac{1}{n} \sum_{j=1}^n \frac{(z_j + \tau r_j/n)^2}{(\sigma_j^2 + \tau/n)^2} + \frac{2}{n} \sum_{j=1}^n \frac{(z_j + \tau r_j/n)r_j}{(\sigma_j^2 + \tau/n)} - \frac{1}{n} \sum_{j=1}^n r_j^2 \right\} \sum_{j=1}^n \frac{v_j^2}{(\sigma_j^2 + \tau/n)}, \end{aligned}$$

where

$$\mathbf{r} = \mathbf{R}^T\mathbf{Q}^{-1}\mathbf{d}, \quad \mathbf{v} = \mathbf{R}^T\mathbf{Q}\mathbf{w}, \quad \mathbf{z} = \mathbf{\Sigma}^T\mathbf{L}^T\mathbf{S}^{-1}\mathbf{y}.$$

An equation solver may be used to seek values of  $\tau$  at which this derivative is zero. We supply lower and upper bounds for  $\tau$  and use an adaptation of Brent's ZERO ([1, Chapt. 4]). In conventional ridge regression, the columns of  $\mathbf{K}$  would be normalized to have zero means and unit variances; we choose initial bounds for  $\tau$  in terms of the norms of the columns of  $\mathbf{S}^{-1}\mathbf{K}\mathbf{Q}$ . If the problem is a well-conditioned one, ZERO may fail to find an axis crossing because  $\partial\mathcal{L}^2/\partial\tau > 0$  for all  $\tau$  (see Fig. 2.3a). In that case, we set  $\tau = 0$  and solve the unconstrained problem. If the a priori bounds are the best obtainable, ZERO also fails since  $\partial\mathcal{L}^2/\partial\tau < 0$  for all  $\tau$ , indicating that the  $Q$ -box provides the best bounds. (See Fig. 2.3b.)

Notice that  $\mathbf{w}$  does not enter into the singular value decomposition of  $\mathbf{S}^{-1}\mathbf{K}\mathbf{Q}$ , so that  $\tau$  can be found for any number of window vectors  $\mathbf{w}$  without doing the  $SVD$  again. In particular,  $\mathbf{w}$  can successively be taken to be  $(1, 0, \dots, 0)^T$ ,  $(0, 1, 0, \dots, 0)^T$ ,  $\dots$ ,  $(0, 0, \dots, 1)^T$  to find new bounds on  $x_1, \dots, x_n$ . The bounds thus obtained define a new interval in  $R^n$  which is guaranteed to contain the intersection of the  $S$ -ellipsoid with the original a priori constraint region. Hopefully the new bounds are all better than the original ones. If they are not better for some of the  $x_j$ , then the original bounds are retained in those cases. The new vector interval can then be used to define a new  $Q$ -matrix and  $\mathbf{d}$ -vector and the whole process can be iterated to improve the bounds further. At each step of the iteration the current  $Q$ -box is not necessarily a 100% guaranteed vector interval for the solution  $\mathbf{x}$ , but it is guaranteed to contain the intersection of the original 100% a priori constraint box with the  $S$ -ellipsoid, so it defines confidence intervals for the  $x_j$  with confidence levels that are at least as great as those of the  $S$ -ellipsoid. The idea in iterating this type of calculation was first

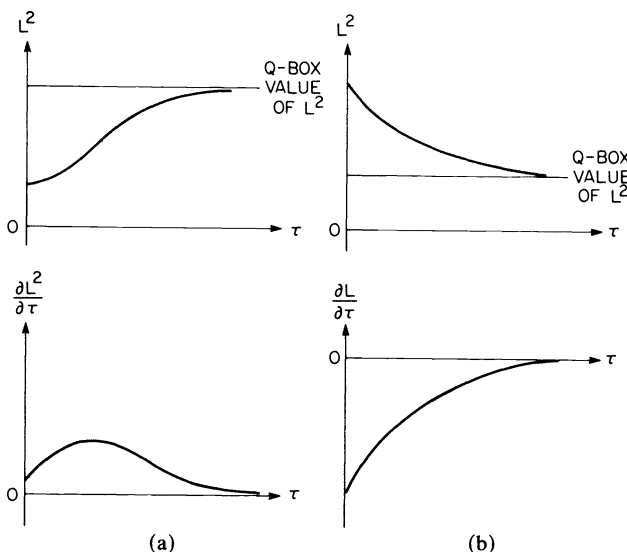


FIG. 2.3. a. Well-conditioned problem. b. A priori bounds best obtainable.

suggested by W. R. Burrus [3, Chapt. 9] and was briefly discussed by M. T. Heath [6, Chapt. 3]. The iteration process is expensive in our procedure because the singular value decomposition must be repeated at each step. This is not a prohibitive disadvantage, however, because the iteration converges very quickly. Two or three iterations have been sufficient for every problem we have tried. A referee suggested reducing the calculations by using a bidiagonalization of  $S^{-1}KQ$  (cf. Elden [5]) rather than the full singular value decomposition.

It is possible to use extra knowledge about the solution vector  $x$  to improve the bounds even more. For example, suppose  $0 \leq x_1 \leq \dots \leq x_n$ . Then  $x$  can be written

$$x = Pu, \quad u \geq 0,$$

where

$$P = \begin{pmatrix} 1 & 0 & 0 & \dots & 0 \\ 1 & 1 & 0 & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots \\ 1 & 1 & 1 & \dots & 1 \end{pmatrix}.$$

The initial bounds on the  $u_j$  can be obtained from

$$p_1 \leq u_1 \leq q_1, \quad p_j - q_{j-1} \leq u_j \leq q_j - p_{j-1}, \quad j = 2, 3, \dots, n.$$

Now suppose that we want to find  $\max_{\min}(x_1 + x_n)$ . Then  $w^T = (1, 0, 0, 1, 0, \dots, 0)$ , and  $w^T x = w^T P u$ . The matrix  $P$  is essentially a "shape" matrix, incorporating a priori knowledge of the shape of the solution. The vector  $w$  is a "window" vector determining which linear combination of components of  $x$  we look at. The final problems are then:

find

$$\max \left\{ (w^T P) u \mid (K P u - y)^T S^{-2} (K P u - y) \leq \mu^2 \right\}.$$

We replace  $K$  by  $K' = KP$ , and the initial bounds on the  $u$  are used to form a matrix

$Q'$  analogous to the  $Q$  in the original problem. The transformation of variables is then  $u' = (Q')^{-1}u$ .

**3. An example.** We now present a well-known integral equation problem as an example illustrating the method and the use of a priori constraints on the solution  $x$ . The problem was originally given by Phillips [11], and was discussed by Rust and Burrus ([13, § 1.5]). The problem is to solve

$$\int_{-6}^6 K(t, s)x(s) ds = y(t), \quad |t| \leq 6,$$

where

$$K(t, s) = \begin{cases} 1 + \cos \frac{\pi(s-t)}{3}, & |s-t| \leq 3, \quad |t| \leq 6, \\ 0, & |s-t| \geq 3, \quad |t| \leq 6, \end{cases}$$

and

$$y(t) = (6 - |t|) \left[ 1 + \frac{1}{2} \cos \frac{\pi t}{3} \right] + \frac{9}{2\pi} \sin \frac{\pi |t|}{3}, \quad |t| \leq 6.$$

The solution is

$$x(s) = \begin{cases} 1 + \cos \left( \frac{\pi s}{3} \right), & |s| \leq 3, \\ 0, & |s| > 3. \end{cases}$$

We present a  $60 \times 41$  discretization of the problem. Let

$$s_j = -3 + \frac{3}{20}(j-1), \quad j = 1, \dots, 41,$$

$$t_i = -6 + \frac{1}{5}(i-5), \quad i = 1, \dots, 60,$$

$$K_{ij} = \begin{cases} \left[ 1 + \cos \left( \frac{\pi}{3}(s_j - t_i) \right) \right] \omega_j, & |s_j - t_i| \leq 6, \quad |t_i| \leq 6, \\ 0, & \text{otherwise,} \end{cases}$$

where the  $\omega_j$  are the quadrature weights for the implied integration. Using Simpson's rule, those weights become

$$\omega_1 = \omega_{41} = \frac{1}{20},$$

$$\omega_{2k} = \frac{1}{5}, \quad k = 1, 2, \dots, 20, \quad \omega_{2k+1} = \frac{1}{10}, \quad k = 1, 2, \dots, 19.$$

The discretized right-hand side becomes

$$y_i = (6 - |t_i|) \left( 1 + \frac{1}{2} \cos \left( \frac{\pi}{3} t_i \right) \right) + \frac{9}{2\pi} \sin \left( \left| \frac{\pi}{3} t_i \right| \right),$$

and the discretized solution is

$$x_j = \begin{cases} 1 + \cos \left( \frac{\pi s_j}{3} \right), & |s_j| \leq 3, \\ 0, & |s_j| \geq 3. \end{cases}$$

Note that  $K_{ij}$  and  $x_j$  are always nonnegative and that  $x$  is symmetric about  $s=0$ ,  $x_1 \leq x_2 \leq \dots \leq x_{20} \leq x_{21} \leq \dots \leq x_{41}$ . Even if we did not know the true solution  $x(s)$ , we could deduce that it must be symmetric because  $y(t)$  is symmetric and the shape

of the kernel, considered as a function of  $s$ , is the same for all values of  $t$ . In order to derive the initial upper bounds for the  $x_j$ , we can use the following technique which works for any problem with all of the  $K_{ij}$  and  $x_j$  nonnegative (see [12, Chapt. 2]).

For all values  $j^*$  of the subscript  $j$  and all  $i$ , we have

$$K_{ij^*}x_{j^*} \leq \sum_{j=1}^n K_{ij}x_j = (Kx)_i.$$

Dividing by  $K_{ij^*}$ , we get

$$x_{j^*} \leq \frac{(Kx)_i}{K_{ij^*}} \quad \text{for all } i,$$

and hence

$$(3.1) \quad x_j \leq \min_{1 \leq i \leq m} \frac{(Kx)_i}{K_{ij}}.$$

We now need to find upper bounds for the  $(Kx)_i$ . From the error ellipsoid (2.1), we have

$$(\mathbf{Kx} - \mathbf{y})^T \mathbf{S}^{-2} (\mathbf{Kx} - \mathbf{y}) \leq \mu^2,$$

where

$$\mathbf{S}^{-2} = \text{diag} \left( \frac{1}{s_1^2}, \dots, \frac{1}{s_m^2} \right).$$

In this example, we let  $s_i = .0001 y_i$ . Equation (2.1) can be written in the form

$$\sum_{i=1}^m \frac{[(\mathbf{Kx} - \mathbf{y})_i]^2}{s_i^2} \leq \mu^2,$$

hence

$$\frac{(\mathbf{Kx} - \mathbf{y})_i^2}{s_i^2} \leq \mu^2 \quad \text{for all } i = 1, \dots, m,$$

or

$$(\mathbf{Kx})_i \leq y_i + \mu s_i.$$

Substituting in (3.1), we have

$$x_j \leq \min_{1 \leq i \leq m} \left\{ \frac{y_i + \mu s_i}{K_{ij}} \right\}, \quad j = 1, \dots, n.$$

In this way we obtain the initial upper bounds  $q_j$  of § 2, and we set  $p_j = 0, j = 1, \dots, n$ . Note that the bounds  $[p_j, q_j]$  in this case are not guaranteed to contain the  $x_j$  with 100% confidence but they are guaranteed to contain the intersection of the error ellipsoid with the positive orthant, and that intersection is the basic constraint region for this example. In practice, initial bounds computed by this method are always extremely conservative and in fact do provide 100% confidence boxes. In addition to using a priori constraints and initial bounds on  $\mathbf{x}$ , we can try to improve the bounds by solving a slightly less ambitious problem. Instead of solving for bounds on all of the  $x_j$ , we find bounds for an average of the  $x_j$ 's. For example,

$$\bar{x}_j \sim \frac{1}{2\Delta s} \int_{s_{j-1}}^{s_{j+1}} x(s) ds.$$

The integral may be approximated by using a 3-point Simpson's rule, yielding

$$\bar{x}_{2j} \cong \frac{1}{6}(x_{2j-1} + 4x_{2j} + x_{2j+1}), \quad j = 1, \dots, 20.$$

For a  $60 \times 41$  example, we let the  $j$ th window vector  $w_j = \frac{1}{6}(0, \dots, 1, 4, 1, \dots, 0)^T$ , with the 4 in the  $2j$ th place, and estimate  $\bar{x}_2, \bar{x}_4, \dots, \bar{x}_{40}$ . Figure 3.1 compares the pointwise estimates with 3-point Simpson averaging for the  $60 \times 41$  problem with a symmetric hump constraint (see below). As can be seen from the graphs, averaging greatly improves bounds in this case. Next, we observe the effect of a priori knowledge of  $x$  by solving the  $60 \times 41$  problem three ways with Simpson 3-point averaging.

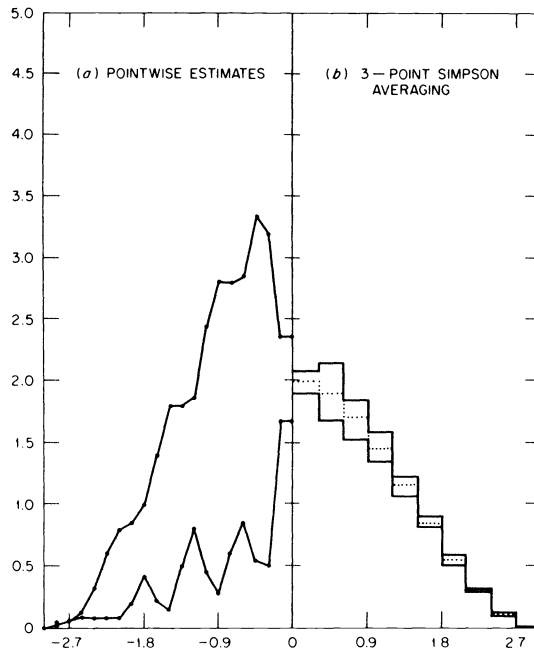


FIG. 3.1.  $60 \times 41$ , symmetric hump constraint.

We first solve the problem using no knowledge of  $x$  except the initial bounds. Next, we incorporate the knowledge of a “hump” in  $x$ ,  $0 < x_1 \cong x_2 \cong \dots \cong x_{21}$ ,  $x_{22} \cong x_{23} \cong \dots \cong x_{41}$ , but we do not assume symmetry. For this case the  $41 \times 41$  “shape” matrix is the following:

$$P = \begin{pmatrix} T & 0 \\ 0 & U \end{pmatrix},$$

where  $T$  and  $U$  are  $21 \times 21$  and  $20 \times 20$  triangular matrices, respectively, of the form

$$T = \begin{pmatrix} 1 & 0 & 0 & \dots & 0 \\ 1 & 1 & 0 & \dots & 0 \\ 1 & 1 & 1 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & 1 & 1 & \dots & 1 \end{pmatrix}, \quad U = \begin{pmatrix} 1 & 1 & 1 & \dots & 1 & 1 & 1 \\ 1 & 1 & 1 & \dots & 1 & 1 & 0 \\ 1 & 1 & 1 & \dots & 1 & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 1 & 0 & 0 & \dots & 0 & 0 & 0 \end{pmatrix}.$$

Finally, we use all we know about  $\mathbf{x}$ , namely that  $0 \leq x_1 \leq x_2 \leq \dots \leq x_{20} \leq x_{21} \leq x_{22} \leq \dots \leq x_{41}$  with  $x_j = x_{42-j}, j = 1, \dots, 20$ . For this "symmetric hump" case, the shape matrix has the form

$$\mathbf{P} = \begin{pmatrix} \mathbf{T} \\ \tilde{\mathbf{U}} \end{pmatrix},$$

with  $\mathbf{T}$  as defined above, and  $\tilde{\mathbf{U}}$  is the  $20 \times 21$  matrix formed by adjoining a column of zeros to  $\mathbf{U}$ , i.e.,  $\tilde{\mathbf{U}} = (\mathbf{U} \mathbf{0})$ . In this case we have effectively reduced the size of the problem by half. Figure 3.2 compares the nonnegativity only and nonsymmetric hump solutions. For the case of the symmetric hump, refer to Fig. 3.1. From this example, the advantage of incorporating all possible knowledge of  $\mathbf{x}$  is clear.

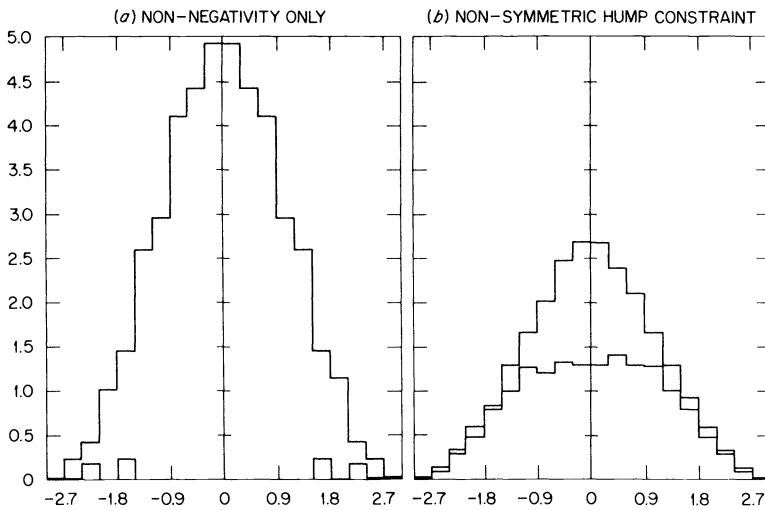


FIG. 3.2.  $60 \times 41$ , 3-point Simpson averaging.

It is reasonable to hope that the more information put into the problem, the better will be the bounds on an  $\mathbf{x}$ -vector of a given size. Accordingly, we compare  $20 \times 41$ ,  $40 \times 41$ , and  $60 \times 41$  examples of the problem with 3-point Simpson averaging and symmetric hump constraint. The  $20 \times 41$  and  $40 \times 41$  graphs are shown in Fig. 3.3. For the  $60 \times 41$  example, see Fig. 3.1. From this example, we see that the more information that can be used, the better the result will be. This is true only up to a point, however. If the integration is very crude ( $n$  small) compared to the amount of information ( $m$  large) the problem becomes inconsistent, due to discretization error. For pointwise estimates the  $40 \times 21$  and  $60 \times 21$  examples fail with the quantity  $\tau + \mu^2 - \rho_0$ , one of the factors of  $\mathcal{L}^2$ , becoming negative. There was a small discrepancy in all of the pointwise estimates in that the bounds for the first four points did not include the true values, which were all close to zero. We surmise that this was caused by the discretization error in approximating the continuous problem with the quadrature rule.

In all cases, the method was allowed to iterate three times, and in all of the cases, the bounds did not improve after the first iteration. However, the authors have encountered problems where bounds kept improving slightly for several iterations.

The method has been applied to several radiation spectrum unfolding problems using real, measured data. In all cases it has produced useful bounds for the unknown



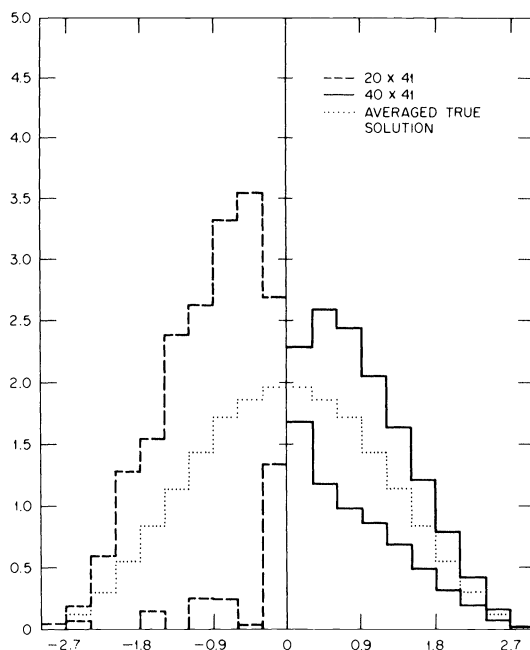


FIG. 3.3. 3-point Simpson averaging, symmetric hump constraint.

spectrum, and in some of these cases the bounds were so sharp that the effect of suboptimality was scarcely noticeable. In most cases, however, the intervals were noticeably wider than those obtained from the quadratic programming solutions of problems (1.10) and (1.11). In these cases the suboptimal intervals provide good starting estimates for the parametric quadratic programming procedure and significantly reduce the work required to obtain the optimal intervals.

**Acknowledgments.** The authors would like to thank Walter R. Burns, Michael T. Heath and Dianne P. O'Leary for their advice and suggestions. We would also like to thank several anonymous reviewers and referees for pointing out numerous corrections and improvements.

#### REFERENCES

- [1] RICHARD P. BRENT, *Algorithms for Minimization Without Derivatives*, Prentice-Hall, Englewood Cliffs, NJ, 1973.
- [2] WILLIAM G. BROWN AND BRUCE R. BEATTIE, *Improving estimates of economic parameters by use of ridge regression with production function applications*, *Amer. J. Agr. Econ.*, 57 (1975), pp. 21-32.
- [3] W. R. BURRUS, *Utilization of a priori information by means of mathematical programming in the statistical interpretation of measured distributions*, Tech. Rept. ORNL-3743, Oak Ridge National Laboratory, Oak Ridge, TN, June 1965.
- [4] W. R. BURRUS, B. W. RUST AND J. E. COPE, *Constrained interval estimation for linear models with ill-conditioned equations*, in *Information Linkage Between Applied Mathematics and Industry II*, Arthur L. Schoenstadt et al., eds., Academic Press, New York, 1980, pp. 1-38.
- [5] LARS ELDEN, *Algorithms for the regularization of ill-conditioned least squares problems*, *BIT*, 17 (1977), pp. 134-145.
- [6] M. T. HEATH, *The numerical solution of ill-conditioned systems of linear equations*, Tech. Rept. ORNL-4957, Oak Ridge National Laboratory, Oak Ridge, TN, July 1974.
- [7] ARTHUR E. HOERL AND ROBERT W. KENNARD, *Ridge regression: biased estimation for nonorthogonal problems*, *Technometrics*, 12 (1970), pp. 55-67.

- [8] ARTHUR E. HOERL AND ROBERT W. KENNARD, *Ridge regression: applications to nonorthogonal problems*, *Technometrics*, 12 (1970), pp. 69–82.
- [9] W. KAHAN, *Circumscribing an ellipsoid about the intersection of two ellipsoids*, *Canad. Math. Bull.*, 11 (1968), pp. 437–441.
- [10] DONALD W. MARQUARDT, *Generalized inverses, ridge regression, biased linear estimation, and nonlinear estimation*, *Technometrics*, 12 (1970), pp. 591–612.
- [11] DAVID L. PHILLIPS, *A technique for the numerical solutions of certain integral equations of the first kind*, *J. Assoc. Comput. Mach.*, 9 (1962), pp. 84–97.
- [12] B. W. RUST AND W. R. BURRUS, *Suboptimal methods for solving constrained estimation problems*, Defense Atomic Support Agency, 2604, January 1971.
- [13] ———, *Mathematical Programming and the Numerical Solution of Linear Equations*, American Elsevier, New York, 1972.
- [14] H. SCHEFFE, *The Analysis of Variance*, John Wiley, London, 1959.

## A SINGLE CODE FOR THE SOLUTION OF STIFF AND NONSTIFF ODE'S\*

G. HALL† AND M. B. SULEIMAN‡

**Abstract.** Many practical stiff problems contain a nonstiff subsystem. Automatic tests for partitioning a system of ODE's are described. Results of a code which uses Adams formulae on the nonstiff subsystem and backward differentiation formulae on the remaining equations are given.

**Key words.** ordinary differential equations, stiff systems, linear multistep methods, partitioning, type insensitive codes, stability

**1. Introduction.** We describe an algorithm for the automatic integration of a system of first order ordinary differential equations with initial conditions,

$$(1.1) \quad y' = f(x, y), \quad y(a) = c, \quad y, f \in \mathbb{R}^s.$$

The algorithm uses the well-known Adams and Backward Differentiation (BDF) multistep formulae.

Our first objective was to provide a single code which would be efficient for both stiff and nonstiff problems. The integration is started with a variable order Adams PECE method. If, during the integration, certain tests are satisfied the system is assumed to be stiff and the implicit BDF methods are introduced.

The implementation of BDF methods for stiff systems requires estimates of the Jacobian matrix  $J = (\partial f / \partial y)$  which are used in a Newton-type iteration for solving the implicit formulae. The second objective has been to reduce the degree of implicitness required by restricting the use of these methods to some subset (chosen automatically) of the system (1.1). The program attempts to separate out what we will refer to as the stiff subsystem of (1.1) and retain the Adams formulae for the remainder. In general therefore we require estimates of a subset of the full Jacobian matrix which, if feasible, can lead to computational savings.

If the problem (1.1) is stiff and the transient region can be correctly identified by our tests there will be further gains in efficiency in using the Adams code over the initial range. The program makes no attempt to revert to a full Adams code in case the transients may be reintroduced, which can happen on some practical problems.

For nonstiff problems our intention is that the algorithm will perform as a normal variable-order Adams code. It is therefore important that not too much time is wasted in testing for stiffness on such problems.

The primary motivation for this work was the paper by Krogh (1974). An initial code, which required partition of the system by the user was developed by Bentley (1975). The code discussed here is given in Suleiman (1979) where some further experimentation was done with a more general algorithm which treats higher order equations directly.

It has been recognised for some time that many practical "stiff problems" contain a nonstiff subsystem, which will usually vary in size over the integration range—initially the whole system may be so treated. Proposals to take advantage of this are contained in the work of Dahlquist (1968), Oden (1971), Krogh (1974), Hofer (1976), Robertson

---

\* Received by the editors February 24, 1983, and in revised form March 19, 1984.

† Department of Mathematics, The University of Manchester, Manchester M13 9PL, England.

‡ Jabatan Matematik, Universiti Pertanian Malaysia, Serdang, Selangor, Malaysia.

(1976), Enright and Kamel (1979), Söderlind (1980) and Shampine, most recently in Shampine (1980b), (1981). Our work is most closely related to that of Hofer and Söderlind; they are primarily concerned with stiff problems and used fixed order methods with the partitioning provided initially by the user. Robertson and Enright and Kamel exploit the situation by proposing the use of stiff methods but with a modified iteration strategy to take advantage of the nonstiff subsystem. The algorithm of Enright and Kamel is fully automatic. Shampine has the objective of providing an efficient code for both stiff and nonstiff problems, and proposes a technique for switching between Adams and BDF methods and vice versa for the system as a whole.

In § 2 we discuss some absolute stability properties of an algorithm consisting of a mixture of Adams PECE and BDF methods, see also Hofer (1976) and Söderlind (1979). In §§ 3 and 4 we describe the tests used in deciding to introduce the BDF methods and in selecting the stiff subsystem. These tests evolved partly from the practical observation of an Adams code on stiff problems. Finally in § 5 we present some numerical results using the code on some well-known test problems.

**2. Absolute stability.** For convenience we suppose that the first  $m$  equations of (1.1) are treated by the implicit BDF method and the remaining  $(s - m)$  equations by an Adams PECE method. Although we assume here that the implicit formula is solved exactly, the predictor for these components does affect the stability except in the case  $m = s$ . Some important theoretical results for the case where two different linear multistep methods are applied to each part of the partition are given in Söderlind (1979).

The appropriate model for absolute stability is the linear constant coefficient system  $y' = Ay$  written in the form,

$$(2.1) \quad \begin{pmatrix} u \\ v \end{pmatrix}' = \begin{pmatrix} U & B \\ C & V \end{pmatrix} \begin{pmatrix} u \\ v \end{pmatrix},$$

where  $u \in \mathbb{R}^m$ ,  $v \in \mathbb{R}^{s-m}$  and the matrix  $A$  is partitioned accordingly. The constant stepsize algorithm may be expressed in the form

$$(2.2) \quad \begin{aligned} \alpha_0^*(p_{n+1} - u_{n+1}) + \rho^*(E)u_{n+1-k} &= \phi h[Uu_n + Bv_n], \\ (q_{n+1} - v_{n+1}) + (v_{n+1} - v_n) &= h\sigma^*(E)[Cu_{n+1-k} + Vv_{n+1-k}], \\ \rho(E)u_{n+1-k} &= h[Uu_{n+1} + Bv_{n+1}], \\ v_{n+1} - v_n &= h\beta_0[C(p_{n+1} - u_{n+1}) + V(q_{n+1} - v_{n+1})] + h\sigma(E)[Cu_{n+1-k} + Vv_{n+1-k}]. \end{aligned}$$

(Note that the appearance of  $u_{n+1}$ ,  $v_{n+1}$  in the first two stages is illusory.)  $E$  is the shift operator;  $p_{n+1}$  and  $q_{n+1}$  are the predicted values for  $u_{n+1}$  and  $v_{n+1}$  respectively. We consider the cases  $\phi = 0$  or  $1$ , both choices corresponding to extrapolation for  $p_{n+1}$  from back values  $\{u_{n+1-i}\}_0^{k \text{ or } k+1}$  by the third equation of (2.2). Note that although nonstiff components are corrected first, which corresponds to our actual implementation, they appear last in (2.2) to correspond to the form of the stability polynomial below. The polynomials  $\rho^*(t) = \sum_{i=0}^k \alpha_i^* t^{k-i}$ ,  $\rho(t) = \sum_{i=0}^k \alpha_i t^{k-i}$  are associated with prediction and correction (BDF) of the first  $m$  components. Similarly  $\sigma^*(t) = \sum_{i=0}^k \beta_i^* t^{k-i}$  and  $\sigma(t) = \sum_{i=0}^k \beta_i t^{k-i}$  are associated with the Adams predictor and corrector respectively.

Eliminating  $p_{n+1} - v_{n+1}$  and  $q_{n+1} - v_{n+1}$  from the last equation in (2.2) the stability polynomial may be immediately written down, from the last two equations, as

$$\det \begin{pmatrix} \rho(t) - ht^k U & -ht^k B \\ -\left\{ h\sigma(t)C + h^2\beta_0\sigma^*(t)VC \right. \\ \left. - h\frac{\beta_0}{\alpha_0^*}\rho^*(t)C + \phi h^2\frac{\beta_0}{\alpha_0^*}t^{k-1}CU \right\} & L(t) - h^2\frac{\beta_0}{\alpha_0^*}\phi t^{k-1}CB \end{pmatrix} = 0$$

where  $L(t) = t^k - t^{k-1} - h\sigma(t)V + h\beta_0V(t^k - t^{k-1} - h\sigma^*(t)V)$ . By the obvious block row operation we may write this as  $\det(W) = 0$  where

$$(2.3) \quad W = \begin{pmatrix} \rho(t) - ht^k U & -ht^k B \\ -\left\{ h\sigma(t)C + h^2\beta_0\sigma^*(t)VC \right. \\ \left. - h\frac{\beta_0}{\alpha_0^*}\rho^*(t)C + \phi h\frac{\beta_0}{\alpha_0^*}\frac{\rho(t)}{t}C \right\} & L(t) \end{pmatrix}.$$

The determinant of the leading diagonal submatrix is the stability polynomial associated with using the given BDF method on the system  $u' = Uu$ . Similarly the other diagonal submatrix,  $L(t)$ , gives the stability polynomial of the Adams PECE method applied to  $v' = Vv$ . Sufficient conditions for the stepsize not to be severely restricted by stability are that  $B$  or  $C$  be zero and the system  $v' = Vv$  be nonstiff. Clearly such a splitting may also be advantageous provided  $B$  or  $C$  contain sufficiently small entries.

The following analysis of the case  $s = 1, m = 1$  shows that it is not, in general, necessary for  $B$  or  $C$  to be small. Let

$$(2.4) \quad A = \begin{pmatrix} u & b \\ c & v \end{pmatrix}, \quad u \ll v < 0.$$

Apply the one-step method

$$(2.5) \quad \rho^*(t) = t - 1 = \rho(t), \quad \sigma^*(t) = 1, \sigma(t) = \frac{1}{2}(t + 1).$$

The stability polynomial reduces to

$$(2.6) \quad \det \begin{pmatrix} t - 1 - hut & -hbt \\ -hc\left(1 + \frac{hv}{2}\right) - \phi\frac{hc}{2}\frac{t-1}{t} & t - 1 - hv - \frac{h^2v^2}{2} \end{pmatrix} = 0.$$

Necessary and sufficient conditions for  $at^2 + bt + c = 0, a > 0$ , to have zeros inside the unit circle are  $a + b + c > 0, a - c > 0, a - b + c > 0$ . Using this result on (2.6) gives the stability requirement,

$$(2.7) \quad \begin{aligned} (i) \quad & h^2(uv - bc)\left(1 + \frac{hv}{2}\right) > 0, \\ (ii) \quad & h^2(uv - \phi bc) - 2h(u + v)\left(1 + \frac{hv}{2}\right) > 0, \\ (iii) \quad & (2 - hu)\left(2 + hv + \frac{h^2v^2}{2}\right) + h^2bc\left(1 + \phi + \frac{hv}{2}\right) > 0. \end{aligned}$$

Conditions 2.7(i) and 2.7(ii) both hold for  $h \in (0, -2/v)$ , the usual absolute stability

restriction for the given Adams PECE method, given  $uv - bc > 0$ . For the case  $\phi = 0$  write (2.7)(iii) in the form

$$2\left(2 + hv + \frac{h^2 v^2}{2}\right) - 2hu(1 + hv)^2 + (3uv + bc)h^2\left(1 + \frac{hv}{2}\right) > 0.$$

This holds for  $h \in (0, -2/v)$  provided  $3uv + bc > 0$ . Therefore sufficient conditions for the stepsize not to be restricted severely are

$$(2.8) \quad -3uv < bc < uv.$$

In the case  $\phi = 1$  we write (2.7)(iii) in the form

$$2\left(2 + hv + \frac{h^2 v^2}{2}\right) - 2hu\left(1 + \frac{\sqrt{3}+1}{4}hv\right)^2 + \left(\frac{\sqrt{3}}{2}uv + bc\right)h^2\left(2 + \frac{hv}{2}\right) > 0,$$

and require, similarly

$$(2.9) \quad -\frac{\sqrt{3}}{2}uv < bc < uv.$$

The eigenvalues of (2.4) may be written as  $(u+v)(1-\theta)$  and  $(u+v)\theta$  where  $\theta^2 - \theta + d = 0$ ,  $d = (uv - bc)/(u+v)^2$ . Conditions (2.8)/(2.9) imply one large and one small eigenvalue on the negative real axis; we have  $d$  of order  $v/u$  and  $\theta \approx d$ . It is unrealistic to expect a separation to be advantageous if both eigenvalues were large in absolute value.

Note finally that the above discussion proves adequate for the following problem, Enright et al. (1975).

$$\begin{aligned} y_1' &= 400y_2 - 100y_1y_3 - 3000y_1^2, \\ y_2' &= -0.04y_2 + 0.01y_1y_3, \\ y_3' &= 30y_1^2, \end{aligned}$$

in the interval  $0 \leq x \leq 40$  with initial conditions  $y_1 = y_3 = 0$ ,  $y_2 = 1$ . Near  $x = 40$  one eigenvalue of the Jacobian is zero and the others are  $-0.17$  and  $-3352$  approximately, and the solution  $y_1 \approx 0.092$ ,  $y_2 \approx 0.72$ ,  $y_3 \approx 28$ . The Jacobian evaluated here is

$$\begin{pmatrix} u & b_1 & b_2 \\ c_1 & v_{11} & v_{12} \\ c_2 & 0 & 0 \end{pmatrix},$$

where  $u = -3352$ ,  $b_1 = 400$ ,  $b_2 = -92$ ,  $c_1 = 0.28$ ,  $c_2 = 5.52$ ,  $v_{11} = -10^{-4}b_1$ ,  $v_{12} = -10^{-4}b_2$ . Applying (2.5) with the BDF method for the first equation only ( $m = 1$ ) gives a stability polynomial with one root equal to 1 and the others determined by (2.6) where  $v = v_{11}$  and  $bc$  is replaced by  $b_1c_1 + b_2c_2$ . The stability is therefore determined by (2.7) with  $bc = 61.2$  and  $uv = 134.08$ . Since (2.8)/(2.9) hold, the range of stable stepsizes is  $h \in (0, -2/v) = (0, 50)$ .

**3. Testing for stiffness.** For all problems an Adams PECE code, permitting different order formulae for different component equations in (1.1) is used initially. We describe here the tests used in this phase to decide whether to introduce the BDF methods. If satisfied these are followed by a further test to select what we will treat as the stiff subsystem, using the information generated here. This further test may then be repeated at different points in the integration; it often happens that the stiff subsystem

will grow. This is natural, corresponding to transient regions associated with different time constants.

The order selection strategy is based on that given in Shampine and Gordon (1975) and the stepsize is increased only if a significant change appears valid. As the problem becomes stiff the code favors low orders and frequent step failures occur arising from nonsmooth behaviour of the local error estimates. In the following description an absolute local accuracy requirement, specified by TOL, is assumed. Denote by  $E_i$  the  $i$ th component of the local error estimate and by  $k_i$  the order of the formula used for this component. We take an error test failure at low order as a first indication that the problem could be stiff, namely

$$(3.1) \quad |E_i| > \text{TOL} \quad \text{and} \quad k_i \leq 5.$$

The tests we have developed for selecting the stiff subsystem, § 4, require knowledge of the  $i$ th row of the Jacobian. In principle this requires a Jacobian evaluation, i.e.,  $s$  function evaluations for nonsparse problems, each of which provides a column of the Jacobian. There is the possibility of writing the function routines so that particular component functions can be evaluated separately. In this case the Jacobian entries required for our tests cost the equivalent of 3 function evaluations, but more than three subroutine calls. This may therefore be a doubtful saving on some computer systems. Our main concern has been to produce a general purpose type-insensitive code and to explore the possibility of automatic selection of the stiff subsystem. We have not considered the detailed implications for very large practical problems which usually have sparse Jacobians and require special handling. In addition to the  $i$ th row we assume that a particular column and the diagonal of the Jacobian are available, although the latter could be dispensed with.

The second test required is that

$$(3.2) \quad \sum_{i=1}^s \frac{\partial f_i}{\partial y_i} < 0,$$

based on the fact that the trace is the sum of the eigenvalues.

The final tests are based on the regions of absolute stability of the Adams PECE methods at fixed stepsize and order for the system as a whole.

Let  $C_k$  denote the shortest distance from the origin to the boundary of the stability region in the left-half plane. For unstable stepsizes we expect  $h\sigma > C_{k_i}$  where  $\sigma = \min \{\|J\|_1, \|J\|_\infty\}$ , since  $\sigma$  is an upper bound for the maximum eigenvalue. To reduce the cost of the tests we approximate  $\sigma$  by

$$\min \sum_{j=1}^s \left| \frac{\partial f_i}{\partial y_j} \right|, \quad \sum_{j=1}^s \left| \frac{\partial f_j}{\partial y_p} \right|,$$

where

$$\left| \frac{\partial f_i}{\partial y_p} \right| = \max_j \left| \frac{\partial f_i}{\partial y_j} \right|.$$

The final tests are

$$(3.3) \quad h \sum_{j=1}^s \left| \frac{\partial f_i}{\partial y_j} \right| > C_{k_p},$$

and, if this holds,

$$(3.4) \quad h \sum_{j=1}^s \left| \frac{\partial f_j}{\partial y_p} \right| > C_{k_r}.$$

Although the framework for the final tests is one of constant stepsize/order, they have proved useful. It is generally observed that  $k_i = \min_r k_r$ , and that  $h$  is as large as if this order was used for the system as a whole. It appears likely that such an Adams code avoids combinations of formulae (orders) that restrict the stepsize unnecessarily, Hall and Suleiman (1980). Many stiff problems have large entries in particular columns of the Jacobian, Robertson (1976), or possibly rows depending on the scaling. In this situation (3.3)/(3.4) are a good indication that the error test failure is due to instability and that a change to BDF formulae is advisable.

It should be noted that one Jacobian approximation is computed when (3.1) occurs. This will occasionally arise on nonstiff problems, particularly at crude tolerances, but the gains to be made on stiff problems can far outweigh this inefficiency. One could replace the tests in this section by the strategy of Shampine (1980a). However we would still use (3.1) as the starting point for determining the stiff subsystem.

**4. Selecting the stiff subsystem.** We note first that if (3.1) holds it is not necessarily the case that the  $i$ th equation should be included in the stiff subsystem. Consider

$$y' = Ay, \quad A = \begin{pmatrix} u & 0 \\ c & v \end{pmatrix}, \quad u \ll v < 0, \quad |c| > |u - v|.$$

Write the computed solution in terms of the eigenvectors of  $A$ ,

$$(4.1) \quad y_n = \alpha \begin{pmatrix} (u - v)/c \\ 1 \end{pmatrix} + \beta \begin{pmatrix} 0 \\ 1 \end{pmatrix}.$$

The first vector in (4.1) corresponds to the eigenvalue  $u$ , and we assume  $|\alpha| < \text{TOL}$ ; the rapid transient has been integrated out, to the requested accuracy. It is the re-excitation of this term in forming the error estimate that leads to the stepsize failure. For example, using the Adams method (2.5) the local error estimate is

$$\frac{h^2}{2} A^2 y_n = \alpha \frac{h^2 u^2}{2} \begin{pmatrix} (u - v)/c \\ 1 \end{pmatrix} + \beta \frac{h^2 v^2}{2} \begin{pmatrix} 0 \\ 1 \end{pmatrix}.$$

An attempt to increase the stepsize after the transient phase will lead eventually to a stepsize failure and this occurs first on a test of the second component; from the discussion in § 2 it is the first equation that should be treated by a BDF formula.

The tests used are now described. In general they may be applied when a stiff subsystem has previously been selected. We assume first that the  $i$ th equation is not part of the stiff subsystem.

*Case I.*  $|E_i| > \text{TOL}$ ,  $k_i \leq 5$  and the  $i$ th equation currently nonstiff. The code uses the error estimate

$$(4.2) \quad E_i = g_{k_i-1,2} f_i^{(0)}[x_{n+1}, x_n, \dots, x_{n-k_i+1}],$$

which is the usual difference between two different order Adams correctors. Here  $g_{i,t}$  is the  $t$ -fold integral

$$g_{i,t} = \int_{x_n}^{x_{n+1}} \int_{x_n}^{x^{(t-1)}} \dots \int_{x_n}^{x^{(1)}} (x - x_n) \dots (x - x_{n-i+1}) dx dx^{(1)} dx^{(2)} \dots dx^{(t-1)}$$

and the divided difference is based on  $f_i(x_{n+1}, y_{n+1}^{(0)})$  where  $y_{n+1}^{(0)}$  is the vector of predicted values. The nonstiff components are corrected first and the corrector iteration has the form,

$$(4.3) \quad y_{i,n+1}^{(m+1)} = y_{i,n+1}^{(0)} + g_{k_i,1} f_i^{(m)}[x_{n+1}, x_n, \dots, x_{n-k_i+1}]$$



$m = 0, 1$ . For the differences between successive iterates we have, writing  $e_i^{(m+1)} = y_{i,n+1}^{(m+1)} - y_{i,n+1}^{(m)}$ , from (4.3),

$$(4.4) \quad e_i^{(1)} = g_{k_i,1} f_i^{(0)}[\dots],$$

and

$$\begin{aligned} e_i^{(2)} &= g_{k_i,1} (f_i^{(1)}[\dots] - f_i^{(0)}[\dots]) \\ &= \alpha_i (f_i(x_{n+1}, y_{n+1}^{(1)}) - f_i(x_{n+1}, y_{n+1}^{(0)})), \end{aligned}$$

where  $\alpha_i = g_{k_i,1} / g_{k_i,0}$ . Therefore we can write

$$(4.5) \quad e_i^{(2)} \approx \alpha_i \sum_{j=1}^s \theta_{ij}, \quad \text{where } \theta_{ij} = \frac{\partial f_i}{\partial y_j} e_j^{(1)}.$$

For the low order Adams formulas if the local error failure is due to instability the iterates will be at best slowly converging. If the stepsize is never allowed to more than double it can also be shown that  $g_{k,1} \cong g_{k-1,2}$ . Hence from (4.2) and (4.4)

$$|e_i^{(1)}| \cong |E_i| > \text{TOL},$$

and in the presence of instability we expect

$$(4.6) \quad |e_i^{(2)}| \cong \text{TOL}.$$

We use (4.5) and (4.6) to select the equations to be included in the stiff subsystem. Our original strategy was to change to BDF for any equation for which

$$(4.7) \quad k_j \leq 6 \quad \text{and} \quad R|\theta_{ij}|\alpha_i > \text{TOL}$$

where  $R$  is an empirical constant. From (4.5) and (4.6) this we regard as a further indication of instability, due to the coupling with equation  $j$ . However, it can happen that the corrector iteration is strongly divergent and several equations be unnecessarily included in the stiff subsystem. Therefore the following strategy is now used. Let  $|\theta_{ir}| = \max_j |\theta_{ij}|$  taken over all equations integrated by Adams methods of order at most 6. Then equation  $r$  is changed to stiff if

$$5|\theta_{ir}|\alpha_i > \text{TOL}.$$

We also make the change for all equations  $j$  for which

$$25|\theta_{ij}| > |\theta_{ir}|,$$

i.e. they must satisfy (4.7) but also lie roughly within an order of magnitude of the maximum contribution to (4.5). Note that  $e_i^{(2)}$  is not actually required.

*Case II.  $|E_i| > \text{TOL}$  and the  $i$ th equation in the stiff subsystem.* For most problems we tested equations were included in the stiff subsystem under Case I. It is, however, possible that the only indication that the stiff subsystem should be enlarged is under Case II arising from a coupling with an equation currently regarded as nonstiff.

For simplicity we assume that the first  $m$  equations comprise the current stiff subsystem and write the problem in the form,

$$\begin{aligned} y' &= f(x, y, z), & y, f &\in \mathbb{R}^m, \\ z' &= g(x, y, z), & z, g &\in \mathbb{R}^{s-m}. \end{aligned}$$

We also write  $e^{(1)} = y_{n+1}^{(1)} - y_{n+1}^{(0)}$ ,  $d^{(1)} = z_{n+1}^{(1)} - z_{n+1}^{(0)}$ , etc. Since the nonstiff components are corrected first, the iteration for the stiff subsystem has the general form

$$(4.8) \quad \begin{aligned} A e^{(1)} &= f(x_{n+1}, y_{n+1}^{(0)}, z_{n+1}^{(1)}) - y_{n+1}^{(0)'} \\ A e^{(2)} &= f(x_{n+1}, y_{n+1}^{(1)}, z_{n+1}^{(1)}) - y_{n+1}^{(1)'} \end{aligned}$$

where  $A = (\partial f / \partial y) - \text{diag}(\beta_i)$  and  $\beta_i$  is a function of the previous  $k_i$  stepsizes. In this way any change in  $z_{n+1}$  has been suppressed in (4.8).

Under Case II we are envisaging a situation where  $f$  is sensitive to changes in  $z_{n+1}$ . If the stiff components were to be corrected first we have found in practice that nonconvergence of the corrector iteration (4.8) is the usual indication that the stiff subsystem should be enlarged. Correcting the nonstiff components first allows us to formulate similar tests to Case I.

The error estimate  $E_i$  is a multiple of  $e_i^{(1)}$ . Large entries in the matrix  $(\partial f / \partial z)$  are necessary to produce sensitivity of the estimate to the nonstiff components. The tests require the  $i$ th row of  $(\partial f / \partial z)$ . Let  $\theta_{ij} = (\partial f_i / \partial z_j) d_j^{(1)}$  and  $|\theta_{ip}| = \max_j |\theta_{ij}|$  over all equations  $j$  with  $k_j \leq 6$ . Then if

$$5|\theta_{ip}| > \left| \frac{\partial f_i}{\partial y_i} e_i^{(1)} \right|$$

the  $p$ th equation is included in the stiff subsystem; we also make the change for any such equations for which

$$25|\theta_{ij}| > |\theta_{ip}|.$$

Since we do not require or examine all of  $(\partial f / \partial z)$  it is possible that the coupling giving rise to  $|E_i| > \text{TOL}$  may not be detected (this has not occurred in practice and we have failed to produce an artificial problem on which it was not detected). Nevertheless to cover this possibility, after 10 occurrences of Case II leading to no enlargement of the stiff subsystem the whole system is thereafter regarded as stiff.

Finally note that a change from Adams to BDF is never made immediately. We wait several steps until the required back information has been stored so that the BDF method does not start with order 1.

**5. Numerical results.** The code has been tested on a large number of problems given in the literature, both stiff and nonstiff. The following five problems give a clear indication of the potential in this approach.

### 5.1. Test problems.

*Problem 1. Nonstiff, Shampine and Gordon (1975).*

$$y_1'' = -y_1 / r^3, \quad y_1(0) = 1, \quad y_1'(0) = 0,$$

$$y_2'' = -y_2 / r^3, \quad y_2'(0) = 0, \quad y_2(0) = 1,$$

$$r = (y_1^2 + y_2^2)^{1/2}, \text{ written as a first order system,}$$

absolute error test,  $0 \leq x \leq 16\pi$ .

*Problem 2. Stiff, Enright et al. (1975).*

$$y_1' = -y_1 + y_2, \quad y_1(0) = 1,$$

$$y_2' = -10^2 y_1 - y_2, \quad y_2(0) = 0,$$

$$y_3' = -10^2 y_3 + y_4, \quad y_3(0) = 1,$$

$$y_4' = -10^4 y_3 - 10^2 y_2, \quad y_4(0) = 0,$$

$0 \leq x \leq 20$ .

*Eigenvalues*  $-1 \pm 10i, -100 \pm 100i$ . Mixed error test.

**Problem 3.** *Stiff, Enright et al. (1975).*

$$\begin{aligned}y_1' &= -y_1 + y_2^2 + y_3^2 + y_4^2, & y_1(0) &= 1, \\y_2 &= -10y_2 + 10(y_3^2 + y_4^2), & y_2(0) &= 1, \\y_3 &= -40y_3 + 40y_4^2, & y_3(0) &= 1, \\y_4 &= -100y_4 + 2, & y_4(0) &= 1, \\0 &\leq x \leq 20.\end{aligned}$$

*Eigenvalues*  $-1, -10, -40, -100$ . Mixed error test.

**Problem 4.** *Stiff, Enright et al. (1975).*

$$\begin{aligned}y_1' &= -0.04y_1 + 0.01y_2y_3, & y_1(0) &= 1, \\y_2' &= 400y_1 - 100y_2y_3 - 3000y_2^2, & y_2(0) &= 0, \\y_3' &= 30y_2^2, & y_3(0) &= 0.\end{aligned}$$

*Eigenvalues*  $0, 0 \rightarrow -3100, -0.04 \rightarrow -0.4 \rightarrow -0.03$ . Mixed error test.

**Problem 5.** *Ehle (1972), variation of Krogh (1973).*

$$y' = UBU^T y + Uw,$$

$$B = \begin{bmatrix} 10 & 100 & 0 & 0 \\ -100 & 10 & 0 & 0 \\ 0 & 0 & -100 & 0 \\ 0 & 0 & 0 & -0.1 \end{bmatrix}, \quad U = \begin{bmatrix} -\frac{1}{2} & \frac{1}{2} & \frac{1}{2} & \frac{1}{2} \\ \frac{1}{2} & -\frac{1}{2} & \frac{1}{2} & \frac{1}{2} \\ \frac{1}{2} & \frac{1}{2} & -\frac{1}{2} & \frac{1}{2} \\ \frac{1}{2} & \frac{1}{2} & \frac{1}{2} & \frac{1}{2} \end{bmatrix},$$

$$w = (\frac{1}{2}(z_1^2 - z_2^2), z_1 z_2, z_3, z_4)^T,$$

$$z = Uy, y(0)^T = (0, -2, -1, -1),$$

$$0 \leq x \leq 25.$$

*Eigenvalues* at  $x = 0, 8 \pm 100i, -2.1, -102,$   
at  $x = 25, -0.1, -10 \pm 100i, -100$ .

Mixed error test.

**5.2. Results.** The following five tables give the results for each problem. The headings are self explanatory except for

*Stiffness tests* the number of tests made in checking for stiffness.

*Time* time in seconds on CDC 7600 (Algol).

*Stiff subsystem* the notation  $(i, j, \dots)[a, b]$  means that equations  $i, j, \dots$  were simultaneously included in the stiff subsystem at  $x = a$  and where the stepsize was equal to  $b$ .

*Shampine* Adams code, results from Shampine and Gordon (1975).

Each table gives the results for the algorithm described above on one of the test problems. For the stiff problems it is of interest to compare the results with a comparable code using standard techniques. To obtain such comparisons each stiff problem was reintegrated by the same code, but this time treating the whole system as stiff from the beginning. Results for these runs are given in brackets below the principal results.

TABLE I  
 Problem 1 (nonstiff).

$\log_{10}$ TOL	Function evaluations	Number of steps	Failed steps	Maximum error	Stiffness tests	Time	Shampine method	
							Function evaluations	Maximum error
-2	464	232	14	1.6 (0)	14	5.84 (-1)	219	2.3 (0)
-3	260	130	6	1.7 (0)	6	3.37 (-1)	253	3.8 (-1)
-4	282	141	4	1.2 (-1)	0	3.58 (-1)	313	1.8 (-1)
-5	338	169	0	1.9 (-2)	0	4.42 (-1)	352	1.9 (-2)
-6	424	212	1	5.5 (-5)	0	5.72 (-1)	494	1.6 (-4)
-7	506	253	0	1.9 (-4)	0	6.77 (-1)	547	1.7 (-4)
-8	600	300	0	1.4 (-5)	0	8.19 (-1)	633	9.4 (-6)
-9	712	356	0	1.8 (-7)	0	9.79 (-1)	974	6.6 (-8)
-10	854	427	0	4.6 (-8)	0	1.23 (0)	899	2.5 (-8)

**5.3. Comments on the results.** On problem 1, nonstiff, there were 14 instances of (3.1) at  $TOL = 10^{-2}$  and 6 at  $10^{-3}$ , each time the problem being diagnosed as nonstiff. Except at  $10^{-2}$ , however, the code compares reasonably well with a good Adams code.

Problems 2 and 3 illustrate the general improvement in efficiency obtained over the same code using a standard approach. Both problems were eventually treated as full stiff problems although for most of the integration range the implicit BDF formulae (and consequent algebraic manipulations) were required only for a subsystem. As anticipated the entry of an equation into the stiff subsystem is delayed with the move to more stringent tolerances, but the stepsize at which this occurs appears to be independent of TOL. The figures for the standard approach show an unnecessary amount of matrix work ( $LU$  factorizations on a full  $s \times s$  matrix) required after stepsize failures, mostly in the transient phase, where the Adams methods are clearly superior. A new Jacobian is formed only after repeated failures (except in linear problems where it is needed only once).

Problem 4 was discussed at the end of § 2 (with the first two components interchanged for convenience). The stiff subsystem was correctly chosen as just one equation confirming the previous analysis. Particularly striking here is the smooth behaviour of the statistics with TOL, much better than in the standard approach (compare the entries in the maximum error column). This suggests that correct partitioning may lead to more reliable software.

On problem 5 the whole system became stiff, at  $x \approx 1$  (approx.), for all values of TOL. Nevertheless the use of the Adams code up to this point gives a considerable improvement in the cost, and also avoided the failure at  $TOL = 10^{-3}$  in the standard approach, due to getting on the wrong solution.

Although it is not possible to give a rigorous theoretical justification for the strategies employed in the code the above selected results clearly indicate the potential in this approach.

**Acknowledgments.** We are grateful for the referees' comments which have led to several improvements in presentation.

TABLE 2  
Problem 2.

$\log_{10}$ TOL	Function evaluations	Number of steps	Failed steps	LU factorizations/ Jacobian evaluations	Maximum error	Time	Stiffness tests	Stiff subsystem
-2	450 (485)	235 (254)	39 (55)	34/2 (55/1)	1.6 (-1) (2.6 (-1))	7.4 (-1) (8.1 (-1))	2	(3, 4)[1.3 (-1), 1.3 (-2)] (1, 2)[2.8 (0), 9.9 (-2)]
-3	418 (561)	216 (287)	25 (54)	16/2 (54/1)	3.3 (-2) (2.2 (-1))	7.0 (-1) (9.8 (-1))	2	(3, 4)[1.6 (-1), 1.6 (-2)] (1, 2)[4.6 (0), 8.8 (-2)]
-4	585 (798)	303 (407)	31 (31)	22/2 (22/1)	3.7 (-3) (4.5 (-3))	9.8 (-1) (1.4 (-1))	2	(3, 4)[2.2 (-1), 1.1 (-2)] (1, 2)[6.1 (0), 9.8 (-2)]
-5	727 (1125)	371 (574)	19 (82)	13/2 (82/1)		1.2 (2.0)	2	(3, 4)[1.8 (-1), 1.4 (-2)] (1, 2)[6.1 (0), 9.8 (-2)]
-6	970 (1580)	495 (800)	27 (101)	13/2 (101/1)		1.7 (2.7)	2	(3, 4)[1.9 (-1), 1.0 (-2)] (1, 2)[1.2 (1), 9.4 (-2)]
-7	1208 (2319)	616 (1170)	34 (119)	20/2 (119/2)		2.1 (4.0)	2	(3, 4)[2.5 (-1), 1.6 (-2)] (1, 2)[1.4 (1), 6.9 (-2)]
-8	1467 (3332)	741 (1675)	18 (138)	6/2 (138/2)		2.6 (5.7)	2	(3, 4)[3.7 (-1), 1.4 (-2)] (1, 2)[1.5 (1), 7.5 (-2)]
-9	1747 (4725)	880 (2370)	15 (136)	4/3 (4/1)		3.1 (8.1)	3	(3, 4)[2.7 (-1), 1.1 (-2)] (2)[1.90 (0), 1.1 (-1)] (1)[1.94 (0), 1.1 (-1)]
-10	2093 (6707)	1052 (3221)	14 (139)	4/2 (139/1)		3.8 ((1.1 (1))	2	(3, 4)[2.3 (-1), 5.2 (-3)] (1, 2)[2.0 (1), 7.2 (-2)]

TABLE 3  
Problem 3.

$\log_{10}$ TOL	Function evaluations	Number of steps	Failed steps	$LU$ factorizations/ Jacobian evaluations	Maximum error	Time	Stiffness tests	Stiff subsystem
-2	104 (122)	55 (62)	6 (12)	1/7 (12/2)	2.7 (-2) (1.5 (-2))	1.8 (-1) (2.1 (-1))	4	(4)[1.1 (-1), 2.3 (-2)](2)[1.4 (0), 3.4 (-1)] (3)[4.3 (-1), 4.0 (-2)](1)[1.6 (1), 4.3 (0)]
-3	130 (161)	68 (82)	6 (13)	2/7 (13/2)	3.0 (-3) (1.3 (-3))	2.2 (-1) (2.8 (-1))	4	(4)[1.4 (-1), 2.2 (-2)](2)[2.0 (0), 4.5 (-1)] (3)[8.0 (-1), 8.5 (-2)](1)[1.4 (1), 3.0 (0)]
-4	169 (221)	87 (112)	5 (16)	1/7 (16/1)	2.6 (-4) (2.6 (-4))	2.8 (-1) (3.9 (-1))	4	(4)[3.9 (-1), 3.8 (-2)], (2)[2.1 (0), 3.2 (-1)] (3)[8.0 (-1), 8.5 (-2)], (1)[1.7 (1), 2.6 (0)]
-5	220 (295)	113 (149)	6 (22)	1/7 (22/1)	2.2 (-5) (2.7 (-5))	3.4 (-1) (5.2 (-1))	4	(4)[3.8 (-1), 3.8 (-2)], (2)[3.7 (0), 3.6 (-1)] (3)[1.3 (0), 1.1 (-1)], (1)[1.7 (1), 2.0 (0)]
-6	293 (376)	151 (190)	10 (22)	2/5 (22/1)	8.3 (-6) (4.6 (-6))	4.4 (-1) (6.7 (-1))	3	(4)[6.8 (-1), 2.5 (-2)], (2)[2.9 (0), 2.3 (-1)] (3)[1.3 (0), 7.4 (-2)]
-7	358 (518)	184 (261)	10 (23)	1/6 (23/1)	6.2 (-7) (4.9 (-7))	5.5 (-1) (9.3 (-1))	3	(4)[8.3 (-1), 1.5 (-2)], (2)[6.9 (0), 3.8 (-1)] (3)[1.9 (0), 1.4 (-1)]
-8	390 (720)	199 (362)	10 (25)	4/5 (25/1)	3.3 (-8) (8.6 (-8))	6.4 (-1) (1.3)	3	(4)[8.6 (-1), 4.1 (-2)], (2)[6.3 (0), 4.1 (-1)] (3)[1.6 (0), 8.7 (-2)]
-9	534 (1000)	275 (502)	16 (27)	2/6 (27/1)	1.7 (-8) (8.4 (-9))	8.7 (-1) (1.7)	3	(4)[1.3 (0), 2.2 (-2)], (2)[1.3 (1), 3.9 (-1)] (3)[2.8 (0), 1.2 (-1)]
-10	557 (1413)	283 (708)	10 (28)	3/2 (28/1)	7.7 (-10)	8.8 (-1) (2.4)	2	(2, 3)[2.7 (0), 1.5 (-1)]

TABLE 4  
Problem 4.

$\log_{10}$ TOL	Function evaluations	Number of steps	Failed steps	LU Factorization/ Jacobian evaluations	Maximum error	Time	Stiffness tests	Stiff subsystem
-2	78 (133)	41 (69)	7 (25)	4/4 (25/3)	3.3 (-2) (2.4 (-1))	9.4 (-2) (1.8 (-1))	2	(2)[4.4 (-3), 1.1 (-3)]
-3	109 (154)	56 (79)	4 (23)	2/2 (23/4)	5.3 (-3) (4.8 (-3))	1.4 (-1) (2.2 (-1))	1	(2)[1.6 (-3), 4.9 (-4)]
-4	158 (215)	82 (110)	9 (31)	4/3 (31/4)	5.2 (-4) (3.4 (-4))	1.9 (-1) (3.1 (-1))	1	(2)[1.3 (-2), 9.5 (-4)]
-5	183 (384)	93 (194)	3 (58)	0/2 (57/4)	4.9 (-5) (2.3 (-3))	2.2 (-1) (5.4 (-1))	2	(2)[1.1 (-2), 2.7 (-3)]
-6	224 (340)	114 (173)	9 (34)	6/3 (34/3)	4.9 (-6) (9.2 (-6))	3.0 (-1) (5.0 (-1))	1	(2)[8.3 (-2), 8.7 (-4)]
-7	287 (447)	146 (228)	5 (43)	0/2 (43/3)	4.2 (-7) (1.5 (-6))	4.0 (-1) (6.6 (-1))	1	(2)[1.3 (-2), 1.6 (-3)]
-8	389 (625)	197 (318)	9 (51)	4/3 (51/4)	6.7 (-8) (1.5 (-7))	5.4 (-1) (9.2 (-1))	1	(2)[1.3 (-2), 1.2 (-3)]
-9	500 (807)	253 (406)	10 (45)	5/3 (45/3)	6.0 (-9) (1.9 (-8))	7.1 (-1) (1.2)	2	(2)[1.2 (-2), 9.9 (-4)]
-10	648 (1118)	326 (565)	8 (51)	4.3 (51/3)	6.9 (-10) (2.1 (-9))	9.2 (-1) (1.6)	1	(2)[1.0 (-2), 4.6 (-4)]

TABLE 5  
Problem 5.

$\log_{10}$ TOL	Function evaluations	Number of steps	Failed steps	LU factorizations/ Jacobian evaluations	Maximum error	Number of tests	Stiff subsystem
-3	865 (FAIL)	436	53	19/5	1.9 (-2)	31	(1, 2, 3, 4)[1.5 (0), 1.1 (-2)]
-6	1768 (3113)	889 (1608)	29 (153)	14/2 (153/9)	2.0 (-5) (4.5 (-5))	1	(1, 2)[7.4 (-1), 8.1 (-3)] (3, 4)[8.2 (-1), 7.5 (-3)]
-8	2780 (6416)	1395 (3271)	26 (164)	14/1 (164/2)	9.7 (-8) (9.9 (-7))	1	(1, 2, 3, 4)[2.1 (0), 8.4 (-3)]

## REFERENCES

- [1] J. BENTLEY, (1975), M.Sc. thesis, Univ. Manchester, Manchester, England
- [2] J. A. I. CRAIGIE, (1975), *A variable order multistep method for the numerical solution of stiff systems of ordinary differential equations*, NA Report No. 11, Dept. Mathematics, Univ. Manchester, Manchester, England
- [3] G. DAHLQUIST, (1968), *A numerical method for some ODE's with large Lipschitz constants*, Information Processing 68, North-Holland, Amsterdam.
- [4] B. L. EHLE, (1972), *A comparison of numerical methods for solving certain stiff ordinary differential equations*, Technical Report No. 70, Dept. Mathematics, Univ. Victoria, British Columbia.
- [5] N. H. ENRIGHT, T. E. HULL AND B. LINDBERG, (1975), *Comparing numerical methods for stiff systems of ODE's*, BIT, 15, pp. 10-48.
- [6] W. H. ENRIGHT AND M. S. KAMEL, *Automatic partitioning of stiff systems and exploiting the resulting structure*, ACM Trans. Math. Software, 5, pp. 374-385.
- [7] E. HOFER, (1976), *Partially implicit method for large stiff systems of ODE's with only a few equations introducing small time constants*, SIAM J. Numer. Anal., 13, pp. 645-663.
- [8] F. T. KROGH, (1973), *On testing a subroutine for the numerical integration of ordinary differential equations*, J. Assoc. Comput. Mach., 20, pp. 545-562.
- [9] ———, (1974), *Changing stepsize in the integration of differential equations using modified divided differences*, Lecture Notes in Mathematics 362, Springer-Verlag, Berlin, pp. 22-77.
- [10] L. ODEN, (1971), *An experimental and theoretical analysis of the SAPS method for stiff ordinary differential equations*, Rep. NA 71.28, Dept. Information Processing, Royal Institute of Technology, Stockholm.
- [11] H. H. ROBERTSON, (1976), *Numerical integration of systems of stiff ordinary differential equations with special structure*, J. Inst. Math. Appl., 18, pp. 249-263.
- [12] L. F. SHAMPINE AND M. K. GORDON, (1975), *Computer Solution of Ordinary Differential Equations*, W. H. Freeman, San Francisco.
- [13] L. F. SHAMPINE, (1980a), *Lipschitz constants and robust ODE codes*, Computational Methods in Nonlinear Mechanics, North-Holland, Amsterdam, pp. 427-449.
- [14] ———, (1980b), *Type insensitive ODE codes based on implicit A-stable formulas*, SAND 79-2444, Sandia National Laboratories, Albuquerque, NM.
- [15] ———, (1981), *Type-insensitive ODE codes based on implicit A(a)-stable formulas*, SAND 81-0707J, Sandia National Laboratories, Albuquerque, NM.
- [16] G. SÖDERLIND, (1979), *Some stability properties of linear multistep compound discretizations of partitioned differential systems*, TRITNA-NA-7910, Royal Institute of Technology, Stockholm.
- [17] ———, (1980), *DASP3—A program for the numerical integration of partitioned stiff ODE and differential-algebraic systems*, TRITA-NA-8008, The Royal Inst. of Technology, Stockholm.
- [18] M. B. SULEIMAN, (1979), *Generalised multistep Adams and backward differentiation methods for the solution of stiff and nonstiff ODE's*, Ph.D. thesis, Univ. Manchester, Manchester, England.



## PARALLEL NETWORKS FOR MULTI-GRID ALGORITHMS: ARCHITECTURE AND COMPLEXITY\*

TONY F. CHAN† AND ROBERT SCHREIBER‡

**Abstract.** We describe and analyze a family of highly parallel special purpose computing networks that implement multi-grid algorithms for solving elliptic difference equations. The networks have many of the features and advantages of systolic arrays. We consider the speedup achieved by these designs and how this is affected by the choice of algorithm parameters and the level of parallelism employed. We find, for example, that when there is one processor per grid-point, the designs cannot avoid suffering a loss of efficiency as the grid-size tends to zero.

**Key words.** multi-grid algorithms, parallel architecture, computational complexity, speedup and efficiency, elliptic difference equations

**1. Introduction.** We shall describe and analyze a family of highly parallel special-purpose computing networks that implement multi-grid algorithms for solving elliptic difference equations. These networks have the same characteristics—regularity, local communication, and repetitive use of a single, simple processing element—that make systolic architectures attractive [9]. These architectural advantages make it possible to build large computing networks of VLSI cells that would be relatively cheap, reliable, and very powerful.

Both a basic and a full multi-grid algorithm are considered. The basic method reduces the error in a given initial approximation by a constant factor in one iteration. The full method requires no initial guess and produces a solution with error proportional to the truncation error of the discretization. These algorithms are representative of many variants of linear and nonlinear multi-grid algorithms.

The analysis assumes that we are solving a linear system originating in a discretization of an elliptic partial differential equation on a rectangle in  $R^d$ , using a regular  $n^d$  point grid. The network is a system of grids of processing elements. For each  $1 \leq k \leq K$ , processor grid  $P_k$  has  $(n_k)^\gamma$  elements, where  $\gamma$  is an integer less than or equal to  $d$ , and  $n_k = n$ . The machine implements a class of multi-grid algorithms using a corresponding system of nested point grids. For each  $1 \leq k \leq K$ , point grid  $G_k$  has  $(n_k)^d$  points. The key assumption, which is quite realistic, is that it takes this machine  $O((n_k)^{d-\gamma})$  time to carry out the computation required by one step of the multi-grid algorithm on point grid  $G_k$  using processor grid  $P_k$ .

We shall consider the efficiency of these parallel implementations, defining efficiency to be the ratio of the speedup achieved to the number of processors employed [8]. We shall consider a design to be efficient if this ratio remains bounded by a positive constant from below as  $n \rightarrow \infty$ . The analysis will show that when  $\gamma < d$  some algorithms can be efficiently implemented. But when  $\gamma = d$  (this is the most parallelism one can reasonably attempt to use) no algorithm can be efficiently implemented. There does exist, in this case, one group of algorithms for which the efficiency falls off only as  $(\log n)^{-1}$ .

---

\* Received by the editors August 9, 1983, and in revised form April 3, 1984.

† Computer Science Department, Yale University, New Haven, Connecticut 06520. The work of this author was supported in part by the U.S. Department of Energy under grant DE-AC02-81ER10996 and was completed while this author was a visitor at INRIA, Le Chesnay, France in 1983.

‡ Computer Science Department, Stanford University, Stanford, California 94305. The work of this author was supported in part by the Office of Naval Research under contract N00014-82-K-0703.

The analysis assumes that we implement the same algorithms used by uniprocessor systems. Convergence results for these algorithms have been rather well developed recently [1], [5], [7]. We make no attempt to develop algorithms that exhibit concurrent operation on several grids. Note, however, that some encouraging experimental results with such an algorithm have been obtained recently by Gannon and Van Rosendale [6].

In any discussion of the practical use of a specialized computing device, it must be acknowledged that overspecialization can easily make a design useless. At least, the designed device should be able to solve a range of size of problems of a particular structure, perhaps solving large problems by making several passes over the data, solving a sequence of smaller subproblems, or with some other techniques. We shall consider how a large grid, with  $(mn)^d$  points can be handled by a system of processor grids with  $n^\gamma$  elements each having  $O(m^d n^{d-\gamma})$  memory cells. Problems on nonrectangular domains can be handled by techniques requiring repeated solutions on either rectangular subdomains or containing domains.

Brandt [3] has also considered parallel implementations of multi-grid methods. He discusses the use of various interconnection networks and appropriate smoothing iterations. One of our results, a  $(\log n)^2$  time bound for fully parallel, full multi-grid algorithms, is also stated in his paper.

**2. Multi-grid algorithms.** We shall consider multi-grid algorithms in a general setting. The continuous problem is defined by the triple  $\{H, a(u, v), f(v)\}$ , where  $H$  is a Hilbert space with a norm  $\|\cdot\|$ ,  $a(u, v)$  is a continuous symmetric bilinear form on  $H \times H$ , and  $f(v): H \rightarrow R$  is a continuous linear functional. The problem is:

$$(1) \quad \text{Find } u \in H \text{ such that } a(u, v) = f(v) \text{ for all } v \in H.$$

It can be shown that if  $a(*, *)$  satisfies certain regularity conditions (for example, that  $a(v, v) \geq c_0 \|v\|^2$  for all  $v \in H$ ), then problem (1) has a unique solution [4].

We consider finite-dimensional approximations of problem (1). Let  $M_j, j \geq 1$ , be a sequence of  $N_j$ -dimensional spaces, on which one can define a corresponding bilinear form  $a_j(u, v)$  and a corresponding continuous linear functional  $f_j(v)$ , which are constructed to be approximations to  $a(u, v)$  and  $f(v)$  respectively. Also, since the multi-grid algorithms involve transferring functions between these spaces, we have to construct extension (interpolatory) operators  $E_j: M_{j-1} \rightarrow M_j$ .

We shall give two multi-grid algorithms, namely BASICMG and FULLMG, with FULLMG calling BASICMG in its inner loop. The two algorithms differ in that BASICMG starts its computation on the finest grid and works its way down to the coarser grids, whereas FULLMG starts with the coarsest grid and works its way up to the finest grid. In the conventional single processor case, BASICMG reduces the error on a certain grid by a *constant factor* in optimal time, whereas FULLMG reduces the error to *truncation error level* in optimal time.

We give the basic multi-grid algorithm BASICMG in Table 2.1. This is a recursive algorithm, although in practice it is usually implemented in a nonrecursive fashion. The iterations are controlled by the predetermined parameters  $(c, j, m)$ . In this sense it is a direct method, unlike related adaptive algorithms which control the iterations by examining relative changes in the residuals [2]. Figure 2.1 illustrates the iteration sequence in the case  $c = 2$ . This particular case is sometimes known as the *W-cycle*, so-called because of the shape of the diagram in Fig. 2.1. Analogously, the case  $c = 1$  is also known as the *V-cycle*. The major computational work is in the smoothing sweeps (subroutine SMOOTH), which usually consists of some implementation of the successive-over-relaxation or Jacobi iteration or the conjugate gradient method. The

TABLE 2.1

**Algorithm BASICMG** ( $k, z, c, j, m, a_k, f_k$ )  
 (Computes an approximation to  $u_k \in M_k$ ,  
 where  $a_k(u_k, v) = f_k(v)$  for all  $v \in M_k$ ,  
 given an initial guess  $z \in M_k$ .  
 Returns the approximate solution in  $z$ .  
 Reduces initial error in  $z$  by a constant factor.)

**If**  $k = 1$  **then**  
 Solve the problem using a direct method. Return solution  $z$ .  
**else**  
 (Smoothing step ( $j$  sweeps):)  
 $z \leftarrow \text{SMOOTH}(j, z, a_k, f_k)$ .  
 (Compute data  $b_{k-1}$  for coarse grid correction equation:)  
 $b_{k-1} = a_{k-1}(q, v_i)$  for all  $v_i \in M_{k-1}$ .  
 (Solve coarse grid problem approximately by  $c$  cycles of BASICMG:)  
 $q \leftarrow 0$ .  
 Repeat  $c$  times:  
     BASICMG ( $k-1, q, c, j, m, a_{k-1}, b_{k-1}$ )  
 (Correction step:)  
 $z \leftarrow z + E_k q$ .  
 (Smoothing step ( $m$  sweeps):)  
 $z \leftarrow \text{SMOOTH}(m, z, a_k, f_k)$ .  
**End If**  
**End BASICMG**

smoothing sweeps are used to annihilate the highly oscillatory (compared to the grid spacing) components of the error in  $z$  efficiently. We require that a suitably “parallel” method, Jacobi or odd/even SOR for example, be used as the smoother. In the next section, we shall discuss new architectures for implementing these smoothing operations in more efficient ways.

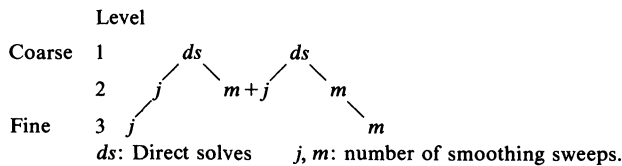


FIG. 2.1. Iterations of BASICMG for  $k=3$  and  $c=2$ .

We give the full multi-grid algorithm FULLMG in Table 2.2. In the BASICMG algorithm, the choice of initial guess for  $u_k$  is not specified. In practice, good initial guesses are sometimes available essentially free (for example, from solutions of a nearby problem, from solutions at a previous time step, etc.). The FULLMG algorithm interpolates approximate solutions on coarser grids as initial guesses for the BASICMG algorithm. It is also recursive and nonadaptive. Figure 2.2 illustrates the iteration sequence in the case  $k=3, c=2$  and  $r=1$ .

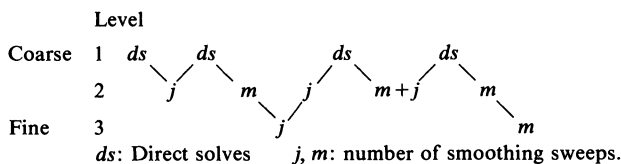


FIG. 2.2. Iteration of FULLMG for  $k=3, c=2$  and  $r=1$ .

TABLE 2.2

**Algorithm FULLMG** ( $k, z_k, r, c, j, m, a_k, f_k$ )  
 (Computes an approximation  $z_k$  to  $u_k \in M_k$   
 where  $a_k(u_k, v) = f_k$  for all  $v \in M_k$ ,  
 using  $r$  iterations of BASICMG,  
 using initial guess from interpolating the approximate  
 solution obtained on the next coarser grid.  
 Solution obtained can be proven to have truncation error accuracy.)  
**If**  $k = 1$  **then**  
   Solve the problem using a direct method to get  $z_1$ .  
**else**  
   (Obtain solution on next coarser grid:)  
     FULLMG ( $k-1, z_{k-1}, r, c, j, m, a_{k-1}, f_{k-1}$ ).  
   (Interpolate  $z_{k-1}$ :)  
      $z_k \leftarrow E_k z_{k-1}$ .  
   (Reduce the error by iterating BASICMG  $r$  times:)  
     Repeat  $r$  times:  
       BASICMG ( $k, z_k, c, j, m, a_k, f_k$ ).  
**End if**  
**End FULLMG**

We would like to summarize briefly the accuracy and convergence behavior of the above two multi-grid algorithms. Since the main emphasis of this paper is on the algorithmic aspects of these multi-grid algorithms, we shall refer the reader to the literature for more details. The framework presented here is based on the work of Bank and Dupont [1] and Douglas [5]. The accuracy and the convergence of the BASICMG algorithm obviously depend on the three crucial steps of the algorithm: smoothing, coarse grid transfer, and fine grid correction. The basic requirements are that the smoothing sweeps annihilate the high frequency components of the error efficiently, the coarse grid correction  $q$  be a good approximation to the fine grid error in the low frequency components, and the interpolation operators ( $E_j$ 's) be accurate enough. These conditions can be formalized into mathematically precise hypotheses which can then be verified for specific applications [5]. Assuming these hypotheses, one can show that Algorithm BASICMG reduces the error on level  $k$  by a constant factor provided that enough smoothing sweeps are performed. Moreover, it can be shown (see § 4) that Algorithm BASICMG (for small values of  $c$ ) can achieve this in optimal time, i.e.  $O(N_k)$  arithmetic operations. Obviously, the work needed depends on the accuracy of the initial guess and increases with the level of accuracy desired. Often, one is satisfied with *truncation error accuracy*, i.e.  $\|z - u\| = O(\|u_k - u\|) \cong C N_k^{-\theta}$  for some fixed  $\theta$  and  $C$  which are independent of  $k$ . For a general initial guess, the straightforward application of Algorithm BASICMG to reduce the initial error to this level takes  $O(N_k \log(N_k))$  time, which is not optimal. The FULLMG algorithm overcomes this problem by using accurate initial guesses obtained by interpolating solutions from coarser grids. The convergence result for Algorithm BASICMG can be combined with the basic approximation properties of the various finite-dimensional approximations  $\{M_j, a_j, f_j\}$  to show that Algorithm FULLMG computes a solution  $z_k$  that has truncation error accuracy in  $O(N_k)$  time.

**3. The computing network.** In this section we describe a simple parallel machine design for multi-grid iteration. We restrict attention to linear elliptic problems in  $d$  dimensions over rectangular domains, to discretizations based on grids of  $n^d$  points, and to multi-grid methods based on a system of point grids  $\{G_k\}_{k=1}^K$  where  $G_k$  has

$(n_k)^d$  gridpoints, with mesh lengths  $h_{k,j}$ ,  $1 \leq j \leq d$ , the finest grid has  $n_K = n$ , and

$$n_{k+1} = a(n_k + 1) - 1, \quad k = 1, 2, \dots, K - 1$$

for some integer  $a \geq 2$ .

The machine consists of a system of processor-grids  $\{P_k\}_{k=1}^K$  corresponding to the point grids. Each processor-grid is an  $(n_k)^\gamma$  lattice in which a processor is connected to its  $2\gamma$  nearest neighbors.

We shall employ a standard multi-index notation for gridpoints and processors. Let

$$\mathbf{z}_n^+ \equiv \{0, 1, \dots, n - 1\}.$$

Let  $\mathbf{z}_{n,s}^+ \equiv (\mathbf{z}_n^+)^s$ , the set of  $s$ -tuples of nonnegative integers less than  $n$ . We shall make use of a projection operator  $\pi_r^s: \mathbf{z}_{n,r}^+ \rightarrow \mathbf{z}_{n,s}^+$  defined for  $r \geq s$  by

$$\pi_r^s((i_1, \dots, i_r)) = (i_1, \dots, i_s).$$

By convention, if  $\mathbf{i} \in \mathbf{z}_{n,s}^+$ , then  $\mathbf{i} = (i_1, \dots, i_s)$ . Also let  $\mathbf{1} = (1, \dots, 1)$ . We shall also use the norm  $|\mathbf{i}| \equiv |i_1| + \dots + |i_s|$  on  $\mathbf{z}_{n,s}^+$ .

We shall label the gridpoints in  $G_k$  with elements of  $\mathbf{z}_{n_k,d}^+$  in such a way that the point with label  $\mathbf{i}$  has spatial coordinates  $(i_1 h_{k,1}, i_2 h_{k,2}, \dots, i_d h_{k,d})$ . Similarly, we label processors in  $P_k$  with indices in  $\mathbf{z}_{n_k,\gamma}^+$ .

Thus, processors  $\mathbf{i}$  and  $\mathbf{k}$  are connected if  $|\mathbf{i} - \mathbf{k}| = 1$ . In order to make the machine useful for problems with periodic boundary conditions, we might also add "wrap-around" connections, so that  $\mathbf{i}$  and  $\mathbf{k}$  are connected if  $|(\mathbf{i} - \mathbf{k}) \bmod n| = 1$ . In § 3.1, it is shown that periodic problems can also be handled without these connections.

Evidently, if each processor has  $O(n^{d-\gamma})$  memory cells, we can store the solution, forcing function, and  $O(1)$  temporary values belonging to the whole of grid  $G_k$  in the processors of  $P_k$ ; we store gridpoint  $\mathbf{i}$  in processor  $\pi_d^\gamma(\mathbf{i})$  for  $\mathbf{i} \in \mathbf{z}_{n,d}^+$ .

With the given connectivity, smoothing sweeps of some types can be accomplished in  $O(n^{d-\gamma})$  time. It is not necessary for the stencil of the difference scheme to correspond to the connectivity of the processor grid. Jacobi or odd/even SOR smoothing can be so implemented, for example. Let  $t$  be the time taken by a single processor to perform the operations at a single gridpoint that, done over the whole grid, constitute a smoothing sweep. If  $S$  is the time to implement a smoothing sweep over the whole of grid  $G_k$  on processor grid  $P_k$ , then

$$(2) \quad S = tn_k^{d-\gamma}.$$

Grid  $P_k$  is connected to grid  $P_{k+1}$ . Processor  $\mathbf{i} \in P_k$  is connected to processor  $a(\mathbf{i} + \mathbf{1}) - \mathbf{1} \in P_{k+1}$ . These connections allow the inter-grid operations (forming coarse grid forcing terms  $b_k$  and interpolation  $E_k$ ) to also be computed in  $O(S)$  time. We refer to the system of processor-grids  $\{P_1, \dots, P_J\}$  as the machine  $L_J$  for  $J = 1, 2, \dots, K$ .

The execution of the BASICMG iteration by  $L_k$  proceeds as follows.

1. First,  $j$  smoothing steps on grid  $G_k$  are done by  $P_k$ . All other processor grids are idle.
2. The coarse grid equation is formed by  $P_k$  and transferred to  $P_{k+1}$ .
3. The  $c$  cycles of BASICMG on grid  $k-1$  are performed by  $L_{k-1}$ .  $P_k$  is idle.
4. The solution  $q$  is transferred to  $P_k$  and interpolation  $E_k q$  is performed by  $P_k$ .
5. The remaining  $m$  smoothing steps are done by  $P_k$ .

Let  $W(n)$  represent the time needed for steps 1, 2, 4 and 5. Then

$$(3) \quad W(n) = (j + m + s)tn^{d-\gamma}$$

where  $s$  is the ratio of the time required to perform steps 2 and 4 to the time needed for one smoothing sweep. Note that  $s$  is independent of  $n$ ,  $d$  and  $\gamma$ . Note that only one processor grid is active at any instant.

The natural way to build such a machine is to embed the  $\gamma = 1$  machine in two dimensions as a system of communicating rows of processors, the  $\gamma = 2$  machine in three dimensions as a system of communicating planes, etc. Of course, realizations in three-space are possible for any value of  $\gamma$ . For example, if  $d = 2$  and  $\gamma = 2$ , we have a machine that consists of  $K$  connected 2-dimensional grids of processors. Suppose  $a = 2$ ,  $K = 3$  and  $n_1 = 1$ ,  $n_2 = 2(1 + 1) - 1 = 3$ ,  $n_3 = 2(3 + 1) - 1 = 7$ . The corresponding machine is shown in Fig. 3.1. Gannon and Van Rosendale [6] consider a similar implementation of the fully parallel machine ( $\gamma = d$ ) on proposed VLSI and multi-microprocessor system architectures.

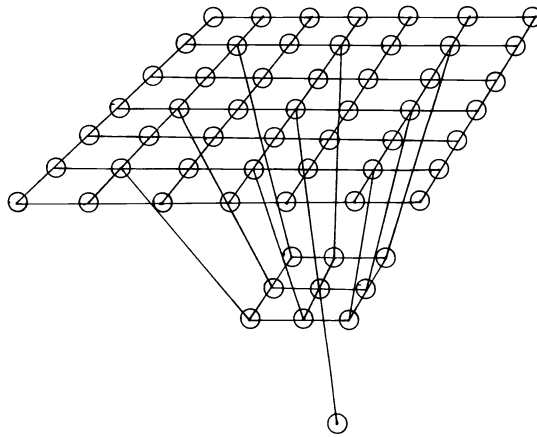


FIG. 3.1. A machine for  $d = 2$ ,  $\gamma = 2$  and  $K = 3$ .

This design differs from systolic array designs in that there is no layout with all wire lengths equal. But for reasonably large machines the differences in wire length should not be so great as to cause real difficulties. Moreover, one need not continue to use ever coarser grids until a  $1 \times 1$  grid is reached. In practice, 3 or 4 levels of grids could be used and most of the multi-grid efficiency retained; this would make the construction much simpler.

**3.1. Solving larger problems.** Suppose there are  $(mn)^d$  gridpoints and only  $n^\gamma$  processors. Assume that each processor can store all information associated with  $m^d n^{d-\gamma}$  gridpoints. Now we map gridpoints to processors in such a way that neighboring gridpoints reside in neighboring processors. To do this we define a mapping  $f_m : \mathbf{z}_{mn}^+ \rightarrow \mathbf{z}_n^+$ , such that, for all  $i, j \in \mathbf{z}_{mn}^+$ ,  $|f_m(i) - f_m(j)| \leq |i - j|$ , as follows. Let  $j = qn + r$  where  $q$  and  $r$  are integers,  $0 \leq r \leq n - 1$ . Now let

$$f_m(j) \in \begin{cases} r & \text{if } q \text{ is even,} \\ n - 1 - r & \text{if } q \text{ is odd.} \end{cases}$$

Now if  $m$  is even, then  $f_m(0) = f_m(mn - 1) = 0$ , so that periodic boundary conditions can be handled without any “wrap-around” connections. This operation corresponds to folding a piece of paper in a fan-like manner; for  $m = 10$ , for example, like this:



To map a multidimensional structure we fold it as above in each coordinate. Let the processor grid have  $n^d$  elements and the point grid have  $m_1 n \times m_2 n \times \dots \times m_d n$  points. Point  $\mathbf{i}$  can be stored in processor  $F_d(m_1, \dots, m_d, n; \mathbf{i})$  where  $F_d(\mathbf{i}) \equiv (f_{m_1}(\mathbf{i}_1), \dots, f_{m_d}(\mathbf{i}_d))$ . If we have only  $n^\gamma$  processors then we map  $\mathbf{i}$  into  $F_d^\gamma(\mathbf{i})$  where  $F_d^\gamma(\mathbf{i}) \equiv F_\gamma(\pi_d^\gamma(\mathbf{i}))$ .

With this mapping, neighboring gridpoints reside in neighboring processors, and therefore relaxation and interpolation processes only require communication with neighboring processors. Consequently,

$$W(mn) = m^d W(n).$$

Thus with a factor of  $m^d$  fewer processors, the time for these processes increases by a factor of  $m^d$ . Therefore, there is no loss of efficiency. It is important to note, and simple to show, that this mapping also preserves locality throughout the hierarchy of point grids  $\{G_j\}$  and processor grids  $\{P_j\}$ .

The mapping is illustrated in Fig. 3.2 for the case  $m = 2, n = 2$  and  $\gamma = d = 2$ .

Processor grid	Processor assignment to grid points
1 2	1 2 2 1
3 4	3 4 4 3
	3 4 4 3
	1 2 2 1

FIG. 3.2. Point-grid to processor-grid mapping.

**4. Complexity.** In this section, we are going to analyze the time complexity of the two multi-grid algorithms, BASICMG and FULLMG, as implemented by the different architectures just discussed. It turns out that the complexity of the two algorithms is very similar. Since the BASICMG algorithm is simpler and is called by FULLMG, we shall discuss and analyze it first. After that, we shall indicate how to derive the results for FULLMG.

**4.1. Complexity of BASICMG.** To simplify the analysis, we shall assume that the computational domain is a rectangular parallelepiped and is discretized by a hierarchy of cartesian grids (corresponding to the  $M_j$ 's) each with  $n_j$  mesh points on each side (denoted the  $n_j$ -grid). Further, we assume that the  $n_j$ 's satisfy  $n_j = a(n_{j-1} + 1) - 1$  where  $a$  is an integer bigger than one. Generally, we denote by  $T(n)$  the time complexity of the BASICMG algorithm on an  $n$ -grid. By inspecting the description of algorithm BASICMG, it is not difficult to see that  $T(n)$  satisfies the following recurrence:

$$(4) \quad T(an) = cT(n) + W(an),$$

where  $W(an)$  denotes the work needed to preprocess and postprocess the  $(an)$ -grid iterate before and after transfer to the coarser  $n$ -grid. We have the following general result concerning the solution of (4), the proof of which is elementary.

LEMMA 1. Let  $T_p$  be a particular solution of (4), i.e.

$$(5) \quad T_p(an) = cT_p(n) + W(an).$$

Then the general solution of (4) is:

$$(6) \quad T(n) = \alpha n^{\log_a c} + T_p(n), \text{ where } \alpha \text{ is an arbitrary constant.}$$

The term  $W(an)$  includes the smoothing sweeps, the computation of the coarse grid correction equation (i.e. the right-hand side  $b_{k-1}$ ) and the interpolation back to the fine grid ( $E_k q$ ). The actual time needed depends on the architecture used to

implement these operations (specifically the dimensionality of the domain and the number of processors available on an  $n$ -grid). In general, as derived in § 3,  $W(n)$  is given by

$$(7) \quad W(n) = (j + m + s)tg(n) \equiv \beta g(n), \quad \text{where } g(n) = n^p \text{ with } p \equiv d - \gamma.$$

In Table 4.1, we give the form of the function  $g(n)$  as a function of the architecture and the dimensionality of the domain. We also give a bound on the total number of processors ( $P$ ) needed to implement the architecture and note that it is always the same order as the number of processors on the finest grid. For a  $d$ -dimensional problem with  $n^\gamma$  processors on the  $n$ -grid, we have

$$P(\gamma) = \begin{cases} 1 & \text{if } \gamma = 0, \\ (a^\gamma / (a^\gamma - 1))n^\gamma & \text{if } \gamma > 0. \end{cases}$$

TABLE 4.1  
Table of  $g(n)$

Architecture	1-D	2-D	3-D	Total # of processors on all grids, $P$
1 processor	$n$	$n^2$	$n^3$	1
$n$ processors	1	$n$	$n^2$	$(a/(a-1))n$
$n^2$ processors	—	1	$n$	$(a^2/(a^2-1))n^2$
$n^3$ processors	—	—	1	$(a^3/(a^3-1))n^3$

Note: Architecture column gives number of processors on the  $n$ -grid.

We have the following general result for this class of functions  $g(n)$ .

LEMMA 2. If  $W(n) = \beta n^p$ , then we can take the following as particular solution of (4):

$$(8) \quad T_p(n) = \begin{cases} \beta(a^p / (a^p - c))n^p & \text{if } p \neq \log_a c, \\ \beta n^p \log_a n & \text{if } p = \log_a c. \end{cases}$$

Combining the results of the last two lemmas, we arrive at our main result.

THEOREM 3. The solution of (4) for  $W(n) = \beta n^p$  satisfies:

$$(9) \quad T(n) = \begin{cases} \beta(a^p / (a^p - c))n^p + O(n^{\log_a c}) & \text{if } c < a^p, \\ \beta n^p \log_a n + O(n^p) & \text{if } c = a^p, \\ O(n^{\log_a c}) & \text{if } c > a^p. \end{cases}$$

Note that in the last case,  $\alpha \neq 0$  (in (6)) because  $T_p(n)$  does not satisfy the boundary conditions.

For the first two cases ( $c \leq a^p$ ), we can determine the highest order term of  $T(n)$  explicitly. However, for the case  $c > a^p$ , the constant in the highest order term depends on the initial condition of the recurrence (4) (i.e. the time taken by the direct solve on the coarsest grid), which is more difficult to measure in the same units as that of the smoothing and interpolation operations. Fortunately, the complexity for this case is nonoptimal and thus not recommended for use in practice and therefore, for our purpose, it is not necessary to determine this constant.

Based on the results in Theorem 3 and the specific forms of the function  $g(n)$  in Table 4.1, we can compute the time complexity of Algorithm BASICMG for various



combinations of  $c$ ,  $a$  and  $p$ , some of which are summarized in Table 4.2, where we tabulated the highest order terms of  $T(n)/\beta$ .

The classical one processor ( $\gamma=0$ ) optimal time complexity results [1], [5] are contained in these tables. For example, in two dimensions ( $d=2$ )  $g(n) = n^2$  and Table 4.2 shows that, for the refinement parameter  $a=2$ ,  $c < 4$  gives an optimal algorithm ( $O(n^2)$ ) whereas  $c \geq 4$  is nonoptimal. *More generally, we have an optimal scheme if and only if  $c < a^d$ .* For example, the larger  $a$  is or the larger  $d$  is, the larger the value  $c$  can take for the algorithm to remain optimal. However, with a larger value of  $a$ , more relaxation sweeps are needed and a larger value of  $c$  has to be taken in order to achieve the same accuracy. We note that the constant in the highest order term of

TABLE 4.2  
Time complexity  $T(n)/\beta$  of algorithm BASICMG.

		$p = d - \gamma$			
		3	2	1	0
$s = 2$	$c$				
	1	$8/7n^3$	$4/3n^2$	$2n$	$\log_2 n$
	2	$8/6n^3$	$4/2n^2$	$n \log_2 n$	$O(n)$
	3	$8/5n^3$	$4n^2$	$O(n^{\log_2 3})$	$O(n^{\log_2 3})$
	4	$8/4n^3$	$n^2 \log_2 n$	$O(n^2)$	$O(n^2)$

		$p = d - \gamma$			
		3	2	1	0
$s = 3$	$c$				
	1	$27/26n^3$	$9/8n^2$	$3/2n$	$\log_3 n$
	2	$27/25n^3$	$9/7n^2$	$3n$	$O(n^{\log_3 2})$
	3	$27/24n^3$	$9/6n^2$	$n \log_3 n$	$O(n)$
	4	$27/23n^3$	$9/5n^2$	$O(n^{\log_3 4})$	$O(n^{\log_3 4})$

		$p = d - \gamma$			
		3	2	1	0
$s = 4$	$c$				
	1	$64/63n^3$	$16/15n^2$	$4/3n$	$\log_4 n$
	2	$64/62n^3$	$16/14n^2$	$4/2n$	$O(n^{\log_4 2})$
	3	$64/61n^3$	$16/13n^2$	$4n$	$O(n^{\log_4 3})$
	4	$64/60n^3$	$16/12n^2$	$n \log_4 n$	$O(n)$

- $a$ : mesh refinement ratio
- $c$ : number of correction cycles in BASICMG
- $\gamma$ :  $n^\gamma$  processors on the  $n$ -grid
- $d$ : dimension of the domain

— : asymptotic efficiency boundary (See Theorem 5)

$T(n)$  does not vary a great deal with either  $c$  or  $a$  (they are all about one, especially for the larger values of  $a$ ). This suggests that the best balance between speed and accuracy can be achieved by choosing the largest value of  $c$  (or close to it) such that the algorithm remains optimal. When  $d - \gamma = 2$  and  $a = 2$ , this means taking  $c$  to be 2 or 3.

**4.2. Efficiency, speedup, accuracy, and optimal design.** Next, we are going to look at the effects of the new architectures on the performance of the BASICMG algorithm. There are four parameters in this study:  $c$ ,  $a$ ,  $\gamma$  and  $d$ . We shall call a particular combination of these four parameters a *design*. We shall use the notation  $T(c, a, \gamma, d)$  to denote the corresponding complexity of the design. One of the main issues that we would like to address is the *efficiency*  $E$  and *speedup*  $S$  of a particular design, which are defined as [8].

DEFINITION 4.

$$S(c, a, \gamma, d) = T(c, a, 0, d) / T(c, a, \gamma, d),$$

$$E(c, a, \gamma, d) = T(c, a, 0, d) / (P(\gamma)T(c, a, \gamma, d)).$$

The speedup  $S$  measures the gain in speed over the one processor architecture while the efficiency  $E$  reflects the tradeoff between processors and time and measures the efficiency with which the architecture exploits the extra processors to achieve the speedup. The optimal efficiency is unity, in which case a  $P$ -fold increase in the number of processors reduces the time complexity  $P$ -fold. In general, the efficiency  $E$  and the speedup  $S$  are functions of  $n$ . We shall call a design *asymptotically efficient* if  $E$  tends to a constant as  $n$  tends to infinity and *asymptotically inefficient* if it tends to zero. We shall primarily be concerned with analyzing the efficiency  $E$  of a design in this section. The speedup  $S$  can be easily read from Table 4.2.

For determining the asymptotic efficiency of a design, it suffices to determine the highest order term of  $E$ . The efficiency  $E$  can be derived from the explicit expressions for  $T$  and  $P$  in a straightforward manner. Since the efficiency for  $\gamma = 0$  is unity by definition, we shall only be interested in  $\gamma > 0$ . We summarize the results in the following theorem.

THEOREM 5. Assume  $\gamma > 0$ .

- (1) If  $c < a^{d-\gamma}$  then  $E(c, a, \gamma, d) = (a^\gamma - 1)(a^{d-\gamma} - c) / (a^d - c)$ .
- (2) If  $c = a^{d-\gamma}$  then  $E(c, a, \gamma, d) = (a^\gamma - 1)a^{d-\gamma} / ((a^d - c) \log_a n)$ .
- (3) If  $c > a^{d-\gamma}$  then

$$E(c, a, \gamma, d) = \begin{cases} O(1/n^{\log_a c - d + \gamma}) & \text{if } c < a^d, \\ O(\log_a n / n^\gamma) & \text{if } c = a^d, \\ O(1/n^\gamma) & \text{if } c > a^d. \end{cases}$$

Based on the above theorem, we can immediately make the following observations:

1. A design is asymptotically efficient if and only if  $c < a^{d-\gamma}$ . This inequality defines an *efficiency boundary* in the four parameter space of  $\{c, a, \gamma, d\}$ , the projections of which are shown in Table 4.2.

2. The fully parallel design ( $\gamma = d$ ) is always asymptotically inefficient. This follows because to have an efficient design in this case requires  $c < 1$  which is meaningless for the multi-grid algorithm.

3. Define a logarithmically asymptotically efficient design to be one with  $E = O(1/\log_a n)$  as  $n$  tends to infinity. A fully parallel design is logarithmically asymptotically efficient if and only if  $c = 1$ . This is case (2) in Theorem 5.

4. *If we start with a nonoptimal design in the one processor case, then adding more processors will not make the design asymptotically efficient.* This corresponds to the last two cases in Case (3) of Theorem 5. The reason is that too many coarse grid correction cycles are performed so that even if more processors are added to speed up the setup time for transferring to the coarser grids, too much time is spent on the coarser grids.

Asymptotically efficient designs are theoretically appealing. They indicate that the extra processors are utilized efficiently to achieve the speedup. For this reason, it is interesting to consider the following problem:

*Optimal design problem.*

For a given problem (i.e. given  $d$ ), find the design that minimizes  $T(n)$  and/or maximizes the accuracy of the computed solution subjected to the constraint that it is asymptotically efficient.

In Table 4.3, we indicate the influence of each of the three design parameters  $\{c, a, \gamma\}$  on the optimality conditions and the constraint. For example, the accuracy of the solution is independent of  $\gamma$  and to achieve maximum accuracy, one should take  $c$  large and  $a$  small. The appropriate choice of optimality condition depends on the requirements of the given problem. Moreover, the general optimal design problem may not have a unique or bounded solution in the three parameter space  $\{c, a, \gamma\}$ . In practice, however, we usually do not have the freedom to choose all three parameters. If the number of free parameters is restricted, then the optimal design problem may have a unique solution.

TABLE 4.3  
*Influence of design parameters on optimality conditions.*

		Design parameters		
		$c$	$a$	$\gamma$
Optimality conditions	Max accuracy	large	small	indep
	Min $T(n)$	small	large	large
Constraint	Efficiency $E$	small	large	small

We shall illustrate this by fixing two of the three parameters in turn and study  $E$  as a function of the free parameter. First, let us fix  $c$  and  $a$  and consider the effect of varying  $\gamma$ . In other words, we consider the case where the multi-grid algorithm and the refinement of the domain are fixed and we are free to choose the architecture. Varying  $\gamma$  corresponds to moving across a particular row of Table 4.2. It is easy to see that one achieves a speedup as we use more processors (i.e. as one moves from left to right in one of these rows). However, the efficiency  $E$  generally goes down as one uses more processors, and after a certain entry the design starts to be asymptotically inefficient. For example, take the three-dimensional case ( $d = 3$ ), with  $a = 2$  and  $c = 2$ . With  $n$  processors on the  $n$ -grid, the efficiency is  $E(2, 2, 1, 3) = 1/3$ . With  $n^2$  processors on  $n$ -grid, we have  $E(2, 2, 2, 3) = 1/\log_2 n$ , and with  $n^3$  processors  $E(2, 2, 3, 3) = O(1/n)$ . Both the last two designs are asymptotically inefficient and thus the  $\gamma = 1$  design is the fastest efficient design. *In general, for fixed  $c$  and  $a$ , the design just to the left of the efficiency boundary is the fastest efficient design.*

Next we shall fix  $a$  and  $\gamma$  and vary  $c$  (columns in Table 4.2). That is, we fix the architecture and vary the multi-grid algorithm. This time the speedup factor  $S$  decreases

slightly as we increase  $c$  which is not surprising as we are doing more work on coarse grids, and this keeps many processors idle. Again, the efficiency  $E$  decreases as we increase  $c$ , and after a certain entry, the algorithm becomes asymptotically inefficient. For example, take the two-dimensional case with  $n$  processors on the  $n$ -grid ( $d = 2$ ,  $\gamma = 1$ ) and  $a = 3$ . Going down the appropriate column, we have  $E(1, 3, 1, 2) = \frac{1}{2}$ ,  $E(2, 3, 1, 2) = \frac{2}{7}$  and  $E(3, 3, 1, 2) = 1/\log_3 n$ . Recall that the larger  $c$  is, the more accurate is the computed solution and the more robust is the overall algorithm. Thus, the  $c = 2$  design is the most accurate efficient design. *In general, for fixed  $a$  and  $\gamma$ , the design just above the efficiency boundary is the most accurate efficient design. If accuracy is no problem, then  $c$  can be chosen smaller to speed up the algorithm.*

Finally, we fix  $c$  and  $\gamma$  and vary  $a$ . Generally, a larger value of  $a$  means fewer processors are needed to implement the architecture. It also means that less work has to be done on the coarse grids because they have fewer points. To see the effect of varying  $a$ , note that the efficiency boundary moves towards the lower right hand corner of the tables in Table 4.2 as  $a$  is increased. This implies that, for a fix architecture ( $\gamma$ ) and algorithm ( $c$ ), using a larger value of  $a$  will generally exploit the available processors more efficiently. However, one cannot indiscriminantly use large values for  $a$  because this leads to larger interpolation errors and less accurate solutions. For example, take the two-dimensional case with  $n$ -processors on the  $n$ -grid and  $c = 2$ . With  $a = 2$ , the algorithm is asymptotically inefficient ( $E(2, 2, 1, 2) = 1/\log_2 n$ ) whereas with  $a = 3$  and  $a = 4$ , it is asymptotically efficient ( $E(2, 3, 1, 2) = 2/7$ ,  $E(2, 4, 1, 2) = 3/7$ ). Thus, the  $a = 3$  design is the more accurate efficient design. *In general, for fixed  $c$  and  $\gamma$ , the smallest value of  $a$  that yields an efficient design is the most accurate efficient design. If accuracy is no problem, then  $a$  can be chosen larger to speed up the algorithm.*

One can carry out similar parametric studies, for example, by fixing one parameter and varying the other two. For instance, if we are free to choose both the architecture ( $\gamma$ ) and the algorithm ( $c$ ), then by inspecting the form of the efficiency constraint  $c < a^{d-\gamma}$ , it can be seen that smaller values of  $c$  allow more processors to be used (larger  $\gamma$ ) to produce a faster efficient design. For example, in the  $a = 2$  case, if  $c = 2$  then the  $p = 2$  entry gives the fastest efficient design whereas if  $c = 1$ , it becomes the  $p = 1$  entry, with the latter being faster. Similarly, for a fixed  $c$ , a larger value of  $a$  allows more processors to be used to achieve a faster efficient design.

**4.3. Complexity of FULLMG.** In this section, we shall derive the complexity of the FULLMG algorithm. Since FULLMG calls BASICMG, the results here depend crucially on the complexity of Algorithm BASICMG.

Let  $F(n)$  denote the time taken by one call to FULLMG. By inspecting the algorithm in Table 2.2, it can easily be verified that  $F(n)$  satisfies the following recurrence:

$$(10) \quad F(an) = F(n) + rT(an),$$

where we have absorbed the cost of the interpolation step in FULLMG into the interpolation costs of BASICMG (i.e. the term  $s$  in (7)). Note that this is just a special case of the recurrence (4), with  $c = 1$  and  $W(an) = rT(an)$ . By inspecting the entries in Table 4.2, we see that the forcing function  $W(n)$  in this case takes the form of either  $n^p$  or  $n^p \log_a n$ . We have the following result for particular solutions of (10) for this class of forcing functions, which can be verified by direct substitution.

LEMMA 6.

(1) *If  $T(n) = \alpha n^p$  then a particular solution of (10) is*

$$F_p(n) = (a^p / (a^p - 1)) \alpha n^p.$$

(2) If  $T(n) = \alpha n^p \log_a n$ , with  $p > 0$ , then a particular solution of (10) is

$$F_p(n) = (a^p/(a^p - 1))r\alpha n^p \log_a n - (a^p/(a^p - 1)^2)r\alpha n^p.$$

(3) If  $T(n) = \alpha \log_a n$  then a particular solution of (10) is

$$F_p(n) = (r\alpha/2)(\log_a^2 n + \log_a n).$$

Since the homogeneous solution of (10) is a constant, the general solution is dominated by the particular solutions. We give the highest order terms of  $F(n)$  in terms of  $T(n)$  in the following theorem.

THEOREM 7.

(1) If  $T(n) = \alpha n^p$  then  $F(n) = (a^p/(a^p - 1))rT(n) + O(1)$ .

(2) If  $T(n) = \alpha n^p \log_a n$ , with  $p > 0$ , then  $F(n) = (a^p/(a^p - 1))rT(n) + O(n^p)$ .

(3) If  $T(n) = \alpha \log_a n$  then  $F(n) = (r/2) \log_a n T(n) + O(\log_a n)$ .

In the first two cases, the complexity of  $F(n)$  is the same as that of  $T(n)$ , except for the *constant* multiplicative factor  $(a^p/(a^p - 1))r$ . The  $(a^p/(a^p - 1))$  part of this constant is very close to unity for the values of  $a$  and  $p$  that occur. The  $r$  part of the constant applies equally to all entries of Table 4.2 and reflects the number of times BASICMG is called by FULLMG. We point out that the choice of  $r$  that results in truncation error level accuracy depends on how efficiently BASICMG reduces its initial error but can be chosen *independent of  $n$*  [5]. Thus, the extra multiplicative factor does not affect the asymptotic efficiency of a particular entry in Table 4.2. The complexity of  $F(n)$  in the last case is actually increased by a factor of  $\log_a n$  over that of  $T(n)$ . However, the corresponding entries in Table 4.2 are already asymptotically inefficient and thus this extra factor again does not affect the asymptotic efficiency of the design. It follows that the discussions concerning the efficiency, speedup and optimal design for the BASICMG algorithm in § 4.1 are also valid for the FULLMG algorithm, with the exception that a fully parallel logarithmically asymptotically efficient design is slower by the factor  $\log_a n$ .

**5. Conclusion.** We have proposed an architecture based on a system of processor grids for parallel execution of multi-grid methods based on a system of point grids. We have analyzed its efficiency and shown that a combination of algorithm and machine is asymptotically efficient if and only if  $c < a^{d-\gamma}$ , where

- $c$  is the number of coarse grid iterations per fine grid iteration,
- $a$  is the mesh refinement factor,
- $d$  is the dimension of the point grids,
- $\gamma$  is the dimension of the processor grids.

We find therefore that fully parallel designs—with  $\gamma = d$ —cannot be asymptotically efficient. There is, however, only a logarithmic fall-off in efficiency when  $c = a^{d-\gamma}$ , and for fully parallel designs this occurs for  $c = 1$ .

The notion of efficiency plays a central role in this paper and provides guidelines in choosing algorithms and architectures. However, we emphasize that very often in practice the real issue is performance (or speedup) and an inefficient design may still be the best choice.

#### REFERENCES

- [1] R. E. BANK AND T. DUPONT, *An optimal order process for solving elliptic finite element equations*, Math. Comp., 36 (1981), pp. 35–51.
- [2] A. BRANDT, *Multi-level adaptive solutions to boundary-value problems*, Math. Comp., 31 (1977), pp. 333–390.

- [3] A. BRANDT, *Multi-grid solvers on parallel computers*, Technical Report 80-23, ICASE, NASA Langley Research Center, Hampton, VA, 1980.
- [4] P. G. CIARLET, *The Finite Element Method for Elliptic Problems*, North-Holland, Amsterdam, 1978.
- [5] C. DOUGLAS, *Multi-grid algorithms for elliptic boundary-value problems*, Ph.D. thesis, Dept. Computer Science, Yale Univ., New Haven, CT, 1982, also available as Technical Report 223.
- [6] D. GANNON AND J. VAN ROSENDALE, *Highly parallel multi-grid solvers for elliptic PDEs: an experimental analysis*, Technical Report 82-36, ICASE, NASA Langley Research Center, Hampton, VA, 1982.
- [7] W. HACKBUSCH, *A multi-grid method applied to a boundary value problem with variable coefficients in a rectangle*, Technical Report 77-17, Mathematisches Institut, Universitat zu Koln, Cologne, 1977.
- [8] DAVID J. KUCK, *The Structure of Computers and Computations*, John Wiley, New York, 1978.
- [9] H. T. KUNG, *Why systolic architectures?* IEEE Trans. Comput., 15 (1982), pp. 37-46.

## PRACTICAL THREE-DIMENSIONAL MESH GENERATION USING TRANSFINITE INTERPOLATION\*

LARS-ERIK ERIKSSON†

**Abstract.** A computational procedure for generating three-dimensional nonorthogonal surface-fitted mesh systems is presented. The method is based on the concept of transfinite interpolation and makes use of normal derivatives of the mapping function at the boundaries to obtain the desired mesh control. A brief description of the theory is presented together with 2D examples to demonstrate the general principles. The generation of 3D meshes of  $O-O$  type around wings and schematic wing-fuselage configurations is described in detail and several computed examples are shown.

**Key words.** numerical grid generation, body-fitted grids, nonorthogonal grids, curvi-linear coordinate systems, mesh singularities, transfinite interpolation, blending function methods, algebraic grid generation

**1. Introduction.** In the past decade the numerical generation of curvilinear coordinate systems has provided the key to the development of finite difference solutions of partial differential equations on regions with arbitrarily shaped boundaries. Although much of the impetus for these developments has come from fluid dynamics, the techniques are equally applicable to heat transfer, electromagnetics, structures, and all other areas involving field solutions.

With coordinate systems generated to maintain coordinate lines or surfaces coincident with the boundaries, finite difference schemes can be devised which are applicable to general configurations and have very simple and yet accurate boundary conditions. Furthermore, the coordinate systems can easily be adapted to the solution to provide a near optimum resolution for a given number of grid points. These advantages of general boundary-fitted coordinate systems are well-known and will not be elaborated upon in this paper.

Considerable progress has been made in the development of numerically generated coordinate systems, and a variety of generating methods have been presented in the literature. A survey of these methods will not be presented here, since several reviews of the latest developments have already been published [1], [2]. However, it is easily observed that all methods can be classified as belonging to either of two fundamentally different groups; direct algebraic methods or methods that generate coordinates by solving partial differential equations. The first group, direct algebraic methods, can subsequently be split into two subgroups; interpolation methods and other direct methods. Since the work presented in this paper belongs entirely to the interpolation category, we may stop the classification process at this point and concentrate on interpolation methods.

The idea of using interpolation as a means of constructing general curvilinear coordinate systems or meshes stems from the fact that in most cases, the coordinates or mesh points are known on several or on all of the boundaries of the computational domain and the problem consists of extending this mesh into the interior of the domain. Since the computational domain almost invariably is a "slab," interpolation from the boundaries into the interior of this region can be accomplished by the so-called transfinite interpolation concept (sometimes referred to as the blending function method). This concept was originally developed by Coons [3] and subsequently extended by Gordon [4]. One of the earliest 2D mesh generation applications using transfinite

---

\* Received by the editors July 12, 1983, and in revised form May 1, 1984.

† FFA The Aeronautical Research Institute of Sweden, 161 11 Bromma, Sweden.

interpolation is described in Gordon and Hall [5]. However, it is only during the last three or four years that the transfinite interpolation concept has been used to generate practical 3D meshes for complex geometries. A few examples of such applications are the works of Gerhard [6], Anderson and Spradley [7] and Spradley et al. [8]. In these applications, the transfinite interpolation in its simplest form is used, i.e. with no control of the normal derivatives of the mesh coordinates at the boundaries. The author's own work in 3D mesh generation (Eriksson [9], [10], Eriksson and Rizzi [11]) differs from these examples in the sense that a more sophisticated form of the transfinite interpolation is used which allows for the specification of any number of normal derivatives of the mesh coordinates on the boundaries. The precise control of the resulting coordinate system or mesh that this feature provides has made it possible to generate meshes of advanced type that are both smooth and efficient in terms of resolution for a given number of mesh points. It should be mentioned that the extension of the transfinite interpolation method to incorporate the specification of normal derivatives on the boundaries is covered by the general theory as described by Gordon [4] and is therefore not a strictly new development. However, the manner in which these normal derivatives are generated and used to control the overall mesh is certainly novel. It is the author's opinion that the key to successful mesh generation using transfinite interpolation lies exactly in this area, i.e. the specification of coordinates and normal derivatives of coordinates on the boundaries of the computational domain in such a way that the desired mesh control is obtained with a minimum of input parameters.

Apart from giving good mesh control, the transfinite interpolation concept offers speed and simplicity when implemented on computers. The speed factor is very important for 3D applications because the generation of a desirable mesh for a given geometry is usually a process involving a series of mesh generation runs with visual checks and adjustments in between. This fact is not always appreciated when evaluating the cost of mesh generation. Naturally, a good graphics software package is an integral part of any 3D mesh generation system.

The fact that transfinite interpolation offers the possibility of generating practically any desired mesh calls our attention to the question of what mesh is desirable. One way of analyzing this problem is to estimate the truncation error of the particular finite difference scheme used and deduce from this what properties the mesh should have in order to minimize the error. Unfortunately, this type of analysis requires some prior knowledge of the desired solution, which is not always available. In most practical applications, a combination of insight into the physics of the problem and computational experience is used to obtain a reasonable mesh. A good example of this is the computation of finite difference solutions to the Euler equations as presented by Rizzi [12] and Eriksson and Rizzi [11]. In these computations, which were performed upon meshes generated by the present method, full use was made of the flexibility and control offered by the transfinite interpolation concept.

The adaption of the mesh to the desired solution in order to minimize errors is not the only example of interdependence between mesh generation and equation solution. In many practical applications, mesh singularities of one sort or another must be faced and the finite difference method used may determine which type of singularity is preferable. A related topic is mesh interfacing, which is of special importance for the so-called multi-block meshes where the computational domain is composed of several interconnected "slabs." Again, the finite difference scheme used may determine the continuity conditions that must be applied to the mesh at these interfaces. It is thus clear that any practical mesh generation system must be flexible enough to satisfy



a number of requirements and it is the author's firm belief that the transfinite interpolation concept is an ideal tool for this purpose.

In the following sections, the relevant aspects of 3D mesh generation using transfinite interpolation are discussed in more detail. Section 2 treats the general topic of mapping type and mesh efficiency, using an aerodynamic application as an example. In § 3, the topic of singularities and mesh interfaces is addressed. The theory and practical application of transfinite interpolation is described at length in § 4, with 2D examples that demonstrate the importance of normal derivatives as a means of controlling the mesh. Section 5 treats the 3D case and describes the practical application of the method to a realistic aerodynamic problem. Finally, some concluding remarks are given in § 6.

**2. Mapping type.** Since the transfinite interpolation method is a means of generating a mesh in the interior of the physical domain by interpolating among the boundaries, it is clear that the first stage of the mesh generation procedure must be the specification of mesh coordinate data on the boundaries. In order to do this, it is first necessary to determine the overall structure of the mapping, the *mapping type*, so that the correspondence between the boundaries in the physical domain and the computational domain is made clear. For any given geometry, there are usually several possible mapping types with different characteristics in terms of efficiency, coordinate cuts, singularities, etc. As an example we may consider the exterior region of a typical airplane wing, for which there are at least six natural mapping types (Fig. 1). All of these alternative mapping types give surface conforming meshes but they vary markedly in terms of mesh efficiency, i.e. the resolution per mesh point. Assuming that the desired flow solution varies most rapidly near the wing surface (a quite reasonable assumption), it is evident that the density of mesh points in this region is a measure of the resolution of the mesh. One way of comparing different mappings is then to compute (approximately) the number of mesh points required for each mapping type to obtain the same density of mesh points near the wing surface. The result of such a comparison is shown

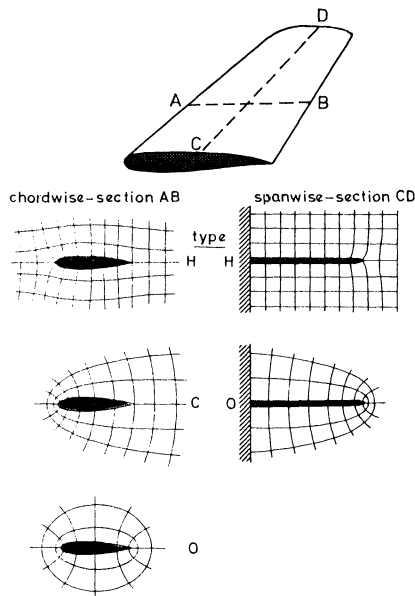


FIG. 1. Natural mapping types for a 3D airplane wing.

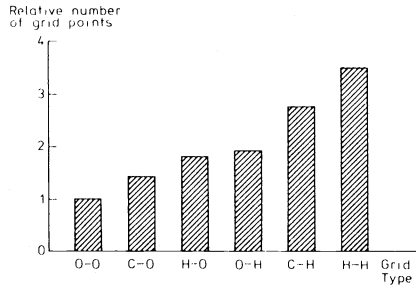


FIG. 2. Comparison of alternative mapping types in terms of relative number of mesh points for approximately equal resolution near the wing surface.

in Fig. 2 and shows that the mapping type designated *O-O* is the most efficient. The notation *O-O* is to be interpreted as “type *O* in the chordwise direction, type *O* in the spanwise direction,” using the 2D notation shown in Fig. 1.

Since the *O-O* type of mapping for the exterior wing problem is efficient and the corresponding meshes have been found to give very detailed flow solutions (using the Euler equations as a flow model), it is appropriate to view it as a realistic example of a 3D mapping. Taking a closer look at the structure of the *O-O* mapping, it is evident that it maps the region around the wing onto a “slab” in the computational space. The correspondence between the various boundaries in the physical domain and in the computational domain is shown in Fig. 3. In this exploded view, the boundaries in the physical space are drawn apart in order to make the picture more clear. It should

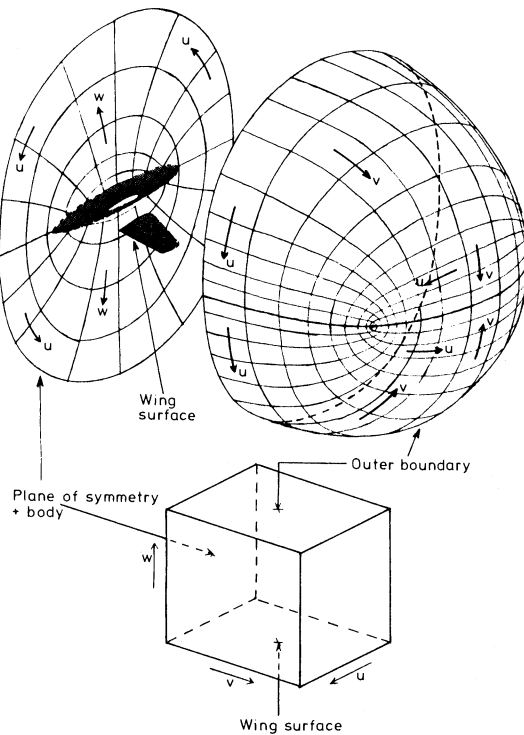


FIG. 3. Structure of the *O-O* mapping for a wing-body configuration.

also be mentioned that the addition of a fuselage is done simply by distorting the plane of symmetry and does not change the overall structure of the mapping. As the figure shows, the entire wing surface (both upper and lower surface) is mapped to the bottom of the computational box, the entire outer boundary is mapped to the top and the combined plane of symmetry and fuselage is mapped to one of the side surfaces. The remaining three surfaces of the computational box constitute coordinate cuts, i.e. they correspond to interior surfaces in the physical domain across which the various flow properties are continuous. A detail of the mapping between the wing surface and the bottom of the computational box is shown in Fig. 4. This mapping may be generated by defining the  $x, y, z$ -coordinates of the wing surface (we assume that a Cartesian

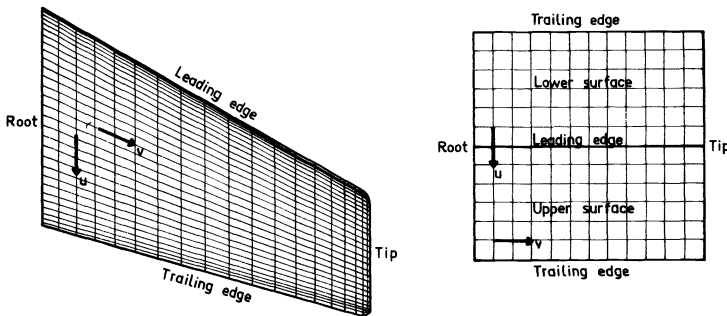


FIG. 4. Structure of the surface mapping between the wing surface and the bottom surface of the computational box for the  $O-O$  mapping.

coordinate system is defined in the physical domain) as functions of the two  $u, v$ -coordinates in computational space. If the  $u, v$ -coordinates are then discretized uniformly, the corresponding  $x, y, z$ -coordinates will form a curvilinear mesh on the wing surface, as indicated in Fig. 4. It is evident that the same procedure can be used to obtain the corresponding surface meshes on the outer boundary and on the combined plane of symmetry and fuselage. Once these surface meshes are specified, the first stage of the mesh generation procedure is completed and the remaining task is the extension of the surface meshes into the interior of the domain. Working in the computational domain, this amounts to the extension of the functions  $x(u, v, w_1), y(u, v, w_1), z(u, v, w_1), x(u, v, w_2), y(u, v, w_2), z(u, v, w_2), x(u, v_1, w), y(u, v_1, w), z(u, v_1, w)$  to the entire computational region  $u_1 \leq u \leq u_2, v_1 \leq v \leq v_2, w_1 \leq w \leq w_2$ . The idea of using transfinite interpolation to accomplish this extension is thus very natural. However, the details of this interpolation are discussed in §§ 4 and 5.

The discussion so far has been limited to the topic of single-block meshes, i.e. meshes that map the physical domain onto a "slab" in the computational domain. This type of mapping is very desirable due to the simplicity of the computational domain. However, for very complicated geometries it can be difficult to generate single-block meshes that are both reasonably smooth and efficient. An example of such a complex region is the exterior of a complete airplane with several lifting surfaces. A possible solution to this problem is to use multi-block meshes, which map the physical domain onto several interconnected slabs in the computational domain. This technique is also referred to as subdomain methods or zonal methods. The development of multi-block meshes and the practical use of such meshes has not yet come as far as the corresponding development of single-block meshes, but the next few years may see a change in this situation. As far as transfinite interpolation is concerned, the

generation of multi-block meshes does not in principle differ from the generation of single-block meshes. Indeed, transfinite interpolation is an ideal method for this purpose since any continuity requirements at mesh interfaces can be satisfied explicitly.

**3. Singularities and interfaces.** In connection with mapping types it is appropriate to discuss the topic of singularities and interfaces. Mesh singularities are undesirable but very often unavoidable and any practical finite difference scheme must be able to cope with them. A familiar example of a mesh singularity in two space dimensions is the polar singularity of a polar coordinate system. In three dimensions, the exterior wing problem discussed in the previous section serves as an instructive example of the kinds of mesh singularities that must be faced for various mapping types. Starting with the  $O-O$  type of mapping, it is readily shown that it gives rise to two singular *lines* in the mesh, extending from the two tip corners of the wing to the outer boundary (Fig. 5). A detail of the leading edge/tip corner of the wing (Fig. 6) shows that this

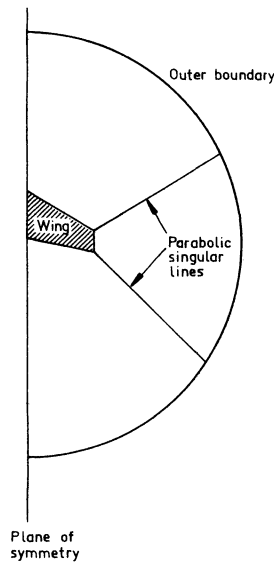


FIG. 5. Position of singular lines for the  $O-O$  mapping.

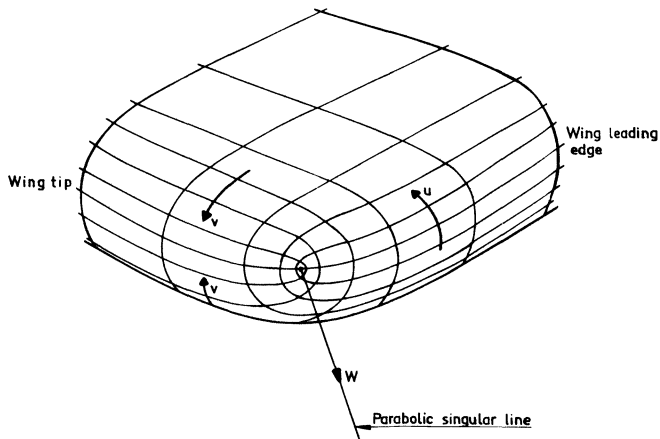


FIG. 6. Detail of wing surface mesh at leading edge/tip corner for the  $O-O$  mapping.

singularity is similar in nature to the singular line of a cylindric-parabolic coordinate system and can therefore be referred to as a "parabolic singular line." The other tip corner of the wing, the trailing edge/tip corner, has the same type of singular line emanating from the surface, but due to the sharp trailing edge the singularity has a more complicated structure near the wing surface (Fig. 7). The fact that the trailing edge is sharp implies that this edge also is a singular line in the mesh. However, it is clear that this singularity falls into another category since it is caused by the actual boundary geometry and not by the choice of mapping type.

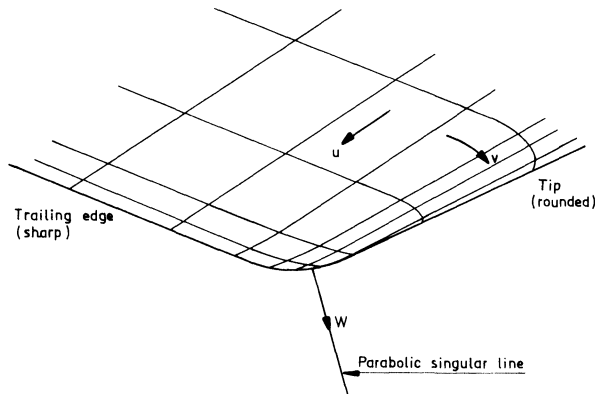


FIG 7. Detail of wing surface mesh at trailing edge/tip corner for the  $O-O$  mapping.

A similar inspection of the  $O-H$  type of mapping reveals that in this case there are also two singular lines of parabolic type stretching between the tip corners and the outer boundary. However, in addition to these singularities, the  $O-H$  mapping also gives rise to a singular line on the wing surface, as indicated in Fig. 8. This is due to the fact that the mesh surface that coincides with the wing is extended outside the wing tip. The nature of this surface-bound singular line is shown in Fig. 9, and it is typical of all  $H$ -type mappings.

If we for the moment assume that the wing trailing edge is not sharp, i.e. that the wing is a smooth closed surface, then it is easily shown that the  $O-O$  type of mapping

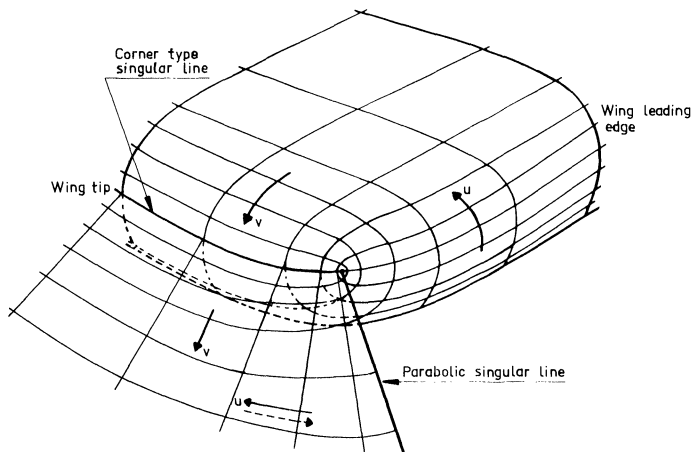


FIG. 8. Detail of wing surface mesh at leading edge/tip corner for the  $O-H$  mapping.

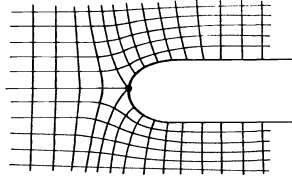


FIG. 9. Mesh singularity caused by *H*-type mapping.

is unique in that it does not create any surface-bound singular line. Taking the sharp trailing edge into account, it is evident that this edge will always be a singular line of some sort or other, regardless of the mapping type. Since the *C-O* mapping type only differs from the *O-O* type in that it creates a singular line along the trailing edge, it is reasonable to say that the *O-O* and *C-O* mapping types are unique in that they do not create any *additional* surface-bound singular lines.

It is clear that the existence of surface-bound singular lines of the type shown in Fig. 9 is very undesirable as it may affect the accuracy of the boundary conditions to a large extent. From this point of view, the other type of singular line that stretches between two surfaces should be preferable, partly because it only affects the surfaces at isolated points and partly because the singularity type of these lines is such that there is a concentration of mesh points around the lines which counteracts the increase of truncation error. This heuristic reasoning is supported by the available computational experience. A deeper analysis of the errors due to mesh singularities is not a trivial task and necessarily involves the equations to be solved, the type of finite difference scheme used, etc. An example of such an analysis is given in Eriksson [13] where the error caused by a parabolic mesh singularity in two space dimensions is analyzed for a linear hyperbolic model problem using central difference approximations. One result of this work is that the so-called finite-volume method, which is a conservative cell-oriented method, can be shown to be stable for any linear first-order symmetrizable hyperbolic system regardless of the type of singularity involved. Furthermore, it is possible to show that in the case of a parabolic singularity the scheme is "almost" second order accurate when measuring the truncation error by the natural  $L_2$  norm. These results together with the available computational experience indicate that the interior parabolic singular lines inherent in most of the mappings considered for the exterior wing problem can be dealt with in a satisfactory manner. In the case of the surface-bound singular lines of the character shown in Fig. 9, a simple analysis of truncation errors for the usual finite difference approximations reveals that these are much more severe than the parabolic singularities. Since it is only the *O-O* and *C-O* mapping types that do not introduce any such severe singularities, the overall conclusion must be that these two mappings are the best, both from the viewpoint of efficiency (as discussed in the previous section) and from the viewpoint of accuracy. Strangely enough, the *C-H* mapping has so far been the most popular type for flow computations around wings, even though this mapping has the more severe kind of singular line along the wing tip. Possibly this may be explained by the fact that the *C-H* mapping can be obtained by a simple "stacking" of 2D chordwise meshes (*C*-type) in the spanwise direction, i.e. by a "quasi-3D" method. It seems clear that the price to be paid for using such a simple mesh generation technique is high.

The topic of mesh interfaces may at first glance seem to be totally unrelated to the topic of mesh singularities. However, it is possible to view interfaces as a more general form of distributed singularity. In order to see this, we may consider the mesh singularity depicted in Fig. 10a. If the mapping is forced to be smooth everywhere

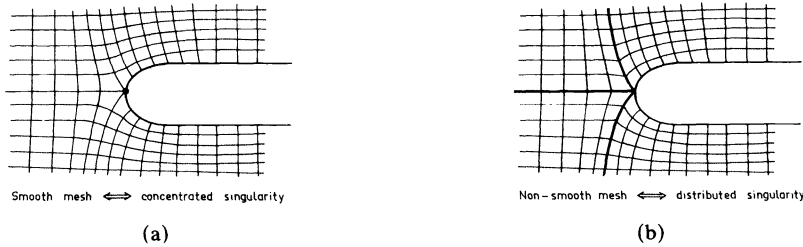


FIG. 10. (a) Singular point caused by H-type smooth mapping. (b) Mesh interfaces (= distributed singularity) caused by H-type nonsmooth mapping.

except at isolated points, it is clear that the result is a concentrated singularity as shown in the figure. If on the other hand the mapping is allowed to be nonsmooth along certain lines, the result may be as shown in Fig. 10b. The overall mapping type of these two meshes are identical but they differ in that the second alternative gives typical mesh interfaces where the mesh metrics are discontinuous. It is quite natural to interpret this as a case of concentrated singularity versus a case of distributed singularity. Obviously, the practical handling of mesh interfaces is not affected by a change of viewpoint, but it is sometimes useful to see the overall connection between different aspects of mesh generation.

**4. Transfinite interpolation.**

*Theoretical background.* Transfinite interpolation is a special case of multivariate interpolation and the theory is best described in terms of projections. Consider at first the subproblem of interpolating an arbitrary function  $f(x)$  on some interval  $[a, b]$ . Given a number of base functions  $\phi_1(x), \phi_2(x), \dots, \phi_n(x)$  the interpolant  $g$  is defined as

$$(1) \quad g(x) = \sum_{j=1}^n c_j \phi_j(x)$$

and is required to coincide with  $f(x)$  at the points  $x_1, x_2, \dots, x_n$  in the interval  $[a, b]$ :

$$(2) \quad g(x_i) = \sum_{j=1}^n c_j \phi_j(x_i) = f(x_i), \quad i = 1, \dots, n.$$

This can be written in the more compact form

$$(3) \quad Ac = d$$

where

$$A = \begin{bmatrix} \phi_1(x_1) & \dots & \phi_n(x_1) \\ \vdots & & \vdots \\ \phi_1(x_n) & \dots & \phi_n(x_n) \end{bmatrix}, \quad c = \begin{bmatrix} c_1 \\ \vdots \\ c_n \end{bmatrix}, \quad d = \begin{bmatrix} f(x_1) \\ \vdots \\ f(x_n) \end{bmatrix}.$$

If the problem is well-posed,  $A$  can be inverted and the coefficient vector  $c$  is given by

$$(4) \quad c = A^{-1}d \quad \text{or} \quad c_i = \sum_{j=1}^n (A^{-1})_{ij} f(x_j), \quad i = 1, \dots, n.$$

The interpolant  $g(x)$  is thus uniquely determined by (1) and (4)

$$(5) \quad g(x) = \sum_{j=1}^n f(x_j) \sum_{i=1}^n (A^{-1})_{ij} \phi_i(x).$$

Defining a new set of functions  $\Psi_1(x), \dots, \Psi_n(x)$  by

$$(6) \quad \Psi_i(x) = \sum_{j=1}^n (A^{-1})_{ji} \phi_j(x)$$

we obtain from (5) and (6):

$$(7) \quad g(x) = \sum_{i=1}^n f(x_i) \Psi_i(x)$$

which implies that the new functions satisfy the relations

$$(8) \quad \Psi_i(x_j) = \delta_{ij}.$$

From this we conclude that any well-posed univariate linear interpolation scheme can be written in the form (7) with functions  $\Psi_1, \dots, \Psi_n$  that satisfy (8). An immediate consequence of this is that two applications of the interpolation scheme give the same result as one application:

*1st application.*

$$g_1(x) = \sum_{i=1}^n f(x_i) \Psi_i(x);$$

*2nd application.*

$$\begin{aligned} g_2(x) &= \sum_{i=1}^n g_1(x_i) \Psi_i(x) \\ &= \sum_{i=1}^n \sum_{j=1}^n f(x_j) \Psi_j(x_i) \Psi_i(x) \\ &= \sum_{i=1}^n f(x_i) \Psi_i(x) = g_1(x). \end{aligned}$$

This means that the interpolation scheme is a projection, i.e. it defines an idempotent linear operator  $\pi$  such that

$$(9) \quad g = \pi f.$$

Writing a complete univariate interpolation scheme as a projection gives a very compact notation that makes it easy to formulate the concept of transfinite interpolation. Consider now a function  $f(x, y)$  of two variables defined on the rectangular region  $[a, b] \times [c, d]$ . Let  $\pi_x$  and  $\pi_y$  be two projections that correspond to some univariate interpolation schemes in the  $x$  and  $y$  directions respectively. This means that the function  $\pi_x f$  coincides with  $f$  along some lines  $x = x_1, x = x_2, \dots, x = x_m$  where  $x_1, \dots, x_m \in [a, b]$  and the function  $\pi_y f$  coincides with  $f$  along some lines  $y = y_1, y = y_2, \dots, y = y_n$  where  $y_1, \dots, y_n \in [c, d]$ . An important property of the projections  $\pi_x$  and  $\pi_y$  is that they commute, i.e.  $\pi_x \pi_y = \pi_y \pi_x$  (this can easily be proved by using formulation (7)). This means that the operator  $\pi_x \pi_y$  is also a projection, since

$$(\pi_x \pi_y)^2 = \pi_x \pi_y \pi_x \pi_y = \pi_x \pi_x \pi_y \pi_y = \pi_x \pi_y.$$

The function  $\pi_x \pi_y f$  can be shown to coincide with  $f$  at the  $m \times n$  points  $(x_i, y_j)$ ;  $i = 1, \dots, m, j = 1, \dots, n$  and is usually called the tensor product interpolant of  $f$ .

If we instead study the operator  $\pi_x + \pi_y - \pi_x \pi_y$  it is evident that this is also a projection since

$$\begin{aligned} (\pi_x + \pi_y - \pi_x \pi_y)^2 &= \pi_x^2 + \pi_y^2 + \pi_x^2 \pi_y^2 + 2\pi_x \pi_y - 2\pi_x^2 \pi_y - 2\pi_x \pi_y^2 \\ &= \pi_x + \pi_y - \pi_x \pi_y. \end{aligned}$$



Furthermore, the function  $(\pi_x + \pi_y - \pi_x\pi_y)f$  can be shown to coincide with  $f$  along the lines  $x = x_1, \dots, x = x_m$  and  $y = y_1, \dots, y = y_n$ , i.e. it combines the properties of  $\pi_x$  and  $\pi_y$ . The proof is very simple and makes use of the explicit formulation (7) of an arbitrary univariate interpolation scheme. The projection  $\pi_x + \pi_y - \pi_x\pi_y$  is sometimes referred to as the Boolean sum  $\pi_x \oplus \pi_y$  of  $\pi_x$  and  $\pi_y$  and the function  $\pi_x \oplus \pi_y f$  is usually called a *transfinite* interpolant because of its coincidence with  $f$  along lines instead of at isolated points.

The concept of transfinite interpolation is easily extended to functions of more than two variables. For three variables, the Boolean sum of the three univariate interpolation operators  $\pi_x, \pi_y$  and  $\pi_z$  is found to be

$$(10) \quad \pi_x \oplus \pi_y \oplus \pi_z = \pi_x + \pi_y + \pi_z - \pi_x\pi_y - \pi_x\pi_z - \pi_y\pi_z + \pi_x\pi_y\pi_z.$$

A useful formula for constructing the Boolean sum of an arbitrary number of operators is the recursive algorithm

$$(11) \quad \begin{aligned} \pi_1^* &= \pi_1, \\ \pi_2^* &= \pi_1^* + \pi_2(I - \pi_1^*), \\ \pi_3^* &= \pi_2^* + \pi_3(I - \pi_2^*), \\ &\vdots \\ \pi_n^* &= \pi_{n-1}^* + \pi_n(I - \pi_{n-1}^*), \\ \pi_1 \oplus \pi_2 \oplus \dots \oplus \pi_n &= \pi_n^*. \end{aligned}$$

It should be mentioned that the general formulation (7) and (8) of an arbitrary linear univariate interpolation scheme is easily extended to osculatory interpolation by writing the interpolant as

$$(12) \quad g(x) = \sum_{i=1}^n \sum_{j=0}^{p_i} f^{(j)}(x_i) \Psi_{ij}(x)$$

where the functions  $\Psi_{ij}(x)$  satisfy the conditions

$$(13) \quad \left( \frac{d^l}{dx^l} \Psi_{ij} \right) (x_k) = \delta_{ik} \delta_{jl}.$$

This means that osculatory interpolation schemes like, for example, cubic Hermite interpolation are also projections that can be combined with other projections to form transfinite interpolants. We shall see from the applications of the transfinite interpolation concept that osculatory interpolation schemes are important for mesh generation.

*Application to mesh generation.* The use of transfinite interpolation to generate meshes is an unusual application of this multivariate interpolation method in the sense that the “quality” of the interpolant is not judged by its “error” but rather by looking at the general properties of the resulting mesh. In order to see this, we consider at first the simple 2D problem of mapping an arbitrary region in  $x, y$ -space onto a rectangle in  $u, v$ -space (Fig. 11). We assume that the mapping function  $f(u, v) = (x(u, v), y(u, v))$  is known on the four boundaries of the rectangle, i.e. that the parametric representations  $f(u_1, v), f(u_2, v), f(u, v_1), f(u, v_2)$  of the four boundary curves in  $x, y$ -space are defined. The problem then consists of extending the mapping function  $f(u, v)$  from the boundaries to the interior of the rectangle in  $u, v$ -space. In order to construct a suitable transfinite projection for this purpose, we must first construct some univariate interpolation schemes in the  $u$  and  $v$  directions. Whatever schemes we choose, they can be

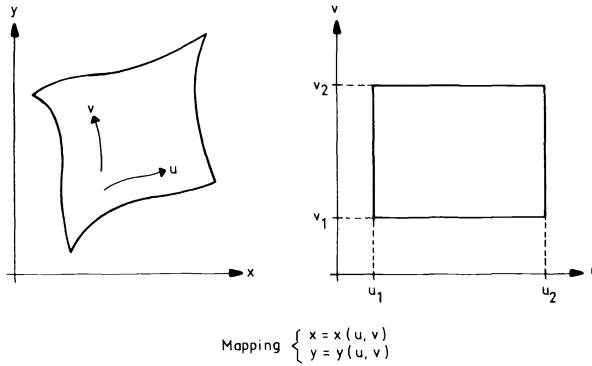


FIG. 11. The 2D mesh generation problem.

written in the form (7) described before:

$$(14) \quad \begin{aligned} \pi_u f &= f(u_1, v)\alpha_1(u) + f(u_2, v)\alpha_2(u), \\ \pi_v f &= f(u, v_1)\beta_1(v) + f(u, v_2)\beta_2(v), \end{aligned}$$

with

$$(15) \quad \begin{aligned} \alpha_1(u_1) &= 1, & \alpha_1(u_2) &= 0, \\ \alpha_2(u_1) &= 0, & \alpha_2(u_2) &= 1, \\ \beta_1(v_1) &= 1, & \beta_1(v_2) &= 0, \\ \beta_2(v_1) &= 0, & \beta_2(v_2) &= 1. \end{aligned}$$

By forming the Boolean sum of these projections, we obtain the transfinite interpolation scheme

$$(16) \quad \begin{aligned} \pi_u \oplus \pi_v f &= (\pi_u + \pi_v - \pi_u \pi_v) f \\ &= f(u_1, v)\alpha_1(u) + f(u_2, v)\alpha_2(u) \\ &\quad + f(u, v_1)\beta_1(v) + f(u, v_2)\beta_2(v) \\ &\quad - f(u_1, v_1)\alpha_1(u)\beta_1(v) - f(u_2, v_1)\alpha_2(u)\beta_1(v) \\ &\quad - f(u_1, v_2)\alpha_1(u)\beta_2(v) - f(u_2, v_2)\alpha_2(u)\beta_2(v). \end{aligned}$$

It is readily confirmed that this interpolant agrees with the given function along the boundaries of the rectangle in  $u, v$ -space. The functions  $\alpha_1, \alpha_2, \beta_1, \beta_2$  have to satisfy relation (15) but otherwise we are free to choose them in any convenient way.

Before showing any examples of this transfinite interpolation scheme, it is convenient to describe the extension of the method to incorporate the specification of normal derivatives of  $f$  at the boundaries. In this case we assume that the function  $\partial f / \partial u$  is known at the boundaries  $u = u_1$  and  $u = u_2$ , and  $\partial f / \partial v$  at the boundaries  $v = v_1$  and  $v = v_2$ . The univariate interpolation schemes in the  $u$  and  $v$  directions must then be osculatory schemes and can be written in the general form:

$$(17) \quad \begin{aligned} \pi_u f &= f(u_1, v)\alpha_1(u) + \frac{\partial f}{\partial u}(u_1, v)\alpha_2(u) + f(u_2, v)\alpha_3(u) + \frac{\partial f}{\partial u}(u_2, v)\alpha_4(u), \\ \pi_v f &= f(u, v_1)\beta_1(v) + \frac{\partial f}{\partial v}(u, v_1)\beta_2(v) + f(u, v_2)\beta_3(v) + \frac{\partial f}{\partial v}(u, v_2)\beta_4(v), \end{aligned}$$

with

$$(18) \quad \begin{aligned} \alpha_1(u_1) &= 1, & \alpha'_1(u_1) &= 0, & \alpha_1(u_2) &= 0, & \alpha'_1(u_2) &= 0, \\ \alpha_2(u_1) &= 0, & \alpha'_2(u_1) &= 1, & \alpha_2(u_2) &= 0, & \alpha'_2(u_2) &= 0, \\ \alpha_3(u_1) &= 0, & \alpha'_3(u_1) &= 0, & \alpha_3(u_2) &= 1, & \alpha'_3(u_2) &= 0, \\ \alpha_4(u_1) &= 0, & \alpha'_4(u_1) &= 0, & \alpha_4(u_2) &= 0, & \alpha'_4(u_2) &= 1, \end{aligned}$$

and similarly for  $\beta_1, \beta_2, \beta_3, \beta_4$ .

As before, we form the Boolean sum of these two projections and obtain the transfinite interpolation scheme

$$(19) \quad \begin{aligned} \pi_u \oplus \pi_v f &= (\pi_u + \pi_v - \pi_u \pi_v) f \\ &= f(u_1, v) \alpha_1(u) + \frac{\partial f}{\partial u}(u_1, v) \alpha_2(u) + f(u_2, v) \alpha_3(u) \\ &\quad + \frac{\partial f}{\partial u}(u_2, v) \alpha_4(u) + f(u, v_1) \beta_1(v) + \frac{\partial f}{\partial v}(u, v_1) \beta_2(v) \\ &\quad + f(u, v_2) \beta_3(v) + \frac{\partial f}{\partial v}(u, v_2) \beta_4(v) - f(u_1, v_1) \alpha_1(u) \beta_1(v) \\ &\quad - \frac{\partial f}{\partial u}(u_1, v_1) \alpha_2(u) \beta_1(v) - f(u_2, v_1) \alpha_3(u) \beta_1(v) - \frac{\partial f}{\partial u}(u_2, v_1) \alpha_4(u) \beta_1(v) \\ &\quad - \frac{\partial f}{\partial v}(u_1, v_1) \alpha_1(u) \beta_2(v) - \frac{\partial^2 f}{\partial u \partial v}(u_1, v_1) \alpha_2(u) \beta_2(v) - \frac{\partial f}{\partial v}(u_2, v_1) \alpha_3(u) \beta_2(v) \\ &\quad - \frac{\partial^2 f}{\partial u \partial v}(u_2, v_1) \alpha_4(u) \beta_2(v) - f(u_1, v_2) \alpha_1(u) \beta_3(v) - \frac{\partial f}{\partial u}(u_1, v_2) \alpha_2(u) \beta_3(v) \\ &\quad - f(u_2, v_2) \alpha_3(u) \beta_3(v) - \frac{\partial f}{\partial u}(u_2, v_2) \alpha_4(u) \beta_3(v) - \frac{\partial f}{\partial v}(u_1, v_2) \alpha_1(u) \beta_4(v) \\ &\quad - \frac{\partial^2 f}{\partial u \partial v}(u_1, v_2) \alpha_2(u) \beta_4(v) - \frac{\partial f}{\partial v}(u_2, v_2) \alpha_3(u) \beta_4(v) \\ &\quad - \frac{\partial^2 f}{\partial u \partial v}(u_2, v_2) \alpha_4(u) \beta_4(v). \end{aligned}$$

It should be mentioned at this point that for practical applications, a recursive algorithm based on (11) is more convenient to use than the rather lengthy explicit form (19), but it is instructive to derive the complete formula and check that the interpolant really satisfies the boundary conditions. The transfinite interpolation schemes (16) and (19) serve as examples to show the general principles of the method, and in the general case, with  $f$  and any number of normal derivatives of  $f$  given on one or several of the four boundaries, the corresponding transfinite interpolation scheme is derived in exactly the same manner.

In order to demonstrate the superiority of the osculatory transfinite interpolation scheme, the geometry shown in Fig. 12 is a suitable test case. Defining the four boundary curves  $A, B, C, D$  is here equivalent to defining the functions  $f(u, v_1), f(u_2, v), f(u, v_2), f(u_1, v)$ , where  $f = (x, y)$  as before. Since this parametric representation of the boundary curves is nonunique, we may choose the functions in such a way that a suitable

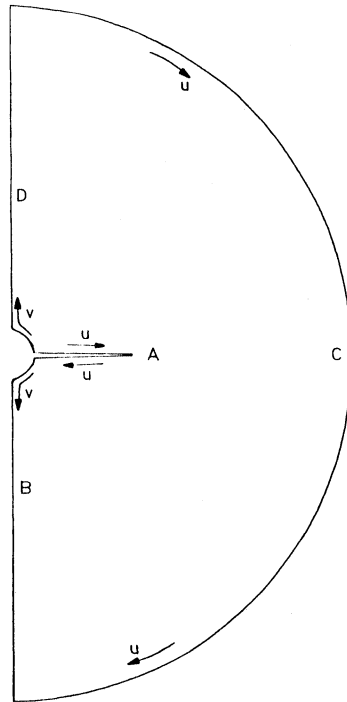


FIG. 12. 2D test case for transfinite interpolation. A = wing, B and D = combined plane of symmetry and body, C = outer boundary.

distribution of mesh points is obtained for a uniform discretization of  $u$  and  $v$ . Once the mapping function  $f$  is specified at the four boundaries, we can use the simplest transfinite interpolation scheme (16) to obtain a complete mapping in the entire domain. A uniform discretization in  $u$  and  $v$  will then give a corresponding curvilinear mesh in the  $x, y$ -space. The mesh obtained for the simplest possible choice of the functions  $\alpha_1, \alpha_2, \beta_1, \beta_2$ ;

$$(20) \quad \begin{aligned} \alpha_1(u) &= \frac{u_2 - u}{u_2 - u_1}, & \beta_1(v) &= \frac{v_2 - v}{v_2 - v_1}, \\ \alpha_2(u) &= \frac{u - u_1}{u_2 - u_1}, & \beta_2(v) &= \frac{v - v_1}{v_2 - v_1}, \end{aligned}$$

corresponding to linear interpolation in  $u$  and  $v$ , is shown in Fig. 13. It is evident that this mesh is not very satisfactory, even though the distribution of mesh points on the boundaries was chosen to be reasonable (the geometry can be seen as a cross-section of a typical airplane). The most severe flaw in the mesh is the discontinuity caused by the two “kinks” in the boundary curves  $B$  and  $D$ . A possible way of damping the influence of these kinks in the interior of the mesh is to alter the interpolation in the  $u$ -direction. By constructing the functions  $\alpha_1(u)$  and  $\alpha_2(u)$  such that they tend to be very small away from the points  $u_1$  and  $u_2$ , like for example

$$(21) \quad \alpha_1(u) = \frac{u_2 - u}{u_2 - u_1} \exp \left[ -K \left( \frac{u - u_1}{u_2 - u_1} \right) \right], \quad \alpha_2(u) = \frac{u - u_1}{u_2 - u_1} \exp \left[ -K \left( \frac{u_2 - u}{u_2 - u_1} \right) \right].$$

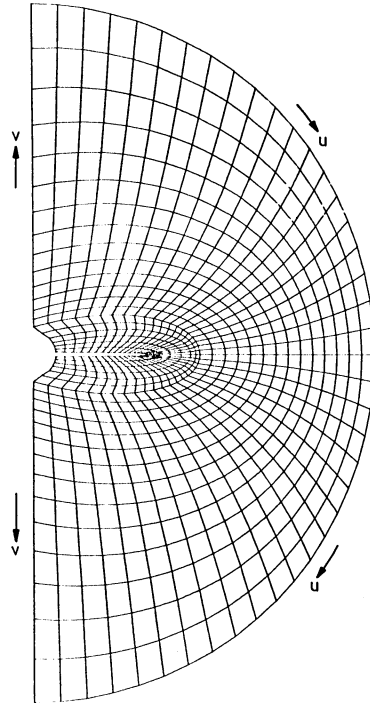


FIG. 13. Mesh obtained for the 2D test case using simplest possible transfinite interpolation (linear interpolation in both  $u$  and  $v$ ).

Where  $K$  is a large positive constant, we can achieve this goal as indicated by the mesh shown in Fig. 14. Here we see the gradual damping of the kinks in the interior of the mesh. However, this brings out the next serious flaw in the mesh. It is evident that the distribution of mesh points in the vicinity of the wing and especially at the wing tip is very poor. This problem is virtually impossible to solve just by altering the functions  $\beta_1(v)$  and  $\beta_2(v)$  for the interpolation scheme in the  $v$ -direction and it is evident that the transfinite interpolation method “needs” some more information about the mapping function in order to give the desired mesh. The natural information to give in this case is the specification of the normal derivative of  $f$  at the wing boundary  $A$ , i.e. the function  $(\partial f / \partial v)(u, v_1)$ . Since  $f$  is a vector-valued function, this amounts to the specification of both the *direction* of the outgoing mesh curves and the *spacing* of the mesh along these lines. Obviously, this is exactly the kind of control that is desired at a surface. To incorporate this extra information in the transfinite interpolation scheme, we use the osculatory interpolation scheme

$$(22) \quad \pi_v f = f(u, v_1)\beta_1(v) + \frac{\partial f}{\partial v}(u, v_1)\beta_2(v) + f(u, v_2)\beta_3(v)$$

in the  $v$ -direction, where  $\beta_1, \beta_2, \beta_3$  satisfy

$$(23) \quad \begin{aligned} \beta_1(v_1) &= 1, & \beta_1'(v_1) &= 0, & \beta_1(v_2) &= 0, \\ \beta_2(v_1) &= 0, & \beta_2'(v_1) &= 1, & \beta_2(v_2) &= 0, \\ \beta_3(v_1) &= 0, & \beta_3'(v_1) &= 0, & \beta_3(v_2) &= 1. \end{aligned}$$

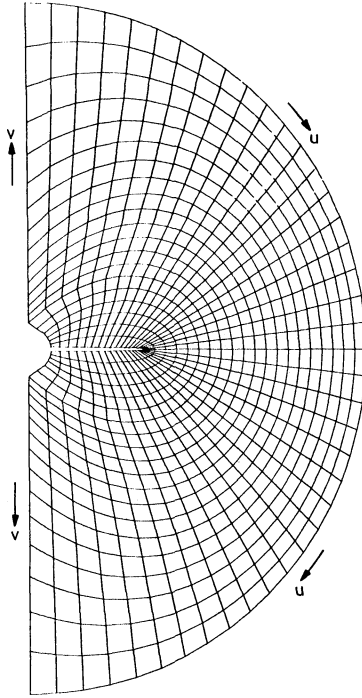


FIG. 14. Mesh obtained for the 2D test case using the same method as in Fig. 13 except that the interpolation in  $u$  is altered.

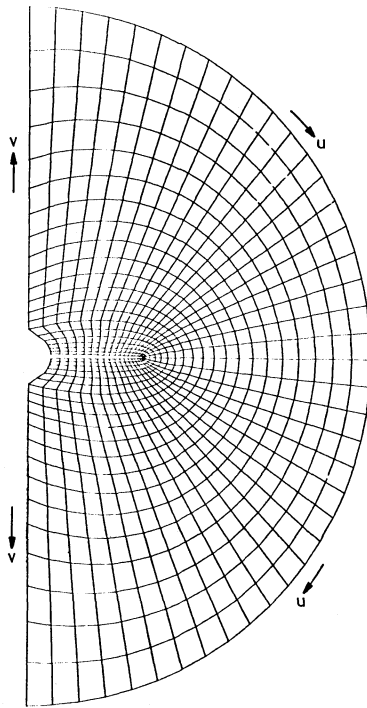


FIG. 15. Mesh obtained for the 2D test case using oscillatory transfinite interpolation. The normal derivative of the mapping function is specified on the wing (boundary A).

The simplest choice of  $\beta_1, \beta_2, \beta_3$  are the polynomial functions

$$\begin{aligned}
 \beta_1(v) &= 1 - \left( \frac{v - v_1}{v_2 - v_1} \right)^2, \\
 \beta_2(v) &= \left[ \left( \frac{v - v_1}{v_2 - v_1} \right) - \left( \frac{v - v_1}{v_2 - v_1} \right)^2 \right] (v_2 - v_1), \\
 \beta_3(v) &= \left( \frac{v - v_1}{v_2 - v_1} \right)^2.
 \end{aligned}
 \tag{24}$$

Forming the Boolean sum of the  $\pi_u$  projection in (14) with  $\alpha_1, \alpha_2$  given by (21) and the  $\pi_v$  projection (22) with  $\beta_1, \beta_2, \beta_3$  given by (24), we obtain a transfinite interpolation scheme that allows us to specify the normal derivative of  $f$  on the wing boundary  $A$ . Choosing this derivative such that the mesh is almost orthogonal at the wing boundary, we obtain the mesh shown in Fig. 15. It is evident that the control offered by the specification of normal derivatives of the mapping function makes this osculatory type of transfinite interpolation much more attractive than the simpler method. Just to show some further possibilities of the method, the mesh obtained by using the functions

$$\begin{aligned}
 \beta_1(v) &= 1 - \frac{e^{Kv'} - 1 - Kv'}{e^K - 1 - K}, \\
 \beta_2(v) &= \left[ v' - \frac{e^{Kv'} - 1 - Kv'}{e^K - 1 - K} \right] (v_2 - v_1), \\
 \beta_3(v) &= \frac{e^{Kv'} - 1 - Kv'}{e^K - 1 - K}, \\
 v' &= (v - v_1)/(v_2 - v_1), \quad K = \text{positive constant},
 \end{aligned}
 \tag{25}$$

instead of those given by (24) is shown in Fig. 16 (the  $K$ -constant used in (25) does not have to be the same as the  $K$ -constant used in (21)). The effect of using this choice of  $\beta_1, \beta_2, \beta_3$  is to “delay” the influence of the outer boundary, i.e. the region where the normal derivative of  $f$  on the wing boundary  $A$  controls the mesh is increased as  $K$  is increased.

These simple 2D examples suffice to show the general principles of mesh generation using transfinite interpolation and to demonstrate the advantages of using osculatory interpolation in terms of mesh control.

**5. 3D mesh generation.** As mentioned in § 4, the derivation of transfinite interpolation schemes in explicit form, like for example scheme (19) in § 4, is not very practical in the general multi-variable case since the formulas tend to be rather lengthy. It is, however, possible to write any transfinite interpolation scheme as a recursive algorithm composed of univariate interpolation steps. The basis of this algorithm is the recursive relation (11) in § 4 that gives the Boolean sum of an arbitrary number of projections. Applying this relation to the three-variable case with the univariate interpolation projections  $\pi_u, \pi_v, \pi_w$ , we obtain (the ordering of  $\pi_u, \pi_v, \pi_w$  is arbitrary)

$$\begin{aligned}
 \pi_u^* &= \pi_u, \\
 \pi_v^* &= \pi_u^* + \pi_v(I - \pi_u^*), \\
 \pi_w^* &= \pi_v^* + \pi_w(I - \pi_v^*), \\
 \pi_u \oplus \pi_v \oplus \pi_w &= \pi_w^*.
 \end{aligned}
 \tag{26}$$

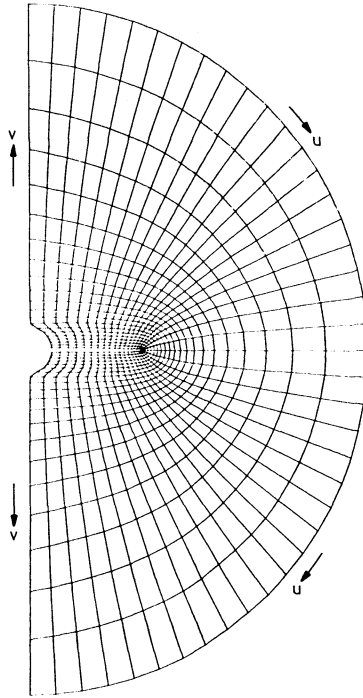


FIG. 16. Mesh obtained for the 2D test case using the same method as in Fig. 15 except that the interpolation in  $v$  is altered.

If we apply these projection steps to a function  $f(u, v, w)$  and set  $f_1 = \pi_u^* f$ ,  $f_2 = \pi_v^* f$ ,  $f_3 = \pi_w^* f$ ,  $\tilde{f} = \pi_u \oplus \pi_v \oplus \pi_w f$ , we get

$$\begin{aligned}
 f_1 &= \pi_u f, \\
 f_2 &= f_1 + \pi_v(f - f_1), \\
 f_3 &= f_2 + \pi_w(f - f_2), \\
 \tilde{f} &= f_3.
 \end{aligned}
 \tag{27}$$

From this we see that the final transfinite interpolant  $\tilde{f}$  is obtained by three successive univariate interpolation steps.

We are now ready to treat a fully three-dimensional case of mesh generation. Once again, we choose the problem of generating meshes around wings and wing-body configurations as a realistic example of mesh generation using transfinite interpolation. In particular, we choose the  $O-O$  mapping type that was described in § 2 and proceed to construct appropriate univariate interpolation schemes for this case. The mapping function  $f(u, v, w) = (x(u, v, w), y(u, v, w), z(u, v, w))$  that maps the physical domain in  $x, y, z$ -space onto the computational box  $[u_1, u_2] \times [v_1, v_2] \times [w_1, w_2]$  in  $u, v, w$ -space is specified at three of the six boundary planes of this computational box (Fig. 3). The “bottom” surface corresponds to the wing surface, i.e.  $f(u, v, w_1)$  is a parametric representation of the wing surface in  $x, y, z$ -space. Just as for curves, the parametric representation of a given surface is not unique, and the function  $f(u, v, w_1)$  can therefore be chosen in such a way that a desirable surface mesh is obtained on the wing for a uniform discretization of the  $u$  and  $v$  variables. In the same way the mapping function at the top of the computational box,  $f(u, v, w_2)$ , is a parametric representation of the



outer boundary and the mapping function at the side surface  $v = v_1, f(u, v_1, w)$ , is a parametric representation of the combined plane of symmetry and fuselage. In addition to these functions, we also specify the normal derivatives  $\partial f/\partial w(u, v, w_1), \partial^2 f/\partial w^2(u, v, w_1), \partial^3 f/\partial w^3(u, v, w_1)$  on the wing surface to obtain the desired mesh control in the vicinity of the wing. The choice of three derivatives and the generation of these derivatives will be discussed, but at this stage we simply assume that the derivatives are defined. The appropriate univariate interpolation schemes for this case can now be formulated:

$$\begin{aligned} \pi_u f &= 0 \quad \text{since nothing is specified at the boundaries } u = u_1, u = u_2; \\ \pi_v f &= f(u, v_1, w)\beta(v); \\ \pi_w f &= f(u, v, w_1)\gamma_1(w) + \frac{\partial f}{\partial w}(u, v, w_1)\gamma_2(w) + \frac{\partial^2 f}{\partial w^2}(u, v, w_1)\gamma_3(w) \\ &\quad + \frac{\partial^3 f}{\partial w^3}(u, v, w_1)\gamma_4(w) + f(u, v, w_2)\gamma_5(w); \end{aligned}$$

with  $\beta(v_1) = 1$

$$\begin{aligned} (28) \quad &\gamma_1(w_1) = 1, \quad \gamma_1'(w_1) = 0, \quad \gamma_1''(w_1) = 0, \quad \gamma_1'''(w_1) = 0, \quad \gamma_1(w_2) = 0, \\ &\gamma_2(w_1) = 0, \quad \gamma_2'(w_1) = 1, \quad \gamma_2''(w_1) = 0, \quad \gamma_2'''(w_1) = 0, \quad \gamma_2(w_2) = 0, \\ &\gamma_3(w_1) = 0, \quad \gamma_3'(w_1) = 0, \quad \gamma_3''(w_1) = 1, \quad \gamma_3'''(w_1) = 0, \quad \gamma_3(w_2) = 0, \\ &\gamma_4(w_1) = 0, \quad \gamma_4'(w_1) = 0, \quad \gamma_4''(w_1) = 0, \quad \gamma_4'''(w_1) = 1, \quad \gamma_4(w_2) = 0, \\ &\gamma_5(w_1) = 0, \quad \gamma_5'(w_1) = 0, \quad \gamma_5''(w_1) = 0, \quad \gamma_5'''(w_1) = 0, \quad \gamma_5(w_2) = 1. \end{aligned}$$

Using the recursive formula (2) we can derive the desired transfinite interpolation algorithm for this case (for convenience we reverse the ordering of the projections  $\pi_u, \pi_v, \pi_w$ ):

$$\begin{aligned} (29) \quad f_1 &= \pi_w f = f(u, v, w_1)\gamma_1(w) + \frac{\partial f}{\partial w}(u, v, w_1)\gamma_2(w) + \frac{\partial^2 f}{\partial w^2}(u, v, w_1)\gamma_3(w) \\ &\quad + \frac{\partial^3 f}{\partial w^3}(u, v, w_1)\gamma_4(w) + f(u, v, w_2)\gamma_5(w), \\ f_2 &= f_1 + \pi_v(f - f_1) = f_1 + (f(u, v_1, w) - f_1(u, v_1, w))\beta(v), \\ f_3 &= f_2 + \pi_u(f - f_2) = f_2, \quad \tilde{f} = f_3. \end{aligned}$$

There are thus really only two steps involved in the final algorithm since the third step is trivial. To complete the scheme, we must define the functions  $\beta, \gamma_1, \gamma_2, \gamma_3, \gamma_4, \gamma_5$  in accordance with the conditions in (28). Assuming that  $u_1 = 0, u_2 = 1, v_1 = 0, v_2 = 1, w_1 = 0, w_2 = 1$ , the functions

$$\begin{aligned} \beta(v) &= e^{-K_1 v}, \\ \gamma_1(w) &= 1 - G(w), \quad \gamma_2(w) = w - G(w), \quad \gamma_3(w) = \frac{1}{2}w^2 - \frac{1}{2}G(w), \\ \gamma_4(w) &= \frac{1}{6}w^3 - \frac{1}{6}G(w), \quad \gamma_5(w) = G(w) \end{aligned}$$

with

$$(30) \quad G(w) = \frac{e^{K_2 w} - 1 - K_2 w - (1/2)(K_2 w)^2 - (1/6)(K_2 w)^3}{e^{K_2} - 1 - K_2 - (1/2)K_2^2 - (1/6)K_2^3}$$

$K_1, K_2$  positive constants

have been found to be satisfactory for this particular application. The constant  $K_1$  controls the rate at which the information at the  $v = v_1$  boundary “decays” in the  $v$ -direction while  $K_2$  controls the asymptotic exponential behavior of the mapping near the boundary  $w = w_2$ . In the physical domain, this simply means that  $K_1$  controls the extent of the region where the geometry of the combined plane of symmetry and fuselage surface influences the mesh significantly whereas  $K_2$  controls the extent of the region around the wing where the wing data (including derivatives) influences the mesh and also the asymptotic exponential stretching of the mesh near the outer boundary.

Having defined the complete transfinite interpolation algorithm, we must now consider how to generate the necessary boundary data. Specifying the mapping function on the three boundaries in question is clearly no difficult task, since it is equivalent to defining suitable parametric representations of the corresponding surfaces in  $x, y, z$ -space. The specification of normal derivatives of the mapping function on the wing surface is not, however, quite as simple. As mentioned in § 4, the derivative  $\partial f/\partial w$  defines both the direction of the outgoing mesh lines and the mesh spacing in this direction. The higher derivatives  $\partial^2 f/\partial w^2$  and  $\partial^3 f/\partial w^3$  similarly control the curvature of the outgoing mesh lines and the rate of change of the mesh spacing. Obviously these quantities cannot be generated in a point by point fashion and it is therefore necessary to construct them by combining a small number of basic distributions. For this particular application, it turns out that the derivatives can be generated by very simple formulas, controlled by a small number of parameters. These formulas are based on the behavior of 2D orthogonal mappings around ellipses. Choosing an arbitrary ellipse with ellipse constant  $e$  (Fig. 17), we find that any orthogonal mapping

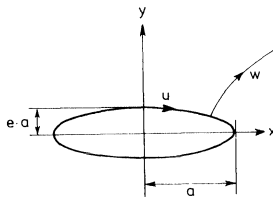


FIG. 17. Schematic of the simple ellipse geometry for which the normal derivatives of orthogonal mapping functions can be computed exactly.

function  $f(u, w) = (x(u, w), y(u, w))$  that maps the infinite region outside this ellipse onto the semi-infinite strip  $u_1 \leq u \leq u_2, 0 \leq w \leq \infty$ , has normal derivatives  $\partial f/\partial w(u, 0), (\partial^2 f/\partial w^2)(u, 0), (\partial^3 f/\partial w^3)(u, 0), \dots$  on the ellipse boundary that can be written as

$$\begin{aligned}
 \frac{\partial x}{\partial w} &= kex, & \frac{\partial y}{\partial w} &= \pm k\sqrt{a^2 - x^2}, \\
 \frac{\partial^2 x}{\partial w^2} &= k^2x, & \frac{\partial^2 y}{\partial w^2} &= \pm k^2e\sqrt{a^2 - x^2}, \\
 \frac{\partial^3 x}{\partial w^3} &= k^3ex, & \frac{\partial^3 y}{\partial w^3} &= \pm k^3\sqrt{a^2 - x^2}, \text{ etc.}
 \end{aligned}
 \tag{31}$$

where  $k$  is an arbitrary scale factor and the plus-minus sign indicates different signs on the upper and lower boundary. This means that for an ellipse the specification of any number of normal derivatives of the mapping function is a simple matter if an exactly orthogonal mesh is desired. However, it is also possible to apply these formulas

to any “approximation” of an ellipse, like for example an airfoil shape, if we do not require exact orthogonality. Furthermore, by varying the parameters  $k$  and  $e$  smoothly between the leading and trailing edges of the airfoil, the behavior of the mesh at these edges can be adapted to the local radius of curvature. This technique has proved to give very satisfactory mesh control for 2D meshes around airfoils. Before discussing how to apply this technique to 3D wings, it is appropriate to demonstrate its validity in the 2D case. Fig. 18 shows a typical mesh around an airfoil generated by a 2D version of the transfinite interpolation method, which uses the extra information provided by the first, second and third normal derivatives of the mapping function at

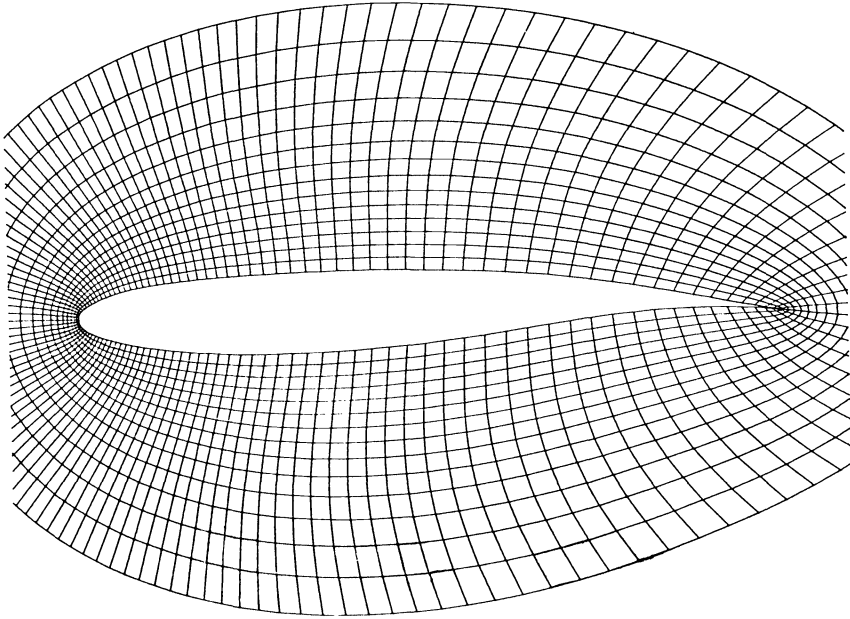


FIG. 18a. Example of 2D mesh around airfoil generated by osculatory transfinite interpolation. The first, second and third normal derivatives of the mapping function are specified on the wing boundary.

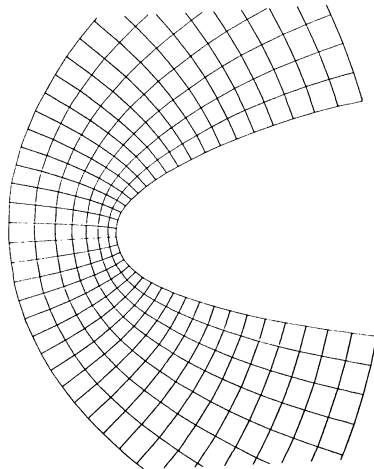


FIG. 18b. Detail of the mesh in Fig. 18a showing the leading edge region.

the airfoil boundary, just as the corresponding 3D method (29). These derivatives were generated by the techniques described above and required no more than five parameters to be defined. To give an idea of the flexibility offered by this technique, various choices of the local behavior of the mesh at the trailing edge of an airfoil are shown in Fig. 19.

Since any chordwise section of a fully three-dimensional wing is an airfoil, the natural way of extending the technique described above to the 3D case is to apply it in a "section by section" manner from the wing root out to the wing tip. By this we mean that for each wing section, the corresponding projection of the normal derivatives of the mapping function are generated by the simple 2D technique for airfoils. The remaining component of these derivatives, essentially the spanwise component, must then be determined by some additional conditions. Over the major part of the wing (all but the outermost tip region) this spanwise component can be set to zero, and it is only in the tip region where a more refined method is needed to specify both the chordwise and the spanwise components of the normal derivatives of the mapping

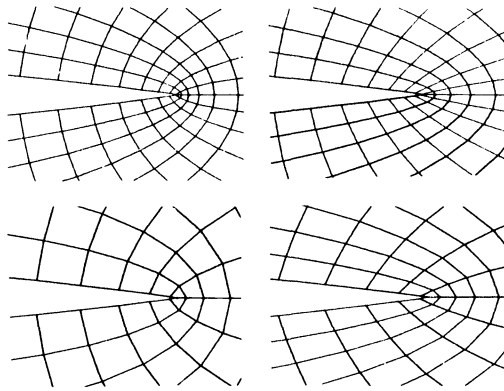


FIG. 19. Examples of meshes at the trailing edge of an airfoil showing the flexibility of the method.

function. Since the  $O-O$  mapping type is such that the mesh wraps around the wing tip also, it is desirable to obtain a mesh that is nearly orthogonal at the tip. This can be achieved to a high degree by again using the idea of copying the behavior of an orthogonal mesh around an ellipse, i.e. by adjusting the chordwise and spanwise components of the normal derivatives according to (31). Just as in the 2D case, the overall technique gives very satisfactory mesh control for a 3D wing with a minimum of parameters to specify.

Before showing some examples of 3D meshes generated by the present method, some words should be said about the outer boundary and the combined plane of symmetry and fuselage. As stated before, the specification of the mapping function at the top of the computational box, the surface  $w = w_2$ , is equivalent to the construction of a parametric representation of the outer boundary in  $x, y, z$ -space. This can be done either by direct algebraic construction or by interpolation. In this work, transfinite interpolation was chosen, i.e. the outer boundary was generated by interpolating between a set of curves in  $x, y, z$ -space. Since the outer boundary surface mesh must have the same topological structure as the wing surface mesh, it follows that it must have two mesh singularities of parabolic type. Using an osculatory type of transfinite interpolation like scheme (19), it is possible to generate a very smooth surface mesh with exactly the "correct" type of singularity. An oblique view of such a surface mesh is shown in Fig. 20. The other surface to be defined, the combined plane of symmetry

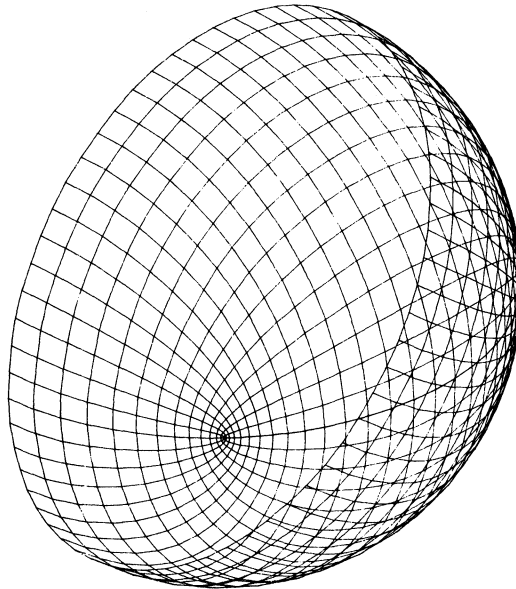


FIG. 20. Oblique view of the outer boundary surface mesh showing one of the two parabolic singularities.

and fuselage, presents some minor difficulties that should be remarked upon. In the wing alone case, when the wing root coincides with the plane of symmetry, the simplest way of obtaining a suitable surface mesh for this plane is to “project” the innermost mesh surface generated by the first step in the interpolation scheme (29) onto the plane of symmetry. In terms of the mapping function  $f(u, v, w)$ , this amounts to defining  $f(u, v_1, w)$  as the geometric projection of  $f_1(u, v_1, w)$ , where  $f_1 = \pi_w f$ , onto the plane of symmetry. The advantage of this method is that it simplifies the specification of the function  $f(u, v_1, w)$ . Since the normal derivatives of  $f$  at  $w = w_1$  in practice are specified such that the function  $f_1(u, v_1, w)$  already coincides with the plane of symmetry, the second step of the interpolation scheme (29) actually drops out altogether. In other words, the wing alone case only needs the first step of scheme (29) if the normal derivatives of  $f$  at the wing surface are specified in the right way. In the wing-fuselage case, the combined plane of symmetry and fuselage surface must be specified in some practical way. Perhaps the simplest method is to use the same geometric projection idea as for the wing alone case, i.e. to project the intermediate function  $f_1(u, v_1, w)$  onto the combined plane of symmetry and fuselage surface, using some kind of oblique geometric projection, and defining this projection as the desired function  $f(u, v_1, w)$ . The reason for using an oblique geometric projection method is to obtain a concentration of mesh points around the fuselage. It is clear that the resulting surface mesh conforms to the combined plane of symmetry and fuselage surface but it is *not* aligned with the curve where the fuselage intersects the plane of symmetry. This type of surface mesh is consistent with the idea of representing the fuselage as a “bump” in the plane of symmetry. To obtain a more accurate fuselage representation, the surface mesh obviously must be aligned with the intersection curve, i.e. this curve must coincide with a mesh line. In this work, the simpler alternative was chosen since it was judged to give enough resolution for the problem at hand.

*Computed examples.* The first computed example is an  $O-O$  type mesh for the Onera M6 wing without any fuselage. An oblique view of three constant- $v$  mesh

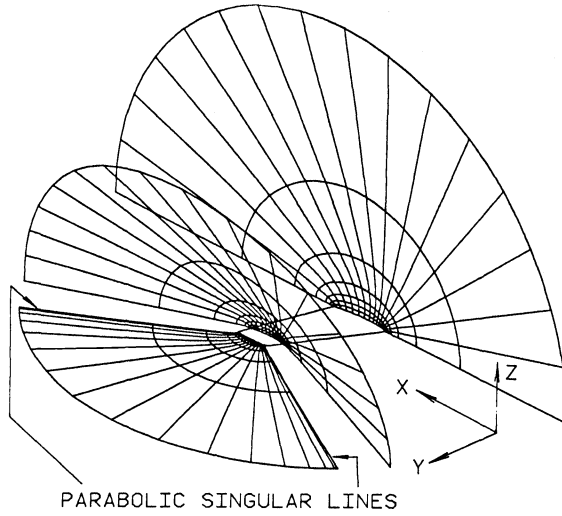


FIG. 21. Oblique view of three chordwise (constant- $v$ ) mesh surfaces of an  $O-O$  mesh for the Onera M6 wing. Only upper halves are shown.

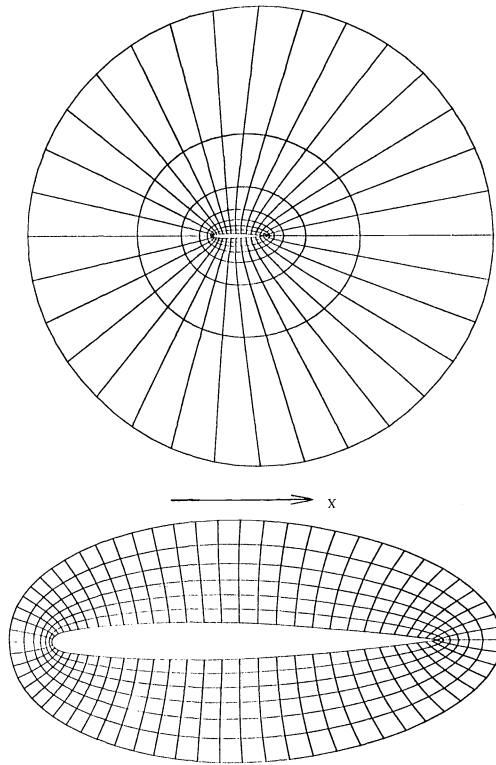


FIG. 22. Planar view of the innermost chordwise mesh surface ( $v = v_1$ ) of the same mesh as in Fig. 21. The bottom figure shows a detail of the top figure.

surfaces (Fig. 21) shows the characteristic structure of this mapping type in the physical domain. The innermost mesh surface ( $v = v_1$ ) lies in the plane of symmetry, the outermost mesh surface ( $v = v_2$ ) lies in the horizontal wing plane and the intermediate mesh surface indicates how these surfaces vary in between. Considering that only the upper halves of the surfaces are shown, it is clear that as  $v$  approaches the upper limit  $v_2$ , the corresponding mesh surface “collapses” into a coordinate cut outside the wing tip. The resulting parabolic singular lines at the edges of this coordinate cut are indicated in the figure. Fig. 22 shows a planar view of the innermost surface ( $v = v_1$ ) while Fig. 23 presents a planar view of a constant- $u$  surface. The latter figure shows

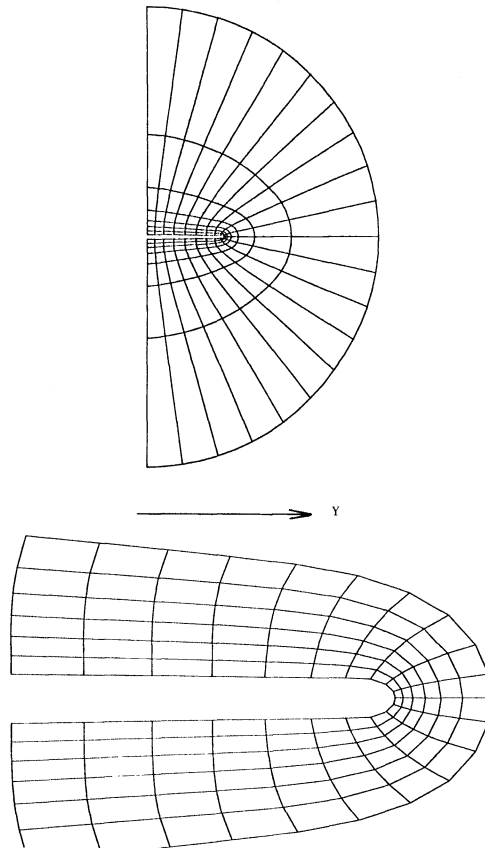


FIG. 23. Planar view of a spanwise (constant- $u$ ) mesh surface of the same mesh as in Fig. 21. The bottom figure shows a detail of the top figure.

the behavior of the mesh around the wing tip. The surface mesh on the wing as seen from above is presented in Fig. 24, together with an oblique and planar view of the tip region that clearly shows the parabolic singularity at the leading edge/tip corner.

The next example is an  $O-O$  type mesh for a wing-body configuration. A planar view from above of some mesh surfaces is shown in Fig. 25. The surface mesh on the fuselage is here visible and it can be seen that this mesh is not aligned with the intersection between the fuselage and plane of symmetry. A planar view of a constant- $u$  surface (Fig. 26) shows how the mesh conforms to the combined plane of symmetry and fuselage surface.

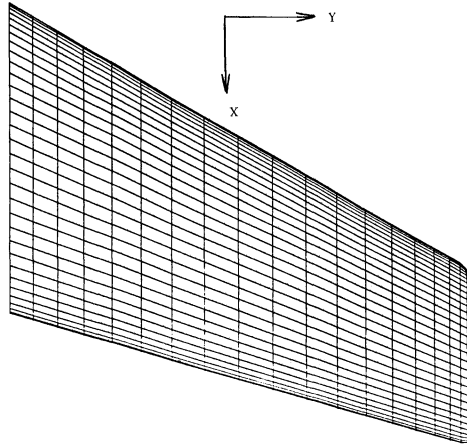


FIG. 24a. Vertical view of wing surface mesh for the same mesh as in Fig. 21.

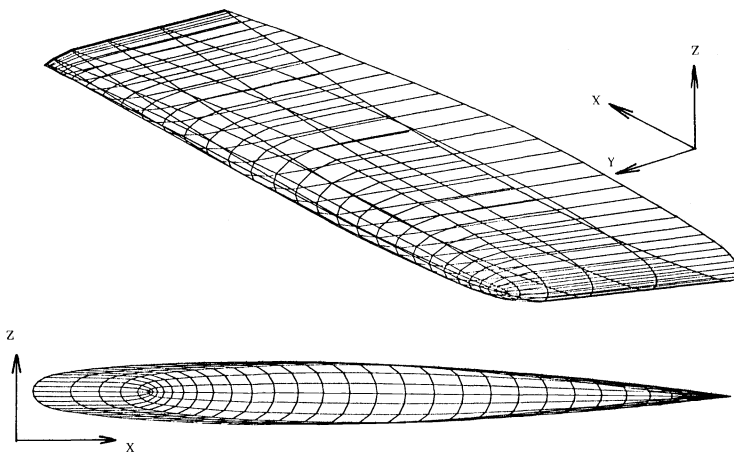


FIG. 24b. Two different views of the tip region of the wing surface mesh in Fig. 24a. The parabolic singular point at the leading edge/tip corner is clearly visible.

In order to demonstrate the applicability of the present mesh generation method to other cases, an  $O-O$  type mesh for a delta wing is shown in Fig. 27. The structure of the mapping in this case is slightly different from that in the ordinary wing case, but it is similar in that the entire wing surface (both upper and lower surface) is mapped to the bottom surface of the computational box and the outer boundary is mapped to the top surface. The difference is that the plane of symmetry is here mapped to two opposing side surfaces instead of just one. In principle this would generate two parabolic singular lines in the mesh, but due to the triangular planform, one singular line is of polar type instead. This is clearly visible in the figure, where a number of constant- $v$  mesh surfaces are shown. Just as for the ordinary wing, the transfinite interpolation scheme for the delta wing uses normal derivatives of the mapping function on the wing surface to control the mesh, and these derivatives are generated by the same technique.



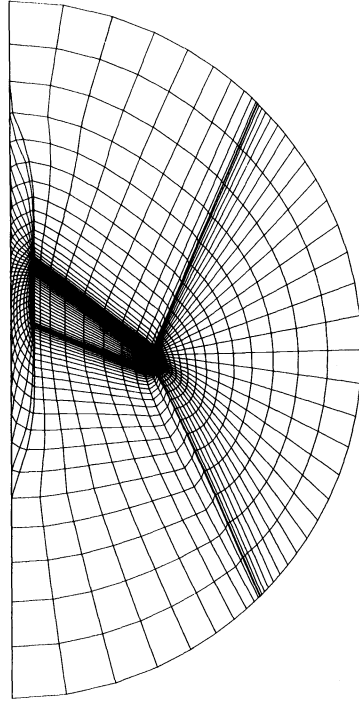


FIG. 25. Vertical view of certain mesh surfaces of an *O-O* mesh for a wing-body configuration.

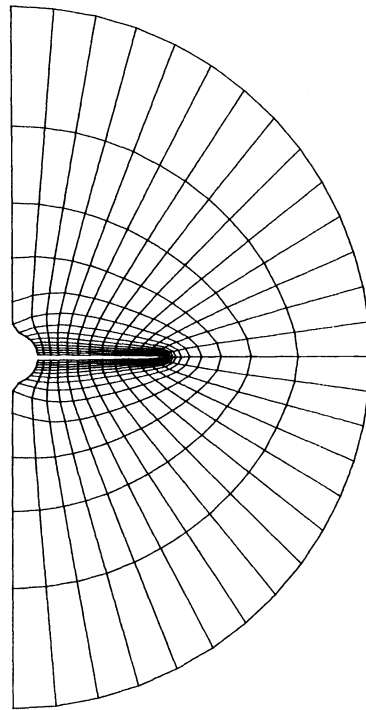


FIG. 26. Planar view of a spanwise (constant-*u*) mesh surface of the same mesh as in Fig. 25.

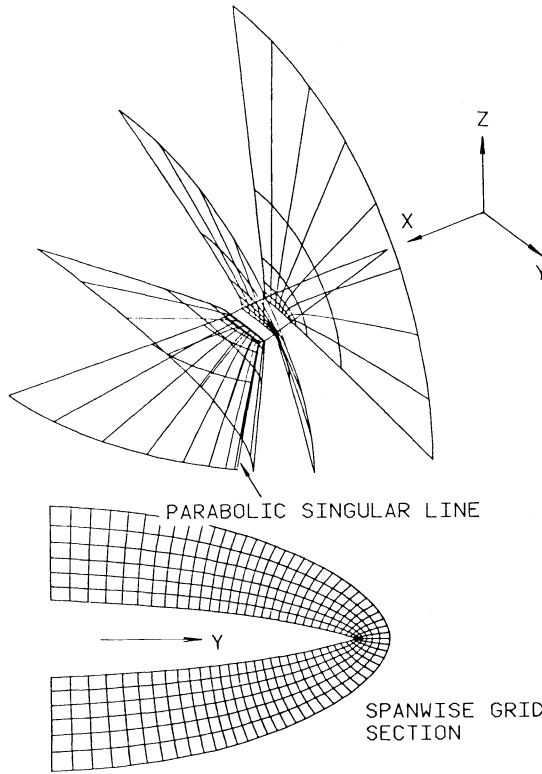


FIG. 27a. Oblique view of spanwise mesh surfaces of an O-O mesh for a delta wing. Also shown is a detail of one of these mesh surfaces.

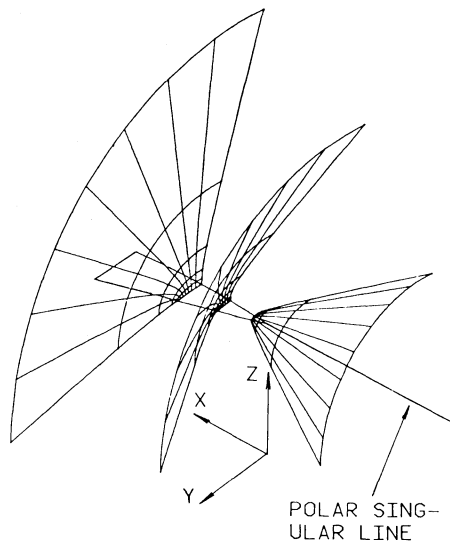


FIG. 27b. Oblique view of the upstream part of the same mesh as in Fig. 27a.

**6. Concluding remarks.** The transfinite interpolation method is clearly an extremely flexible tool for generating meshes in both two and three space dimensions and the particular applications described in this work are to be seen as examples of what can be accomplished with this tool. As far as the theory of the method is concerned, there is not much to be added to what has already been said, except for a few comments about "consistency of boundary data." In the derivation of the transfinite interpolation scheme, the given boundary data was assumed to be consistent, i.e. the specification of the function to be interpolated (including any normal derivatives) on the boundaries was assumed to be such that no contradictions occurred at the intersections between different boundaries. This is an important condition because the recursive formulation of the transfinite interpolation method is independent of the order in which the successive univariate interpolation steps are taken *only if this condition holds*. In connection with the theory it should also be mentioned that the omission of any error analysis, which is usually considered to be an integral part of any interpolation method, is deliberate and motivated by the fact that for mesh generation, the interpolant (=resulting mesh) is not judged by looking at some sort of error but rather by looking at some relevant mesh properties. From this viewpoint, the transfinite interpolation method is only the *means* of achieving the desired mesh whereas the *control* of the generated mesh lies almost exclusively in the generation of boundary data. A good example of this is the described 3D wing case, where the generated normal derivatives of the mapping function on the wing surface provide an excellent mesh control in the vicinity of the wing. This case is also typical in a practical programming sense; the actual coding of the transfinite interpolation scheme does not take more than 25 FORTRAN statements whereas the complete boundary specifications including normal derivatives takes of the order of a thousand statements.

*Software distribution.* A user-oriented computer program of this 3D mesh generation method is commercially available for public use on a number of different computers, among others several Control Data CYBER 205 systems throughout the world including the CYBERNET machines in Minneapolis and Paris. For more information contact Mr. D. Astertun, Control Data, Box 7, 163 93 Stockholm, Sweden, Tel. (08) 752 00 20.

#### REFERENCES

- [1] *Numerical grid generation techniques*, NASA CP 2166, 1980.
- [2] J. F. THOMPSON AND Z. U. A. WARSİ, *Boundary-fitted coordinate systems for numerical solution of partial differential equations—a review*, J. Comput. Phys., 47 (1982), pp. 1–108.
- [3] S. A. COONS, *Surfaces for computer aided design of space forms*, Project MAC, Design Div., Dept. of Mech. Engineering, MIT, 1964, Revised to MAC-TR-41, 1967.
- [4] W. J. GORDON, *Blending-function methods of bivariate and multivariate interpolation and approximation*, SIAM J. Numer. Anal., 8 (1971), pp. 158–177.
- [5] W. J. GORDON AND C. A. HALL, *Construction of curvilinear coordinate systems and applications to mesh generation*, Int. J. Numer. Method Engrg., 7 (1973), pp. 461–477.
- [6] M. A. GERHARD, OASIS, *A general purpose mesh generator for finite element codes*, M-101, Lawrence Livermore Lab., Univ. California, 1979.
- [7] P. G. ANDERSON AND L. W. SPRADLEY, *Finite difference grid generation by multivariate blending function interpolation*, Numerical Grid Generation Techniques, NASA CP 2166, 1980.
- [8] L. W. SPRADLEY, J. F. STALNAKER AND A. W. RATLIFF, *Solution of the three-dimensional Navier-Stokes equations on a vector processor*, AIAA J., 19 (1981), pp. 1302–1308.
- [9] L. E. ERIKSSON, *Three-dimensional spline-generated coordinate transformations for grids around wing-body configurations*, Numerical Grid Generation Techniques, NASA CP 2166, 1980.

- [10] L. E. ERIKSSON, *Generation of boundary-conforming grids around wing-body configurations using transfinite interpolation*, AIAA J., 20 (1982), pp. 1313-1320.
- [11] L. E. ERIKSSON AND A. W. RIZZI, *Computation of vortex flow around wings using the Euler equations*, Proc. the Fourth GAMM-Conference on Numerical Methods in Fluid Mechanics, H. Viviand, ed., Vieweg Verlag, Paris, Oct. 1981.
- [12] A. W. RIZZI, *Damped Euler-equation method to compute transonic flow around wing-body combinations*, AIAA J., 20 (1982), pp. 1321-1328.
- [13] L. E. ERIKSSON, *A study of mesh singularities and their effects on numerical errors*, FFA Tech. Note TN 1984-10, Stockholm, 1984.

## A COMPARISON OF ELLIPTIC SOLVERS FOR GENERAL TWO-DIMENSIONAL REGIONS\*

TONY F. CHAN† AND FAISAL SAIED†

**Abstract.** We present results of some numerical experiments performed to compare the relative efficiency and storage requirements of some elliptic solvers on general two-dimensional regions. The techniques employed by the solvers included: capacitance matrix techniques, sparse matrix techniques, multi-grid techniques and preconditioned conjugate gradient techniques. Special emphasis is placed on the many right-hand side case, as motivated by applications to time simulation studies.

**Key words.** Poisson equation, irregular domains, multiple right-hand sides, sparse matrix methods, capacitance matrix methods, pre-conditioned conjugate gradient methods, multi-grid methods

**1. Introduction.** The advent of fast elliptic solvers [24], [25] in the past decade has tremendously advanced the state of the art of the field of numerical simulation of dynamical physical systems. One of the major applications has been in computational fluid dynamics. Here, various versions of the Navier-Stokes equation are solved and at each time step an elliptic system governing either the pressure field or the stream function has to be solved, which often constitute a major part of the computational cost. Unfortunately, these fast solvers have a rather limited domain of applicability, one of which is the restriction to rectangular domains. With the success of using fast solvers becoming widespread, many users are starting to attempt more realistic simulations on more general domains. Since fast solvers cannot be used directly on irregular domains, alternative methods have to be used. One method that has worked very well is the Capacitance Matrix (CM) Method [21], which embeds the irregular domain within a rectangle and exploits the speed of a fast solver on the latter. However, the work involved is substantially more than in the rectangular case, thus raising the possibility that other alternative methods, which may not have been competitive on rectangular domains, may be competitive in this new situation. Among these are Sparse Matrix (SM) methods [8], [11], [14], Multi-grid (MG) methods [2], [3], [13] and Preconditioned Conjugate Gradient (PCG) methods [4], [12], [17], [18]. In this paper, we shall report the results of some numerical experiments designed to compare the relative performance of these competing methods in terms of computing time and storage requirements. Comparisons of elliptic solvers have been carried out before, primarily on rectangular domains and for one-shot problems [23]. In our study, special emphasis has been placed on irregular domains and situations where the same problem has to be solved many times with different right-hand sides, such as in a time simulation environment. To the best of our knowledge, such a comparison has not been published before.

Since the efficiency of these methods is rather sensitive to implementation details, we have employed publicly available computer packages embodying each of the above mentioned numerical techniques in carrying out these experiments. Our criteria for choosing the particular codes have been that the code (or the technique behind the code) be recognized as representative and competitive, and the ease of obtaining the

---

\* Received by the editors June 13, 1983, and in revised form March 21, 1984.

† Computer Science Department, Yale University, New Haven, Connecticut 06520. This work was supported in part by the Office of Naval Research under grant N00014-80-C-0076 under a subcontract from Florida State University and in part by the U.S. Department of Energy under grant DE-AC02-81ER 10996.

code. Only finite difference discretizations have been considered. We have not considered the issue of vectorization for machines like the Cray-1. Due to implementation details, variations in machine architecture and compiler features, there is always the danger of drawing conclusions that are too general from an empirical comparison like this one. Nevertheless, we hope that such comparisons can still serve a useful purpose by giving indications as to the relative performances of competing algorithms.

In § 2, we describe the model problem and the discretizations used in the numerical experiments. The next four sections (3–6) then describe briefly the essential features of the four competing methods and their potential trade-offs in terms of speed and storage. Section 7 summarizes the theoretical space and time complexity of the various methods and serves as a basis for comparing them. The results of the numerical experiments are presented in § 8, and we try to draw some conclusions in § 9.

**2. The model problem.** We consider the problem

$$(1) \quad \begin{aligned} \mathcal{L}u &= f && \text{in the domain } \Omega, \\ u &= g && \text{on the boundary of } \Omega, \end{aligned}$$

where  $\mathcal{L}$  is a linear, two-dimensional elliptic operator, and  $F$  and  $G$  are given functions.

Depending on the particular solver, the problem class has to be appropriately restricted. For the CM method,  $\mathcal{L}$  has to be separable, so that a fast solver can be used on a rectangle. The MG package we used restricts  $\mathcal{L}$  to be the Helmholtz operator. For the SM and PCG methods,  $\mathcal{L}$  can be rather general. Except for the MG solver, the domain can also be quite general, e.g. not simply-connected. Without loss of generality, we assume that  $\Omega$  can be inscribed in the unit square,  $R$ , on which we impose a uniform grid with  $n$  intervals on each side. The mesh-width is thus equal to  $1/n$ . We denote the number of grid points inside  $\Omega$  by  $N$ . An inner grid point is called *irregular*, if one or more of its neighbors is not an interior point. The number of irregular points is denoted by  $p$ .

To discretize (1), we use the standard 5-point centered finite difference discretization for the interior grid points. For the irregular points, we normally use the Shortley–Weller formula, which has second order accuracy, and is also used by the CM and MG packages. However, since our test problems all involve self-adjoint operators which give rise to symmetric discretization matrices, the use of the Shortley–Weller formula sometimes destroys this symmetry. For some of the methods which crucially depend on this symmetry, namely the SM and the PCG method, we also use a less accurate boundary approximation to preserve the symmetry in order to fully exploit the power of the methods. This boundary approximation amounts to modifying the boundary of  $\Omega$  so that it lies on the grid lines.

With the above discretization, we obtain the following  $N$  by  $N$  system of linear equations for the grid-valued vector  $u_h$ :

$$(2) \quad A_h u_h = b.$$

We note that if the Shortley–Weller formula is used, all four methods solve the same linear system (2), although with different accuracy.

**3. Sparse Gaussian elimination.** In this class of methods, the linear system (2) obtained by discretizing the partial differential equation (1) is solved directly using Gaussian elimination. This approach was not feasible for large problems until the development of special sparse matrix techniques that do not store or operate on the zeros of the matrix [8], [14]. The usual factor and solve phases are preceded by a

preprocessing phase which operates on the sparsity pattern of  $A_h$ . In this phase, the rows and columns of  $A_h$  are permuted, usually with some heuristic strategies, to reduce the fill-in that arises in the subsequent factorization. Thus we have:

$$(3) \quad PA_h Q = LU$$

where  $L$  is lower triangular and  $U$  is unit upper triangular. The solve phase then consists of two back-substitutions:

$$(4) \quad Ly = Pb, \quad Uz = y, \quad u_h = Qz.$$

The factorization phase is usually more time consuming than the solve phase. Moreover, even with a good ordering strategy, the amount of fill-in may still be substantial, thus increasing the storage overhead. However, if several systems with the same matrix have to be solved, the  $LU$  factors need only be computed once. This is the most important advantage of this class of methods. Note that the factorization phase can also take advantage of systems with the same sparsity pattern but different numerical entries, such as in elliptic systems with time dependent or nonlinear coefficients.

The sparse matrix program that we tested is the Yale Sparse Matrix Package (YSMP) [11]. It uses the minimum degree ordering algorithm to symmetrically reorder the rows and columns of  $A$  and assumes that no pivoting is needed for maintaining numerical stability. It has one driver (SDRV) for symmetric systems and three drivers (TDRV, NDRV and CDRV) for nonsymmetric systems. The nonsymmetric drivers offer different trade-offs between storage and work. We give the results for NDRV and CDRV, both of which are more suitable for the many right-hand side cases. CDRV has a smaller storage requirement than NDRV but needs more computation time because of a compressed storage format. The symmetric driver SDRV needs about half as much storage and time for the preprocessing and factoring as the nonsymmetric ones. The speeds of the solve phases are roughly the same.

**4. Capacitance matrix methods.** Capacitance matrix methods were developed in an attempt to exploit the advantages of fast elliptic solvers for solving problems on nonrectangular domains [21]. In this approach, the domain  $\Omega$  is embedded in a rectangle  $R$  with a uniform mesh. Let  $B$  be the matrix that represents the discretization of the elliptic operator  $\mathcal{L}$  on  $R$  and let  $A$  be the matrix corresponding to extending  $A_h$  to  $R$  with  $u$  and  $b$  the corresponding extended solution and right-hand side. Then it can easily be verified that  $A$  differs from  $B$  by a low rank correction matrix corresponding to the irregular points on the boundary:

$$(5) \quad A = B + WZ^T,$$

where the matrix  $W$  represents an extension operator that maps  $\Omega$  onto  $R$  and the matrix  $Z$  can be regarded as a compact representation of  $A - B$ . Note that both  $W$  and  $Z$  are sparse and thus their applications to grid-valued vectors are relatively cheap. The capacitance matrix technique can be viewed as applying the Sherman-Morrison-Woodbury formula to (5):

$$(6) \quad A^{-1} = B^{-1}[I + W(I + Z^T B^{-1} W)^{-1} Z^T B^{-1}].$$

The solution  $u = A^{-1}b$  can then be computed by two solves with the matrix  $B$  (e.g. by a fast solver for rectangular regions) and one solve with the  $p$  by  $p$  capacitance matrix:

$$(7) \quad C = I + Z^T B^{-1} W.$$

For the many right-hand side case, the capacitance matrix  $C$  can be computed and factored once and its  $LU$  factors stored for repeated back-substitutions. Note that  $C$

is dense and nonsymmetric in general. However, since  $p$  is usually much smaller than  $N$ , factoring and solving with  $C$  is usually much cheaper than doing the same things with  $A_h$ . Thus, the efficiency of the capacitance matrix method derives from exploiting the lower dimensionality of the boundary of the irregular region and the speed of a fast solver on a rectangular region.

The capacitance matrix program that we tested is a revised version of the routine HELMIT written by Proskurowski [22]. This code solves the Helmholtz equation with constant coefficients and can handle both Dirichlet and Neumann boundary conditions. The storage requirement is  $p^2 + n^2 + O(p)$ . The underlying fast solver is based on FFTs which restrict the mesh width to powers of two. The package includes a subroutine HLMHLZ, which solves the capacitance matrix equation by applying a conjugate gradient method to the normal equation for  $C$ . This has a smaller storage requirement than HELMIT (since  $C$  is not stored), but requires two fast solves per conjugate gradient iteration. Our numerical experiments showed that it is not competitive for the multiple right-hand sides case and therefore the results will not be presented. Recent work on preconditioning the capacitance matrix may make these methods more competitive for the multiple right-hand sides [27].

**5. Multi-grid methods.** This is the class of optimal order methods described in [3] that are based on relaxation sweeps on a hierarchy of grids. Relaxation methods for solving linear systems of equations, due to their “local” nature, can rapidly damp out errors whose frequencies are high compared to the mesh width, but slow down after that. Multi-grid techniques can be viewed as using a hierarchy of coarser grids to smooth out the lower frequency errors, and derive their efficiency from the fact that smoothing relaxation sweeps on these coarser grids are much cheaper to carry out than on the fine grid. For rather general elliptic problems, these methods have been proven to be “optimal order”, that is, they take only  $O(N)$  arithmetic operations and storage to reduce the error to truncation error level [2], [3], [7], [16].

Although multi-grid methods have very nice theoretical properties, their implementations are by no means straightforward, as witnessed by the small number of general purpose multi-grid codes that are available. Our tests were carried out with the MG01 code from the G.M.D. in Germany [26]. This package solves the Dirichlet problem for the Helmholtz equation on a domain which is specified by two functions  $f_l(x)$  and  $f_h(x)$  that define the lower and upper boundaries, respectively. As a result, it cannot handle domains with “holes” in them. The method implemented is a fixed version of the Full Multi-Grid (FMG) method which starts from the coarsest grid and works its way up to the finest grid, using the solution obtained from a coarse grid as the initial guess for the next finer grid. Theoretically, it computes a solution that is accurate to truncation error level. On the coarsest grid, MG01 uses Gauss–Seidel rather than a direct solver. The program allows the user to adapt to the difficulty of a given problem by providing a complexity parameter (ITYP), which can take on values from 1 to 4 (the higher values are recommended for problems with highly oscillatory solutions or very irregular domains). ITYP controls the number of relaxation sweeps performed on each level after the ( $h \rightarrow 2h$ ) and ( $2h \rightarrow h$ ) transfers. The smoothing sweep used is Gauss–Seidel with red/black ordering. For the grid transfers, linear interpolation is used for the ( $2h \rightarrow h$ ) transfer, except when a finer grid is visited for the first time, in which case cubic interpolation is used, and half residual weighting is used for the ( $h \rightarrow 2h$ ) transfer with the following operator:

$$(8) \quad I_h^{2h} = \frac{1}{8} \begin{pmatrix} 0 & 1 & 0 \\ 1 & 4 & 1 \\ 0 & 1 & 0 \end{pmatrix}.$$



The package can be run either in the Full Approximation Storage (FAS) mode or in the Correction Storage (CS) mode. Our tests were conducted using the CS mode because it is applicable to linear problems and is faster.

It should also be pointed out that the MG01 code does not offer any savings for multiple right hand sides. Moreover multi-grid methods are of optimal order only when computing the solution to truncation error level. Foerster, Stüben and Trottenberg [13] found that the solution  $v_h$  computed by MG01 satisfies

$$(9) \quad \|v_h - u^*\|_2 \leq k \|u_h - u^*\|_2 \quad \text{with } 1 < k \leq 2,$$

where  $u^*$  is the exact solution of the partial differential equation (1),  $u_h$  is the exact solution of the discretized linear problem (2) and  $k$  is independent of  $h$ . On the other hand, the results of Dendy and Hyman [6] indicate that multi-grid methods need substantially more (by a factor of 4 or 5) time to compute a solution with a small residual (e.g. discrete 2-norm  $\leq 10^{-6}$ ). Whether an accuracy at the level of truncation error level is enough depends on the particular application. In time simulations, a good predictor is often available and a moderate accuracy in the corrector *may* be good enough to control the stability of the time integration. The optimal stopping criterion iteration remains an open question.

**6. Preconditioned conjugate gradient methods.** Conjugate gradient methods, originally designed as a direct method for symmetric positive definite linear systems [17], have been resurrected in the last ten years as a fast iterative method for solving large sparse systems, especially when used with preconditioning techniques ([4], [12], [18], and references therein).

The basic Conjugate Gradient (CG) method solves a symmetric positive-definite system,  $Ax = b$ , iteratively, and is optimal in the sense that it minimizes the  $A$ -norm of the error in the  $i$ th Krylov subspace at the  $i$ th iteration. The sparsity of the matrix is exploited by only requiring a routine for computing a matrix-vector product. The performance of the CG method is known to depend, often sensitively, on the symmetry, positive definiteness and the distribution of the eigenvalues of the coefficient matrix,  $A$ . In fact, until recently [5], [12], [18], [28], relatively little was known about applications to nonsymmetric problems. For one-shot large elliptic problems, the method has proven to be competitive with direct methods. However, as is true for many other iterative methods, one potential drawback of the CG method is its inability to take advantage of multiple right hand sides beyond saving the preconditioning matrix (or its factors). Although some information about the Krylov subspace can be saved and block methods have been devised to exploit multiple right-hand sides [20], it is not easy to exploit the information gathered from previous iterations, since, in a time simulation, not all the right-hand sides are known simultaneously.

The code package that we have chosen for the CG method is the PCGPACK produced at Yale (Elman, [12])). It handles general nonsymmetric systems and contains a large number of the known competitive CG algorithms and preconditioning techniques. We use the basic Conjugate Gradient method. Only the symmetric discretization of the partial differential equation was used, since this case fully exploits the power of the CG method. The results obtained can be regarded as lower bounds for the work in the general nonsymmetric case.

We shall present results computed by two preconditioning techniques, namely the Dupont-Kendall-Rachford (DKR) preconditioning [9], [15], which is related to the class of Incomplete Cholesky factorization methods [19] and the Symmetric Successive Over-Relaxation (SSOR) preconditioning technique [1], which requires less storage.

As tests reported in [12] show, these are representative of the most successful preconditionings for elliptic problems. The storage requirement of the PCG method is essentially  $O(N)$  with a relatively small constant ( $< 10$ ).

Since the PCG method is an iterative method, the stopping criteria has to be specified. In our tests, we stop the iterations when the residual is reduced by a factor  $\epsilon$ . In the numerical results that we present, we have chosen  $\epsilon$  to be  $10^{-6}$ . We also give the times the PCG method takes to achieve the same relative residuals as the MG algorithm.

**7. Complexity.** In Table 7.1, we summarize the complexity of the four competing methods in terms of the number of arithmetic operations and storage requirements. Although all the estimates are asymptotic, they do serve as a basis for comparing the algorithms.

TABLE 7.1  
Theoretical complexity of the algorithms.

	SM (*)	CM	MG	PCG
Factor	$O(n^3)$	$p^3/3 + O(n^2 \log n)$	none	$O(n^2)$
New RHS	$O(n^2 \log n)$	$O(n^2 \log n) + p^2$	$O(n^2)$	$O(n^{2.5} \log(1/\epsilon))$ $O(n^{2.5} \log(n))$ , if $\epsilon = O(n^{-2})$
Storage	$O(n^2 \log n)$	$p^2 + n^2$	$O(n^2)$	$O(n^2)$

$p$  = number of irregular points.  
 $n$  = 1/mesh-width.

$\epsilon$  = reduction factor in error.  
(\*)  $\Rightarrow$  with nested dissection.

In deriving these results, we have assumed that the number of unknown grid points,  $N$  is  $O(n^2)$ . The estimates for the SM method are based on the nested-dissection ordering [14]. The ordering routine in YSMP produces a minimum degree ordering but it is not known whether it achieves the nested-dissection storage estimate,  $O(n^2 \log(n))$ . The estimates for the CM method are based on a direct  $LU$  factorization of the capacitance matrix  $C$ . The estimates for the MG method are for reducing the error to truncation-error level [13]. The new right-hand side estimate for the PCG method is based on well-known estimates for model elliptic problems (e.g. Poisson's equation on a rectangle) [4]. The parameter  $\epsilon$  is the reduction factor in the residual of the iterates used in the stopping criterion. The results for stopping at truncation-error level are obtained by setting  $\epsilon = O(n^{-2})$ .

From Table 7.1, one can see that the MG method has optimal asymptotic complexity in both storage and work. However, the work estimate only holds if one solves to truncation-error level, which may not be adequate in a time-simulation context. Besides, practical implementations of MG are rather complicated, resulting in large constants in the operation counts. It is also not clear how to take advantage of multiple right hand sides.

For the PCG method, the constants involved in the complexity bounds are usually quite small. However, it is not easy to take advantage of multiple right-hand sides other than by using previous direction vectors [20] or by reusing the preconditioner, which unfortunately is equivalent to only a few iterations and is usually not a dominant part of the total work.

The computational costs for the SM and the CM methods are also equal in complexity, whereas the storage for the SM method is higher. The actual cost of the

CM method is dependent on  $p$ , the number of irregular boundary points. In this study, we have assumed a rather general domain, implying that  $p \approx cn$  with  $c \approx 4$ . With this value of  $p$ , we see that the part of the complexity for the CM method that depends on  $p$  (i.e. the part that relates to the capacitance matrix) can be dominant over the FFT part in the preprocessing stage. Thus, the total work for CM method may be higher than that for the SM method.

As we shall see in the next section, the above observations are borne out by the results of the numerical experiments.

**8. Numerical experiments.** In this section, we present the storage requirements and CPU times of the methods tested. The test problem was

$$-\Delta u = f \text{ in } \Omega \quad \text{and} \quad u = 0 \text{ on the boundary of } \Omega$$

with  $f \equiv 1$ .

The tests were conducted in single precision (27 bit mantissa) using optimized Fortran on a DEC-2060. All times are in milliseconds. The three domains used in the tests are shown in Fig. 8.1.

The tests were run with a uniform grid imposed on the unit square that enclosed the domains with  $n = 16, 32, 64$ . For the methods and domains for which the storage was adequate, the problem was also solved with  $n = 128$ . In Table 8.1, the corresponding

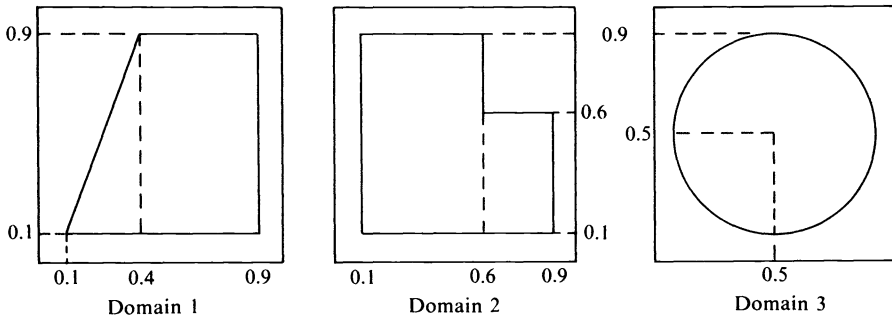


FIG. 8.1. Computational domains.

TABLE 8.1  
Problem size.

Domain		$n$			
		16	32	64	128
1	$N$	136	511	2113	8594
	$P$	43	87	181	369
2	$N$	144	544	2240	9088
	$P$	47	95	199	407
3	$N$	129	509	2061	8245
	$P$	32	72	144	288

$N$  = Total number of grid points in domain.

$P$  = Number of irregular points in domain (points with one or more neighbors outside domain).

values of  $N$  and  $p$  are tabulated for these domains. Note that indeed  $N \approx O(n^2)$  and  $p \approx O(n)$ . Figs. 8.2, 8.3, and 8.4 give the storage requirements for the various methods for each domain, Figs. 8.6, 8.7 and 8.8 give the corresponding preprocessing times and Figs. 8.9, 8.10 and 8.11 give the corresponding times to solve for new right-hand sides. In these plots, we have plotted storage and time *per inner grid point* ( $N$ ) versus  $n$ .

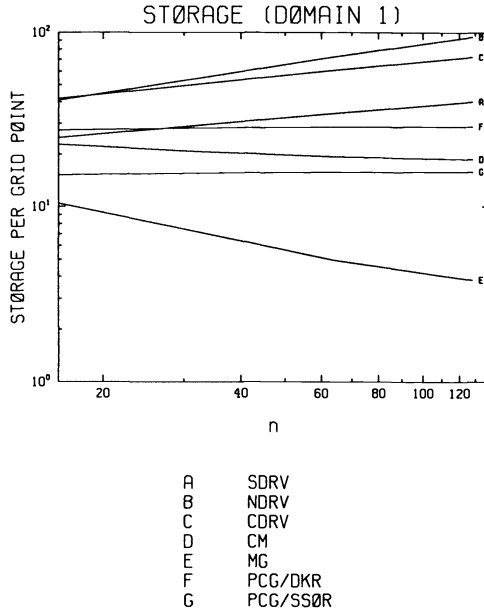


FIG. 8.2. Domain 1. Storage per grid point vs.  $n$ .

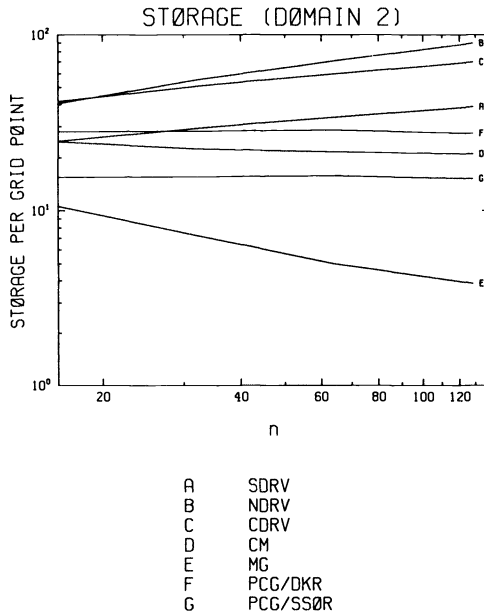


FIG. 8.3. Domain 2. Storage per grid point vs.  $n$ .

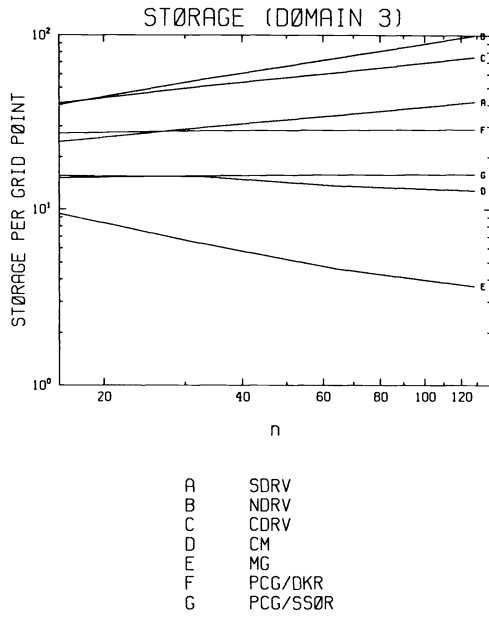


FIG. 8.4. Domain 3. Storage per grid point vs. n.

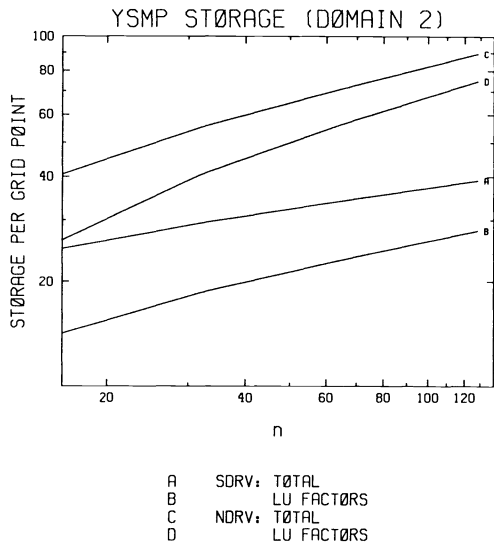


FIG. 8.5. YSMP: Breakdown of storage.

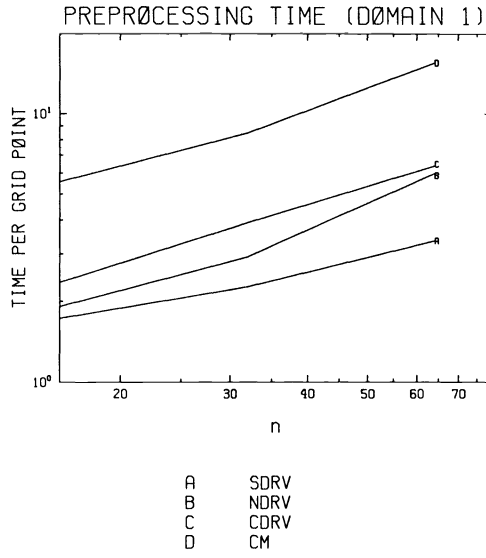


FIG. 8.6. Domain 1. Preprocessing time vs. n.

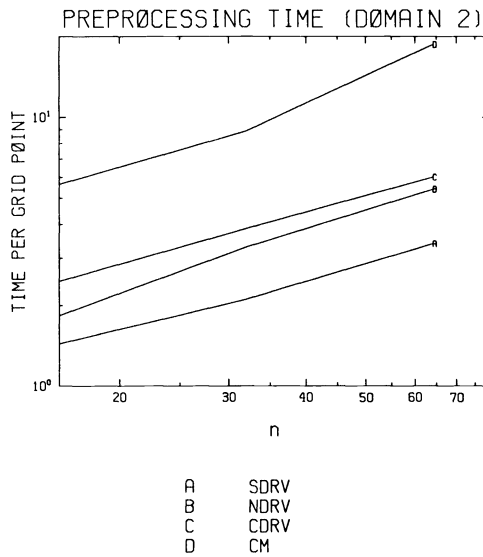


FIG. 8.7. Domain 2. Preprocessing time vs. n.

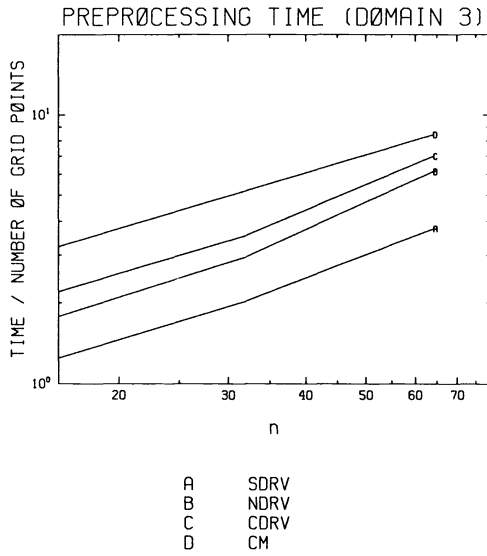


FIG. 8.8. Domain 3. Preprocessing time vs. n.

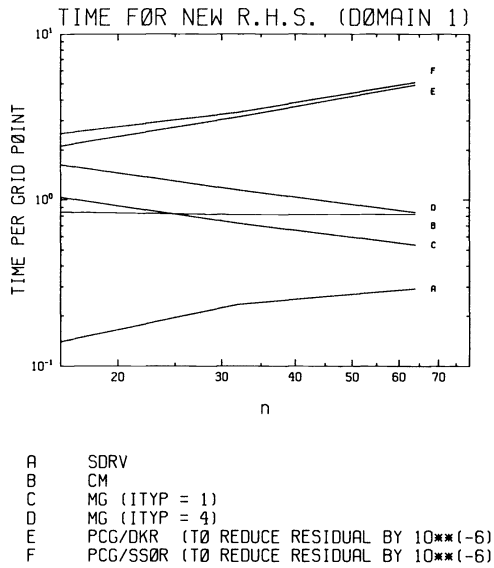


FIG. 8.9. Domain 1. Time for new RHS vs. n.

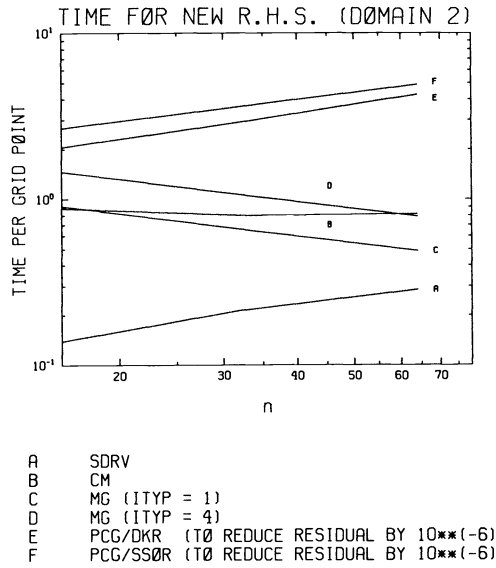


FIG. 8.10. Domain 2. Time for new RHS vs. n.

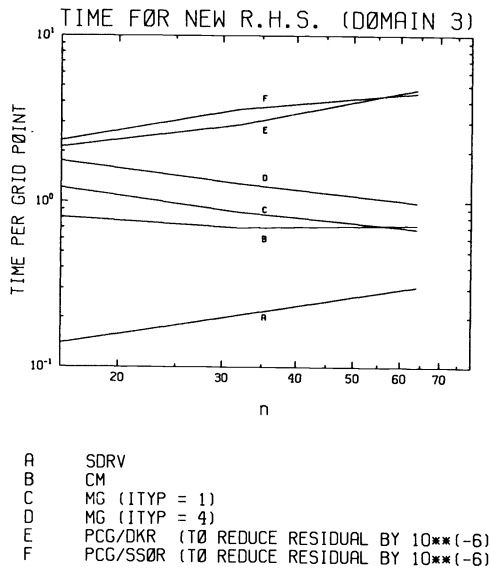


FIG. 8.11. Domain 3. Time for new RHS vs. n.

Figs. 8.2, 8.3 and 8.4 show the high storage needs of the sparse matrix solver. Even though the symmetric driver, SDRV, needed only half as much space as the nonsymmetric drivers, NDRV and CDRV, it still used more storage than any of the other methods. The MG code, on the other hand, required the least memory. PCG with the SSOR preconditioning needs only four vectors in addition to the matrix, the right hand side and the approximate solution. Approximately 11  $N$  words were needed to store the nonzeros of the matrix and the necessary pointers in the sparse storage format that was used. The DKR preconditioning took up almost twice as much memory as SSOR because the approximate factorization of the matrix required just as much



storage as  $A$  itself. We add that this higher storage requirement is attributable to the code used and that for two-cyclic matrices, efficient implementations with substantially less storage requirements are possible [10]. The storage for CM and PCG agrees with the  $O(n^2)$  behaviour as discussed in § 7. For MG01, the amount of storage required per grid point actually decreased as  $n$  was increased, but this is probably due to the relatively large overhead for smaller grid sizes. Fig. 8.5 shows how the size of the  $LU$  factors grows with the size of the problem for YSMP and Domain 2. The storage for the  $LU$  factors includes the integer storage used for storing the pointers to the  $LU$  factors and the permutation used to reorder the matrix. In our tests, an integer variable took up as much storage as a real variable. The total storage requirements of YSMP shown in the storage plots include the space needed to store the original matrix, the right hand side and the solution in addition to the space used for the  $LU$  factors. For  $n = 16, 32$  and  $64$ , we compute the following values for (size of  $LU$  factors)/( $N \log(N)$ ), for Domain 2: 2.86, 2.95 and 2.98 for the symmetric driver, SDRV, and 5.30, 6.43 and 7.25 for the nonsymmetric driver, NDRV. The  $LU$  factors appear to grow as  $N \log(N)$  for the symmetric case and somewhat faster for the nonsymmetric case. On computers where integers may take up less space than real numbers, it is possible to lower the storage requirements. For example, with the nonsymmetric driver, NDRV, roughly half the total space used is for integers. On a computer where a real number takes up twice as many bits as an integer, this represents a potential saving of 25% in terms of total storage requirements. The other drivers, SDRV and CDRV, use a compressed storage format for the pointers for the  $LU$  factors and would offer a smaller saving.

Figs. 8.6, 8.7 and 8.8 show that the CM method takes the most preprocessing time of all the methods. It is substantially more than that for the SM method, although the complexity for the two methods are of the same order. This is probably so because, in order to simulate a truly general region, we have not taken special advantage of the straight edges of the domains in our implementation, resulting in a larger value for  $p$  than necessary. However, even for the circular region in Domain 3, the preprocessing time for the CM method is more than twice that of the SM method. For the PCG methods, only the DKR preconditioning involves a preprocessing phase. The times can be found in Table 8.2 but they are negligible on the scale of the plots for the CM and SM methods. There is no preprocessing for the MG method.

Figs. 8.9, 8.10 and 8.11 show how the time needed per grid point to solve for a new right hand side varies with  $n$ . The times for NDRV and CDRV are indistinguishable from those for SDRV on this scale and hence were not plotted. The CM times exhibit  $O(n^2)$  behaviour in the plot while for MG the time per grid point is actually decreasing, probably due to the relatively large overhead for small problems. For PCG the times required for reducing the initial residual by a factor of  $10^{-6}$  are shown. These figures indicate that for larger grid sizes ( $n > 100$ ), MG would be faster than the solve phases of both YSMP and the CM method. We see that the SM method is the most efficient method for even reasonably large problems ( $n < 100$ ). On the other hand, the PCG methods are not competitive at this accuracy level.

In what follows, we shall give somewhat more details on the numerical results.

Table 8.2 gives in more details the times for the Preconditioned Conjugate Gradient method, with the DKR and the SSOR preconditionings, and also for the CG method (no preconditioning). The two preconditionings, DKR and SSOR, depend on scalar parameters,  $\alpha$  and  $\omega$  respectively. In our numerical tests, we used near optimal values of these parameters which were determined experimentally. For example, Figs. 8.12 and 8.13 give the dependence of  $N_1$  on  $\alpha$  and  $\omega$  for the DKR and the SSOR

TABLE 8.2  
Times for the preconditioned conjugate gradient method.

No pre-conditioning:

$n$		16	32	64	128
Domain 1:	$N_1$	36	74	240	
	$t_1$	15	56	231	
Domain 2:	$N_1$	29	69	240	
	$t_1$	16	58	249	
Domain 3:	$N_1$	18	45	204	
	$t_1$	14	54	222	

DKR pre-conditioning ( $\alpha = 0.0$ ):

Domain 1:	$N_1$	12	17	27	
	$t_p$	57	237	930	
	$t_1$	24	95	385	
Domain 2:	$N_1$	11	16	23	
	$t_p$	61	224	1039	
	$t_1$	27	99	415	
Domain 3:	$N_1$	11	16	25	
	$t_p$	58	257	887	
	$t_1$	25	92	391	

SSOR pre-conditioning ( $\omega$  given in brackets):

Domain 1:	$N_1$	11 (1.55)	15 (1.65)	22 (1.80)	31 (1.90)
	$t_1$	31	115	490	2078
Domain 2:	$N_1$	11 (1.55)	15 (1.65)	21 (1.80)	
	$t_1$	35	131	521	
Domain 3:	$N_1$	10 (1.55)	15 (1.65)	20 (1.80)	32 (1.90)
	$t_1$	30	122	465	1994

$N_1$  = Number of iterations required to reduce initial residual by  $10^{-6}$ .

$t_p$  = Time required for pre-processing (only for DKR).

$t_1$  = Time required for one iteration (without pre-processing).

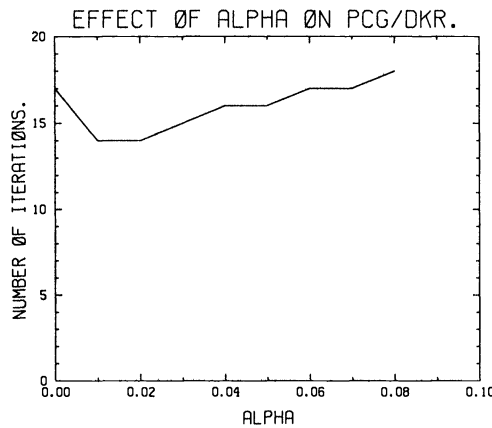


FIG. 8.12. PCG: Effect of parameter  $\alpha$  on DKR preconditioning.

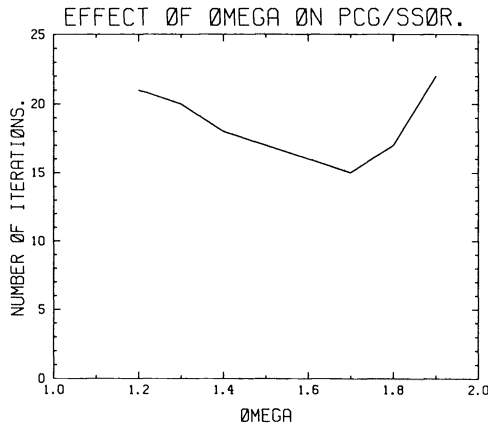


FIG. 8.13. PCG: Effect of parameter  $\omega$  on SSOR preconditioning.

preconditionings respectively, for Domain 1 with  $n = 32$ .  $N_1$  is the number of iterations required to reduce the initial residual by a factor of  $10^{-6}$ . The value  $\alpha = 0.0$  was used for all domains and grid sizes for the DKR preconditioning. For the SSOR preconditioning, the best value of  $\omega$  varied with grid size but not over the domains. The values of  $\omega$  used are given in Table 8.2.

The choice of  $10^{-6}$  for the error reduction factor for the PCG method was arbitrary but commensurate with the accuracy of the direct methods, starting with a zero initial guess. Thus, the number of iterations actually needed would be correspondingly lower if one had a better initial guess, and the initial residual did not have to be reduced by such a small factor. In order to compare the PCG method to the other competing methods, we give the number of PCG/DKR iterations (rounded up) that could be performed in the time needed by other methods (Table 8.3). Here the preprocessing times for the PCG method were included in computing the number of iterations performable in the Factor+Solve times of YSMP and the CM method. As can be seen, in the multiple right hand side case, the PCG method is competitive only if a few iterations are enough for controlling the accuracy and/or stability.

In order to compare the PCG method to the MG method in the case where truncation error accuracy is sufficient, we give the times for MG01, the relative residual achieved by it and the times and numbers of iterations needed by PCG/DKR and PCG/SSOR to achieve the same relative residual (see Table 8.4). The discrete 2-norm of the relative residuals should give an indication of the accuracy of the final solutions. We see that even at truncation error level, PCG is not competitive with MG. For all

TABLE 8.3  
PCG: Comparison with SM, CM and MG for domain 1,  $n = 64$ .

Method	Number of PCG/DKR iterations that can be performed in the same time (rounded up)	
YSMP (SDRV)	Factor+Solve:	18
YSMP (SDRV)	Solve:	2
CM	Factor+Solve:	92
CM	Solve:	5
MG	ITYP=1:	3
MG	ITYP=4:	4

TABLE 8.4  
PCG: Comparison with MG.

The following table gives the times and numbers of iterations required by PCG/DKR and PCG/SSOR to achieve the same relative residuals as MG01.

Domain 1	$n$	Rel residual (MG01)	MG01 time	PCG/DKR Time	( $N$ )	PCG/SSOR Time	( $N$ )
ITYP = 1	16	0.55 E-2	141	120	(5)	155	(5)
	32	0.242E-2	369	950	(10)	920	(8)
	64	0.698E-3	1131	6545	(17)	6370	(13)
ITYP = 4	16	0.293E-4	221	240	(10)	279	(9)
	32	0.267E-4	586	1330	(14)	1380	(12)
	64	0.440E-5	1775	9625	(25)	9800	(20)
Domain 2	$n$	Rel residual (MG01)	MG01 time	PCG/DKR Time	( $N$ )	PCG/SSOR Time	( $N$ )
ITYP = 1	16	0.133E-1	131	135	(5)	175	(5)
	32	0.582E-2	358	792	(8)	917	(7)
	64	0.159E-0	1096	2905	(7)	3126	(6)
ITYP = 4	16	0.611E-4	211	216	(8)	280	(8)
	32	0.759E-4	579	1188	(12)	1310	(10)
	64	0.508E-2	1770	4980	(12)	5210	(10)
Domain 3	$n$	Rel residual (MG01)	MG01 time	PCG/DKR Time	( $N$ )	PCG/SSOR Time	( $N$ )
ITYP = 1	16	0.711E-7	157	325	(13)	360	(12)
	32	0.270E-6	441	1564	(17)	1952	(16)
	64	0.130E-5	1416	9384	(24)	9300	(20)
ITYP = 4	16	0.365E-7	227	325	(13)	360	(12)
	32	0.226E-6	655	1564	(17)	1952	(16)
	64	0.112E-5	2035	9384	(24)	9300	(20)

values of  $n$ , the number of levels in the MG algorithm,  $m$ , was chosen so as to make the mesh-width of the coarsest grid,  $h_0 = \frac{1}{2}$ , i.e.  $m = \log_2(n)$ , since this was found to be faster than using a finer coarsest grid. Recall that MG01 uses Gauss-Seidel iterations to solve on the coarsest grid, rather than a direct solver and generates its own initial guess for the finest grid.

The solve phase for the CM method includes 2 calls to the Fast Helmholtz Solver (FHS) and one back-substitution of the capacitance matrix system. Asymptotically, the FHS is the higher order part of this phase and will dominate it. The observed percentage of the solve phase time that was spent in the FHS is given in Table 8.5. We see that about two thirds of the solve phase time is spent in the FHS. In other words, the time taken for one call of the FHS is about the same as one back-substitution of the capacitance matrix. These statistics may facilitate the usage of the results presented here when a different fast solver is used, e.g. when applying the CM method to more general elliptic problems.

TABLE 8.5  
CM: Percentage of time spent in the fast Helmholtz solver.

Domain	$n$		
	16	32	64
1	60.2	68.7	73.1
2	58.7	66.4	70.5
3	63.9	74.1	78.3

Finally, in Table 8.6 we compare the theoretical complexities of the different methods with the actual times that were obtained from our tests. The theoretical values for  $\alpha$  are given as well as the "empirical" values for each domain. For example, if the time for a method is expected to be  $Cn^\alpha$ , we determined the  $\alpha$  that gave the best linear fit to  $\ln T = \alpha \ln n + \ln C$  in the least squares sense. Because of the more complicated forms of the complexities for the CM method, we do not have sufficient data to a comparable fit. Hence the values for the CM method are not included. We see that the observed results agree quite well with the theoretical estimates.

**9. Concluding remarks.** Based on the results of the experiments presented in § 8, we can draw the following general conclusions:

1. If enough storage is available and many right-hand sides are to be processed, then the Sparse Matrix method seems to be the best choice even for relatively large values of  $n$  ( $\approx 100$ ).

2. The Preconditioned Conjugate Gradient method is not competitive if more than a few ( $\approx 5$ ) iterations are required to achieve the desired accuracy.

3. If truncation error level accuracy is adequate and a Multi-Grid solver is available for the particular elliptic operator and domain, then the MG method is very competitive in both work and storage, especially for larger problems ( $n > 64$ ).

4. The performance of the Capacitance Matrix method lies between those of the SM and MG methods. It requires less storage than the SM method, but quite a bit more than MG. For values of  $n$  up to 64, the work for a new right-hand side is slightly less than that of the SM method, but a factor of 2 or 3 more than that of the MG method. The preprocessing time is also substantially larger for a general domain. If a

TABLE 8.6  
Empirical time complexities.

The three empirical values are the least square fits of the timing figures to the theoretical estimates for each of the domains.

		Theoretical complexity	Theoretical $\alpha$	Empirical $\alpha$ (Domains 1, 2, 3)
SDRV	Factor	$n^\alpha$	3	2.82, 2.89, 3.05
	Solve	$n^\alpha \ln n$	2	2.22, 2.21, 2.20
NDRV	Factor	$n^\alpha$	3	3.16, 3.13, 3.27
	Solve	$n^\alpha \ln n$	2	2.21, 2.23, 2.22
MG		$n^\alpha$	2	1.69, 1.69, 1.74
PCG/DKR		$n^\alpha$	2.5	2.59, 2.50, 2.58
PCG/SSOR		$n^\alpha$	2.5	2.49, 2.41, 2.48

MG solver is not available and not enough storage is available for the SM method, or if the number of irregular points is small (e.g., almost rectangular domains), then the CM method should be considered.

While the experiments presented in this paper are necessarily limited in scope, we believe that this does not limit the usefulness of the results. We shall conclude by making a few remarks about using the results presented here to estimate relative performances of the methods in more general settings.

The first remark concerns the generality of the problems tested. While the Poisson problem is not the most general elliptic system, it is nonetheless heavily used in practice, very often as a preconditioner for more general problems. Moreover, the results reported in the paper do not really take advantage of any special features of the Poisson equation. The relative efficiency of these methods, in terms of speed and storage, should be about the same for more general problems. This is certainly true of the sparse matrix methods because the work there only depends on the sparsity pattern and not on the particular values of the elements of the matrix. The convergence rate of multi-grid methods have also been shown to be not very sensitive to variations in the coefficients of the elliptic system, as long as they are reasonably smooth. Similarly, the convergence rate of conjugate gradient methods depends more on the distribution of eigenvalues rather than on the specific matrix involved and for that the Poisson equation is representative. The only possible exception is the capacitance matrix code that we used, which does only work for the Helmholtz equation. However, even here the results can still be useful, because the only place where this matter is in the fast solver on a rectangle and fast solvers do exist for more general elliptic operators. In any case, the timings for this part of the calculation are tabulated separately and thus the results can still be used to extrapolate to more general problems when the timings for the appropriate fast solver are substituted.

The second remark concerns the choice of initial guesses for the CG and MG methods as the actual amount of computation taken by the methods obviously depend on them. In a time dependent problem, initial guesses would be available from extrapolations from previous time steps. However, comparing these two methods in a true time dependent setting will actually make the results less general because the dependence on the particular problem, e.g. its transient features, the predictor used etc., would make it more difficult to extrapolate the results to another problem. We record the work required for reducing the initial error by a certain factor which should be more problem independent. In fact, point number 2 above is stated in this fashion.

The above considerations also apply to the factorization time for the sparse matrix methods for time dependent problems. The frequency with which the refactorization has to be done is necessarily problem dependent. Again, we present the time for the factorization phase separately so that for a specific problem they can be used to estimate the actual time. In any case, the conclusions that we made are valid for cases where the refactorization cost is relatively small.

**Acknowledgments.** The authors would like to thank Professor Martin H. Schultz and Stanley C. Eisenstat, and Dr. Howard C. Elman for their many helpful suggestions throughout this project.

#### REFERENCES

- [1] O. AXELSSON, *A generalized SSOR method*, BIT, 13 (1972), pp. 443-467.
- [2] R. E. BANK AND T. DUPONT, *An optimal order process for solving elliptic finite element equations*, Math. Comp., 36 (1981), pp. 35-51.

- [3] A. BRANDT, *Multi-level adaptive solutions to boundary-value problems*, Math. Comp., 31 (1977), pp. 333-390.
- [4] RATI CHANDRA, *Conjugate gradient methods for partial differential equations*, Ph.D. thesis, Dept. Computer Science, Yale Univ., New Haven, CT, January, 1982, also available as Technical Report # 129.
- [5] 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 Proc. Symposium on Sparse Matrix Computations, J. R. Bunch and D. J. Rose eds., Academic Press, New York, 1975, pp. 309-332.
- [6] J. E. DENDY, JR. AND J. M. HYMAN, *Multi-grid and ICCG for problems with interfaces*, in Elliptic Problem Solvers, M. H. Schultz ed., Academic Press, New York, 1981, pp. 247-253.
- [7] C. C. DOUGLAS, *Multi-grid problems for elliptic boundary-value problems*, Ph.D. thesis Dept. Computer Science, Yale Univ., New Haven, CT, May, 1982, also available as Technical Report 223.
- [8] I. S. DUFF, *Sparse matrix software for elliptic PDE's*, in Multigrid Methods, Proceedings, Köln-Porz, 1981, W. Hackbusch and U. Trottenberg eds., Springer-Verlag, Berlin, 1982.
- [9] 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.
- [10] S. C. EISENSTAT, *Efficient implementation of a class of preconditioned conjugate gradient methods*, this Journal, 2 (1981), pp. 1-4.
- [11] S. C. EISENSTAT, M. C. GURSKY, M. H. SCHULTZ AND A. H. SHERMAN, *The Yale sparse matrix package I: The symmetric codes*, Int. J. Numer. Methods Eng., 18 (1982); *The Yale sparse matrix package II: The unsymmetric codes*, to appear.
- [12] H. C. ELMAN, *Iterative methods for large, sparse, nonsymmetric systems of linear equations*, Ph.D. thesis, Dept. Computer Science, Yale Univ., New Haven, CT, April, 1982, also available as Technical Report #229.
- [13] H. FOERSTER, K. STÜBEN AND U. TROTTEBERG, *Nonstandard multi-grid techniques using checkered relaxation*, in Elliptic Problem Solvers, M. H. Schultz ed., Academic Press, 1981, pp. 285-300.
- [14] J. A. GEORGE AND J. W.-H. LIU, *Computer Solution of Large Sparse Positive Definite Systems*, Prentice-Hall, Englewood Cliffs, NJ, 1981.
- [15] I. GUSTAFSSON, *A class of first order factorizations*, BIT, 18(1978), pp. 142-156.
- [16] W. HACKBUSCH, *Convergence of multi-grid iterations applied to difference equations*, Math. Comp., 34 (1980), pp. 425-440.
- [17] M. R. HESTENES AND E. STIEFEL, *Methods for conjugate gradient for solving linear systems*, J. Res. National Bureau of Standards, 49 (1952), pp. 409-436.
- [18] K. C. JEA, *Generalized conjugate gradient acceleration of iterative methods*, Ph.D. thesis, Center of Numerical Analysis, Univ. Texas at Austin, 1982, also available as CNA Research Report #CNA-176.
- [19] 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.
- [20] B. N. PARLETT, *A new look at the Lanczos algorithm for solving symmetric systems of linear systems*, L.A.A., 29 (1980), pp. 323-346.
- [21] 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.
- [22] W. PROSKUROWSKI, *Four Fortran programs for numerically solving Helmholtz's equation in an arbitrary bounded planar region*, Report LBL-7516, Lawrence Berkeley Laboratory, Univ. California, Berkeley, 1978.
- [23] J. R. RICE, *Performance analysis of 13 methods to solve the Galerkin method equations*, L.A.A., 52/53 (1983), pp. 533-546.
- [24] M. H. SCHULTZ, ed., *Elliptic Problem Solvers*, Academic Press, New York, 1981.
- [25] U. SCHUMANN, ed., *Fast Elliptic Solvers*, Advance Publications, United Kingdom, 1977.
- [26] K. STÜBEN, *A multi-grid program to solve  $\Delta U - c(x, y)U = f(x, y)$  (on  $\Omega$ ),  $U = g(x, y)$  (on  $\Omega$ ) on nonrectangular bounded domains  $\Omega$* , IMA-Report 82.02.02, GMD, Bonn, 1982.
- [27] A. J. WALLCRAFT, *The preconditioned capacity matrix technique*, in Advances in Computer Methods for Partial Differential Equations-IV, R. Vichnevetsky and R. S. Stepleman, eds., IMACS, Rutgers Univ., New Brunswick, NJ, 1981.
- [28] O. WIDLUND, *A Lanczos method for a class of nonsymmetric systems of linear equations*, SIAM J. Numer. Anal., 15 (1978), pp. 801-812.

## ELEMENT PRECONDITIONING USING SPLITTING TECHNIQUES\*

B. NOUR-OMID† AND B. N. PARLETT‡

**Abstract.** The finite element method is a good way to turn elliptic boundary value problems into large symmetric systems of equations. These large matrices,  $\mathbf{A}$ , are usually assembled from small ones. It is simple to omit the assembly process and use the code to accumulate the product  $\mathbf{A}v$  for any  $v$ . Consequently the conjugate gradient algorithm (CG) can be used to solve  $\mathbf{A}\mathbf{x} = \mathbf{b}$  without ever forming  $\mathbf{A}$ .

It is well known, however, that CG performs best when applied to preconditioned systems. In this paper we show one way to precondition without forming any large matrices.

The trade-off between time and storage is examined for the 1-D model problem and for the analysis of several realistic structures.

**Key words.** finite element, conjugate gradient, preconditioning

**1. Introduction.** Much effort has been expended on exploiting the sparsity structure of large symmetric matrices  $\mathbf{A}$  when solving linear systems

$$(1) \quad \mathbf{A}\mathbf{x} = \mathbf{b}$$

(see [1] and [2]). Our interest is in the systems which arise from the application of the finite element method to engineering structures.

The field, or speciality, called sparse matrix technology has been concerned to a large extent with minimizing, or at least reducing, the storage requirements for solving  $\mathbf{A}\mathbf{x} = \mathbf{b}$  by direct methods. The experts in the subject relentlessly hunt down unfortunate matrix elements which get "filled-in" during the process of Choleski factorization. By various clever techniques the menacing fill-in is often kept down to the order of the original number of nonzero elements in the matrix. Nevertheless big problems make big demands on storage.

The outsider to this field of sparse matrix technology can be forgiven for wondering why the original matrix components are treated with such respect, while the fill-ins are despised and only with the greatest reluctance given a place to stay. After all, each component of a matrix makes the same storage penalty.

If storage really is in short supply, then *every* aspect of the solution procedure should be examined for possible economies. Why discriminate solely against fill-in?

It has been pointed out in [3] and [4] that it is not essential to assemble the stiffness matrix  $\mathbf{A}$  from the sum of many simple matrices. The code which produces  $\mathbf{A}$  from its usual definition

$$(2) \quad \mathbf{A} = \sum_e \mathbf{N}_e \mathbf{a}_e \mathbf{N}_e^t$$

can be modified to produce, for any given column vector  $\mathbf{u}$ ,

$$(3) \quad \mathbf{A}\mathbf{u} = \sum_e (\mathbf{N}_e \mathbf{a}_e \mathbf{N}_e^t \mathbf{u})$$

---

\* Received by the editors December 13, 1982, and in final revised form October 15, 1984. This paper was presented at the Sparse Matrix Symposium held at Fairfield Glade, Tennessee, October 25-27, 1982. This work was supported in part by the Office of Naval Research under contract N00014-76-C-0013.

† Center for Pure and Applied Mathematics, University of California, Berkeley, California 94720.

‡ Department of Mathematics, and the Computer Science Division of the Department of Electrical Engineering and Computer Science, University of California, Berkeley, California 94720.



without ever forming the elements of  $\mathbf{A}$ . Here the  $\mathbf{N}_e$  are long, thin Boolean connectivity matrices and  $\mathbf{a}_e$  denotes the small stiffness matrix for element  $e$ . Moreover,  $\mathbf{a}_e$  itself may be a product, say  $\mathbf{r}_e' \mathbf{k}_e \mathbf{r}_e$ , such that  $\mathbf{k}_e$  and  $\mathbf{r}_e$  require less space than  $\mathbf{a}_e$ . It is our belief that B. Irons was the first to make use of this observation. Of course, it requires more arithmetic operations to use (3) than to use  $\mathbf{A}$  explicitly in some compact form. The extra arithmetic might double the cost of a step, but with clever coding on the right type of computer it might cause very little increase in elapsed time. The motivation in [3] and [4] was to avoid cancellation of digits during the assembling of  $\mathbf{A}$ . Ours is to reduce storage demands at the cost of increasing the arithmetic effort.

In the absence of  $\mathbf{A}$ 's elements direct methods are out but it is easy to use the conjugate gradient algorithm (CG hereafter) to solve  $\mathbf{Ax} = \mathbf{b}$ , since  $\mathbf{A}$  is symmetric, positive definite. The CG method does not modify  $\mathbf{A}$  and needs only 3  $n$ -vectors in the fast store. In fact  $\mathbf{A}$  is needed only to multiply a vector at each step of the process.

Frontal methods [5] successfully keep down the fast storage needs but, in the end, all the nonzeros of the Choleski factor of  $\mathbf{A}$  have to be kept somewhere, presumably in secondary store and accessed as needed. So total storage and access do exceed those of the element approach.

Unfortunately CG can take many steps to reduce the residual norm to an acceptable level. One lesson at least has been learnt about CG during the last 10 years.

*CG should be used with preconditioning.*

The problem which remains is

*How to precondition an unformed matrix?*

There are a number of different ways to produce implicit preconditioners that use the same structure as in (3). This communication describes a few approaches based on the popular splitting technique for solving differential equations. Our goal is to demonstrate how preconditioners can be constructed and to exhibit their effect on the convergence of the CG algorithm. We make no claim that these techniques are among the best to be used. Two of our numerical tests are taken from realistic problems but the size of the matrices which we can handle on our system (time-shared VAX) is too small to model serious applications.

We end this introduction by pointing out that our interest in element preconditioning is in keeping down storage requirements in the analysis of regular structures but the advent of parallel computing may make this approach a fast one as well.

**2. Preconditioned conjugate gradient.** Let  $\kappa = \kappa(\mathbf{A})$  be  $\mathbf{A}$ 's condition number;  $\kappa(\mathbf{A}) = \|\mathbf{A}\| \|\mathbf{A}^{-1}\|$ . The standard worst case theory of CG (see [6]) says that the residual norm is reduced by at worst a factor of  $(\sqrt{\kappa} - 1)/(\sqrt{\kappa} + 1)$  at each step on the average. However, in practice convergence is observed to be superlinear near the end of the iteration. The whole eigenvalue distribution is more important than the ratio of the largest to smallest.

Given an initial guess  $\mathbf{x}_0$  and a positive definite matrix  $\mathbf{M}$  close to  $\mathbf{A}$ , the preconditioned CG algorithm generates a sequence  $\mathbf{x}_k$  of approximations to the solution  $\mathbf{x}$  as follows:

- (1) Set  $\mathbf{p}_0 = \mathbf{r}_0 = \mathbf{b} - \mathbf{Ax}_0$ .
- (2) Solve  $\mathbf{Md}_0 = \mathbf{r}_0$ .
- (3) For  $k = 0$  step 1 until convergence do
  - (a)  $\alpha_k = (\mathbf{r}_k, \mathbf{d}_k) / (\mathbf{p}_k, \mathbf{Ap}_k)$ ,
  - (b)  $\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{p}_k$ ,

- (c)  $\mathbf{r}_{k+1} = \mathbf{r}_k - \alpha_k \mathbf{A} \mathbf{p}_k$ ,
- (d) Solve  $\mathbf{M} \mathbf{d}_{k+1} = \mathbf{r}_{k+1}$ .
- (e)  $\beta_k = (\mathbf{r}_{k+1}, \mathbf{d}_{k+1}) / (\mathbf{r}_k, \mathbf{d}_k)$ ,
- (f)  $\mathbf{p}_{k+1} = \mathbf{d}_{k+1} + \beta_k \mathbf{p}_k$ .

**Polynomial preconditioning.** A natural choice for  $\mathbf{M}^{-1}$  is the polynomial preconditioning (PP here after) proposed by Johnson et al. [7]. This employs a low degree polynomial in  $\mathbf{A}$ ,  $P_m(\mathbf{A})$ , as the inverse of the preconditioning matrix. The matrix-vector product,  $P_m(\mathbf{A})\mathbf{v}$ , for a given vector  $\mathbf{v}$  can be computed via  $P_m(\sum_e \mathbf{N}_e \mathbf{a}_e \mathbf{N}_e') \mathbf{v}$  in much the same way as  $\mathbf{A}\mathbf{v}$ . This method of preconditioning increases the cost of one CG iteration by  $m$  matrix vector multiplications and  $m$  vector additions. However, on vector processors this extra work takes little time.

The theoretical weakness of PP is as follows. Let  $k^j[\mathbf{f}; \mathbf{A}]$  denote the Krylov subspace spanned by the basis vectors  $[\mathbf{f}, \mathbf{A}\mathbf{f}, \mathbf{A}^2\mathbf{f}, \dots, \mathbf{A}^{j-1}\mathbf{f}]$ . After  $j-1$  steps it produces the best approximation to  $\mathbf{A}^{-1}\mathbf{b}$  from a special Krylov subspace of dimension  $j$ , namely

$$k_1 = k^j[P_m(\mathbf{A})\mathbf{b}; P_m(\mathbf{A})\mathbf{A}].$$

For the same number of matrix-vector products CG would produce the best approximation from

$$k_2 = k^{j(m+1)}[\mathbf{b}; \mathbf{A}].$$

But  $k_1$  is a subspace of  $k_2$ , and so there are better approximations to the solution in  $k_2$  than in  $k_1$ . Consequently standard CG will produce a better approximation than PPCG for the same number of matrix-vector products. As indicated above machines like the CRAY and the CYBER 205 make the assessment of efficiency more complicated. It depends on the extent to which the computation of  $\mathbf{A}\mathbf{v}$  dominates a step of the iteration. We must also emphasize that the weakness is a theoretical one, based on exact arithmetic. PPCG is useful.

**3. Element preconditionings.** The code which forms  $\mathbf{v} = \sum_e \mathbf{N}_e \mathbf{a}_e \mathbf{N}_e' \mathbf{u}$  can also produce  $\mathbf{w} = \sum_e (\mathbf{N}_e \mathbf{b}_e \mathbf{N}_e' \mathbf{v})$  for any choice of the element conditioning matrices  $\mathbf{b}_e$  which have the same dimensions as the corresponding  $\mathbf{a}_e$ . Consequently it is feasible to use as preconditioner any matrix of the form

$$(4) \quad \mathbf{B} = \sum_e (\mathbf{N}_e \mathbf{b}_e \mathbf{N}_e').$$

This raises three technical questions.

- (1) For a given collection  $\{\mathbf{a}_e\}$ , what is the best choice of  $\{\mathbf{b}_e\}$  such that  $\kappa(\mathbf{B}\mathbf{A})$  is minimized?
- (2) How significant a reduction in the condition number does this optimal  $\mathbf{B}$  produce?
- (3) Can one find a good, if not optimal,  $\mathbf{B}$  in practice?

In contrast to explicit conditioners (which approximate  $\mathbf{A}$ ) our method often approximates  $\mathbf{A}^{-1}$  (step 3(d) in the algorithm becomes  $\mathbf{d}_{k+1} = \mathbf{B}\mathbf{r}_{k+1}$ ). That is, standard preconditioners actually solve the system  $\mathbf{M}\mathbf{d}_{k+1} = \mathbf{r}_{k+1}$  in step 3(d) of the algorithm. Our preconditioners usually operate directly on  $\mathbf{r}_{k+1}$  to find  $\mathbf{d}_{k+1}$ . Consequently,  $\mathbf{B}$  defines  $\mathbf{M}^{-1}$ . The matrix  $\mathbf{B}$  is restricted to have the same sparsity structure as  $\mathbf{A}$  and consequently yields a weak preconditioner. For example, no tridiagonal matrix approximates well the inverse of the modal tridiagonal matrix (diagonal elements 2, and off diagonal elements  $-1$ ). However our goal is to keep down storage costs.

These questions, though interesting, will not be pursued here because we have no theoretical results as yet. Instead we present a special class of conditioners. The code which forms  $\mathbf{v} = \sum_e (\mathbf{N}_e \mathbf{a}_e \mathbf{N}_e^t \mathbf{u})$  can, with trivial modifications, also form either  $\mathbf{v} = \prod_e (\mathbf{I} + \mathbf{N}_e \mathbf{g}_e \mathbf{N}_e^t) \mathbf{u}$  or  $\mathbf{v} = \prod_e (\mathbf{I} + \mathbf{N}_e \mathbf{g}_e \mathbf{N}_e^t)^{-1} \mathbf{u}$  for any matrix  $\mathbf{g}_e$  with same dimensions as  $\mathbf{a}_e$ . The three questions raised above can now be posed again.

In the sequel we construct various forms for  $\mathbf{g}_e$  by considering "splitting methods" for the solution of linear systems as the steady state of diffusion equations.

**Preconditioning based on splitting technique.** Let  $\mathbf{A} = \sum_e \mathbf{A}_e$ , where  $\mathbf{A}_e = \mathbf{N}_e \mathbf{a}_e \mathbf{N}_e^t$ . Then, based on the splitting technique (see [8], [9], and [10]) for solving  $\dot{\mathbf{x}} + \mathbf{A}\mathbf{x} = \mathbf{b}$ , we approximate  $\mathbf{A}^{-1}$  by

$$(5) \quad \mathbf{P} = \prod_{e=1}^{n_e} (\mathbf{I} + \sigma \mathbf{A}_e)^{-1}$$

where  $n_e$  is the total number of finite elements, and  $\sigma$  is a free parameter.

The matrix  $\mathbf{P}$  is not symmetric and so we do not use it directly as  $\mathbf{M}^{-1}$  in the PCG algorithm described in § 2. Here we propose three natural ways to create symmetric positive definite preconditioners.

(I) *Element Choleski factor (ECF)*. Unless  $\sigma$  is too large, the matrix  $(\mathbf{I} + \sigma \mathbf{A}_e)$  has a Choleski factorization which can be written as

$$(6) \quad \mathbf{I} + \sigma \mathbf{A}_e = \mathbf{C}_e \mathbf{C}_e^t$$

where  $\mathbf{C}_e$  is a function of  $\sigma$ . Define

$$(7) \quad \mathbf{C} = \prod_{e=1}^{n_e} \mathbf{C}_e$$

and use  $\mathbf{M} = \mathbf{C}\mathbf{C}^t$  to precondition  $\mathbf{A}$ . Here,  $\mathbf{C}$  approximates the Choleski factor of  $\mathbf{A}$  by the product of  $n_e$  lower triangular matrices. Then,  $\mathbf{M}^{-1}$  is  $\prod_{e=1}^{n_e} \mathbf{C}_e^{-T} \prod_{e=n_e}^1 \mathbf{C}_e^{-1}$ .

(II) *Symmetric element product (SEP)*. As in the SSOR method it is possible to use  $\mathbf{M}^{-1} = \mathbf{P}\mathbf{P}^t$  as a preconditioner, with  $\mathbf{P}$  given by (5).

(III) *Element splitting (ES)*. Let  $\mathbf{u}_e + \mathbf{u}_e^t = \mathbf{a}_e$ . Define

$$(8) \quad \mathbf{U} = \prod_{e=1}^{n_e} (\mathbf{I} + \sigma \mathbf{U}_e)$$

where  $\mathbf{U}_e = \mathbf{N}_e \mathbf{u}_e \mathbf{N}_e^t$ . Then use  $\mathbf{M} = \mathbf{U}^t \mathbf{U}$  as a preconditioner.  $\mathbf{M}^{-1}$  is obtained the same way as in ECF method.

The first technique (ECF), though not consistent with known splitting methods is first order accurate when applied directly to systems of first order ordinary differential equations. The second method (SEP) can be identified with the alternating direction method and is second order accurate. This method has been used to solve certain differential equations with some degree of success [8], [9]. The third preconditioner (ES) is best viewed as applying the splitting method directly to  $\mathbf{A} = \sum_e \mathbf{N}_e \mathbf{u}_e \mathbf{N}_e^t + \mathbf{N}_e \mathbf{u}_e^t \mathbf{N}_e^t$  and therefore is first order accurate.

In summary, our idea is to take the methods used in [8] and [9] as system solvers but to treat them simply as preconditioners.

**Model problem.** In this section we try to assess the effectiveness of the different preconditioners described in the previous section. As an example we apply one of the above methods, namely ECF, to our model problem. Consider the one-dimensional problem with  $-y'' = f$  in the interval  $[0, n + 1]$  with Dirichlet boundary conditions. The domain is discretized using  $n + 1$  finite elements. This results in a common element

matrix

$$(9) \quad \mathbf{a}_e = \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix}.$$

The resulting global matrix  $\mathbf{A}$  is tridiagonal, with diagonal elements 2, and off-diagonal elements  $-1$ . The matrices involved in (5) are of the form

$$(10) \quad (\mathbf{I} + \sigma \mathbf{A}_e) = \begin{bmatrix} I & & & \\ & 1 + \sigma & -\sigma & \\ & -\sigma & 1 + \sigma & \\ & & & I \end{bmatrix}.$$

The Choleski factor of this matrix is

$$(11) \quad \mathbf{C}_e = \begin{bmatrix} I & & & \\ & \sqrt{1 + \sigma} & & \\ & \frac{-\sigma}{\sqrt{1 + \sigma}} & \left(\frac{1 + 2\sigma}{1 + \sigma}\right)^{1/2} & \\ & & & I \end{bmatrix}.$$

Note that in forming  $\mathbf{C}$  the boundary conditions are applied to  $\mathbf{C}_e$  rather than  $\mathbf{A}_e$ . The preconditioning matrix, say  $\mathbf{M}$ , resulting from ECF method can be written as

$$(12) \quad \mathbf{M} = \mathbf{C}\mathbf{C}^T = \begin{bmatrix} a & & & \\ b & a & & \\ & \cdot & \cdot & \\ & & \cdot & a \\ & & & b & a \end{bmatrix} \begin{bmatrix} a & b & & \\ & a & \cdot & \\ & & \cdot & \\ & & & a & b \\ & & & & a \end{bmatrix}$$

where  $a = \sqrt{1 + 2\sigma}$  and  $b = -\sigma/\sqrt{1 + \sigma}$ . The explicit form of  $\mathbf{M}$  can now be written as

$$(13) \quad \mathbf{M} = \begin{bmatrix} a^2 & ab & & & \\ ab & a^2 + b^2 & ab & & \\ & ab & a^2 + b^2 & \dots & \\ & & \cdot & \cdot & ab \\ & & & ab & a^2 + b^2 \end{bmatrix}.$$

Scaling this matrix by a factor  $2/(a^2 + b^2)$ , we get

$$(14) \quad \tilde{\mathbf{M}} = \begin{bmatrix} 2\alpha & -\beta & & & \\ -\beta & 2 & & & \\ & & \ddots & & \\ & & & 2 & -\beta \\ & & & -\beta & 2 \end{bmatrix}$$

where

$$\alpha = \frac{2\sigma^2 + 3\sigma + 1}{3\sigma^2 + 3\sigma + 1}$$

and

$$\beta = \frac{2\sigma\sqrt{(1 + 2\sigma)(1 + \sigma)}}{3\sigma^2 + 3\sigma + 1}.$$

Defining the error matrix  $E = A - \tilde{M}$ , we get

$$(15) \quad E = \begin{bmatrix} 0 & \beta - 1 & & & \\ \beta - 1 & 0 & & & \\ & & \dots & & \\ & & & 0 & \beta - 1 \\ & & & \beta - 1 & 0 \end{bmatrix} + 2(1 - \alpha)e_1e_1^T$$

where  $e_1$  is the first column of the identity. The matrix  $e_1e_1^T$  is only of rank one and may increase the total number of iterations in a CG run by at most one and therefore may be ignored in this analysis. The norm of the remainder depends on  $|\beta - 1|$  which gets small for large values of  $\sigma$ . In fact

$$(16) \quad \lim_{\sigma \rightarrow \infty} |\beta - 1| = 1 - \frac{2\sqrt{2}}{3} \approx 0.058$$

which suggest that  $M$  may be a good preconditioning matrix for  $A$ .

We ran tests for various values of  $\sigma$ . The condition number of  $M^{-1}A$  dropped from 50 for  $\sigma = 0$  to below 10 for  $\sigma \geq 2$ . See [11] for more details.

The CG method was used to obtain the solution to the discretized model problem for a given right-hand side. The actual number of iterations required to converge is plotted against  $\sigma$  (Fig. 1). The tolerance for convergence was set at  $10^{-8}$ . The value of  $\sigma = 0$  corresponds to CG with no preconditioning. It can be seen that for this example the correct choice for  $\sigma$  results in reduction in the number of iterations by a factor of nearly 4. This analysis does reveal the limitations of this form of preconditioning. Note that our tolerance is quite severe.

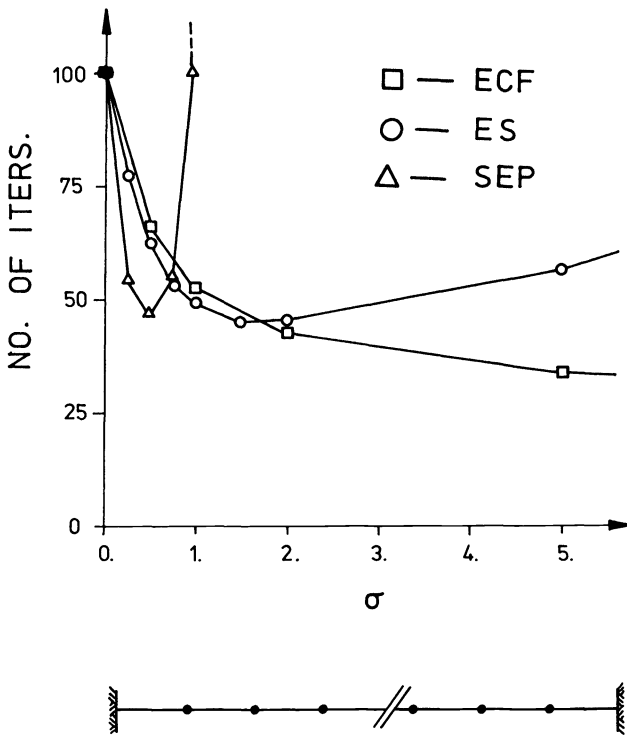


FIG. 1. 100 × 100 1-D model problem.

**4. Implementation.** We view the formation of  $\mathbf{Au}$  via (3) as a mechanism for saving storage in return for extra arithmetic work. The reduction in storage demand is due to the following observations:

1. In most practical finite element (FE hereafter) problems there is a considerable amount of repetition of a given element in the mesh structure.
2. The element matrices of a number of element types, such as beams, trusses, etc., are known explicitly and depend on only a few fundamental parameters.









The first observation allows us to create a data structure which keeps the element matrix of one element to represent a whole group of elements. The second observation results in a canonical form for each element type, and therefore only a few parameters need be stored to define each element matrix. Hence the storage requirements for all the distinct  $\mathbf{a}_e$  is often significantly less than the number of words required to hold  $\mathbf{A}$ , even when a sophisticated sparse storage scheme is used (see [1], [2]). Furthermore, one can always recompute the element matrices  $\mathbf{a}_e$  each time the product  $\mathbf{Av}$  is required.

**Cost overhead.** The overhead for the reduction in storage is the increased number of operations. However, two comments are in order.

1. The cost of a multiply no longer dominates arithmetic evaluations.
2. Vector machines or other special purpose devices can execute  $\sum_e (\mathbf{N}_e \mathbf{a}_e \mathbf{N}_e' \mathbf{u})$  very efficiently.

TABLE 1

*The cost factor  $\gamma$  and reduction in storage for different elements on regular mesh with 2 unknowns per node.  $r$  is the average number of elements connected to each node and  $m$  is the number of nodes per element.*

Dimension of space	element type	$r$	$m$	$\gamma$	% savings in storage
1-D		2	2	1.33	85
		$\frac{3}{2}$	3	1.06	89
2-D		4	4	1.77	88
		$\frac{25}{9}$	9	1.27	94
		6	3	2.57	88
		4	6	1.57	94
3-D		8	8	2.37	90
		$\frac{125}{27}$	27	1.40	98

When using the implicit form of  $Av$  it can be seen that different elements operate on different parts of the vector  $v$ . One can take advantage of this fact by performing some of the element matrix operations in parallel.

The implicit product increases the cost of matrix operations by a factor of, say  $\gamma$ , where  $\gamma$  depends on  $r$ , the average number of elements connected to a node.  $\gamma$  is computed for a few different elements on regular meshes. It can be seen from Table 1 that the magnitude of  $\gamma$  is not very large, although one could design examples that result in large  $\gamma$ . One such example is the star graph with all the nodes constrained except the center node. For this example  $\gamma = n_e$ .

**5. Numerical examples.** We performed a number of tests designed to illustrate the effectiveness and limitations of the three preconditioners developed here. Each method was used to solve the problems for a range of  $\sigma$ . The standard CG method corresponds to  $\sigma = 0.0$ . We set the factor for the reduction in residual norm at  $10^{-8}$ .

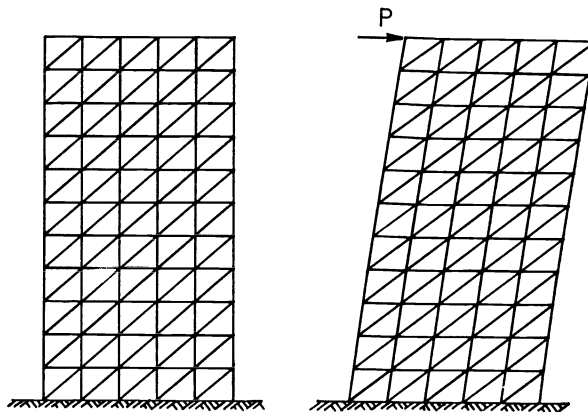
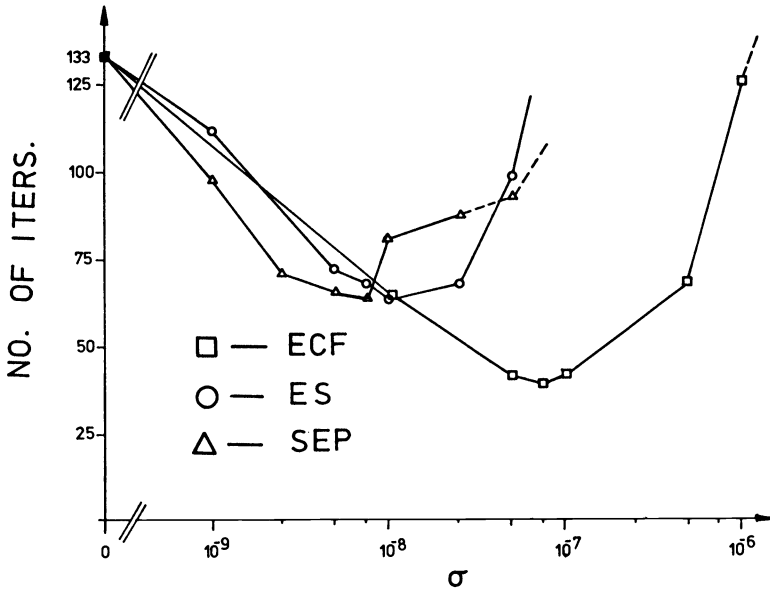


FIG. 2. Truss building example.

The first set of tests was performed on the  $100 \times 100$  model problem (see Fig. 1). The numerical results demonstrate the reduction in the number of iterations for the ECF method for large values of  $\sigma$ . The smallest number of iterations achieved with the ECF method was 26 when  $\sigma = 10^3$ . The minimum number of iterations for the ES method was 45 with any  $\sigma$  in the interval  $[1.5, 2.0]$ . The curve for the SEP method has a very sharp minimum, and therefore small changes in  $\sigma$  will result in large changes in the number of iterations and therefore the optimum  $\sigma$  (0.4 in this case) is difficult to estimate.

We performed similar tests on two different, but realistic problems occurring in FE applications:

(1) a building example consisting of truss elements with a total of 132 unknowns (Fig. 2),

(2) a plane stress problem (Laplacian on rectangular domain) corresponding to a beam with a total of 160 unknowns (Fig. 3).

In all the test performed a ECF preconditioner was found to be the most effective. A reduction by a factor of four in the number of iterations was observed with the model problem and the building example. However the ES method required less

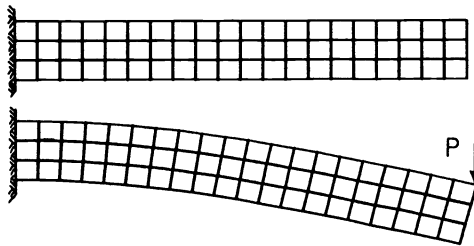
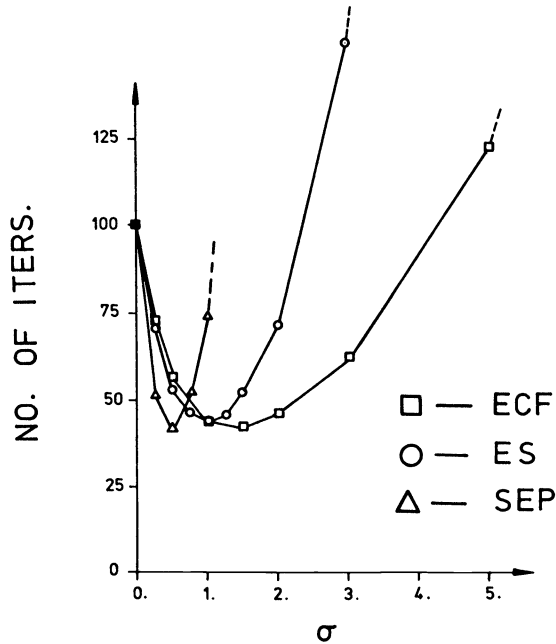


FIG. 3. Plane stress problem.



computer time. This is because there is no factorization of element matrices in the ES method.

The result of our tests demonstrate that the optimum value of  $\sigma$  can vary a lot. This is due to the fact that  $\sigma_{opt}$  is not invariant under scaling of  $\mathbf{A}$  and depends on the range of the terms in  $\mathbf{A}$ .

We should point out that in all the tests that were carried out the problems were too small and too well conditioned to display the advantages of preconditioning.

There is much more work to be done. Our goal was to introduce a new approach to the analysis of huge, regular structures.

**Acknowledgment.** The authors wish to thank Carlos Rodriguez for performing the numerical tests reported here.

#### REFERENCES

- [1] A. GEORGE AND J. W. LIU, *Computer Solution of Large Sparse Positive Definite Systems*, Prentice-Hall, Englewood Cliffs, NJ, 1981.
- [2] I. S. DUFF, *A survey of sparse matrix research*, Proc. IEEE, 65 (1977), pp. 500-535.
- [3] R. L. FOX AND E. L. STANTON, *Developments in structured analysis by direct energy minimization*, AIAA J., 6 (1968), pp. 1036-1042.
- [4] I. FRIED, *More on gradient iterative methods in finite-element analysis*, AIAA J., 7 (1969), pp. 565-567.
- [5] B. M. IRONS, *A frontal solution program for finite element analysis*, Int. J. Numer. Meth. Engrg., 2 (1970), pp. 5-32.
- [6] 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.
- [7] O. G. JOHNSON, C. A. MICCHELLI AND G. PAUL, *Polynomial preconditioners for conjugate gradient calculations*, SIAM J. Numer. Anal., 20 (1983), pp. 362-376.
- [8] T. J. R. HUGHES, I. LEVIT AND J. WINGET, *Element-by element implicit algorithms for heat conduction*, J. Engng. Mech., 109 (1983).
- [9] M. ORTIZ, P. M. PINSKY AND R. L. TAYLOR, *Unconditionally stable element-by element algorithms for dynamic problems*, Report No. UCB/SESM-82/01, Dept. Civil Engineering, Univ. California, Berkeley, 1982.
- [10] A. R. GOURLAY, *Splitting methods for time dependent partial differential equations*, in The State of the Art in Numerical Analysis, D. Jacobs, ed., Academic Press, New York, 1977.
- [11] B. NOUR-OMID AND B. N. PARLETT, *Element preconditioning*, Report No. PAM-103, Center for Pure and Applied Mathematics, Univ. California, Berkeley, Oct. 1982.

## GLOBAL EXTRAPOLATION OF A FIRST ORDER SPLITTING METHOD\*

J. G. VERWER<sup>†</sup> AND H. B. DE VRIES<sup>‡</sup>

**Abstract.** This note deals with the numerical solution of multi-space dimensional parabolic partial differential equations and advocates classic global Richardson extrapolation for splitting methods. The paper concentrates on the first order locally one-dimensional splitting method. This robust, low order splitting method is known to possess two favourable properties. It rapidly damps high frequency components, which is of importance for problems having nonsmooth initial data, and it possesses excellent stability properties for nonlinear problems. Global extrapolation to higher order leaves these properties invariant. In addition, global extrapolation is easy to implement. A comparison is made with a local extrapolation procedure which has been proposed by Lawson and Morris (SIAM J. Numer. Anal., 15 (1978), pp. 1212–1224). A few numerical results are reported.

**AMS(MOS) 1980 subject classifications.** 65L05, 65M05, 65M20

**1982 CR categories.** 5.17

**Key words.** numerical analysis, parabolic partial differential equations, splitting methods, method of lines, global extrapolation

**1. Introduction.** In the numerical solution of parabolic partial differential equations with nonsmooth initial data, it is desirable to employ a time discretization which, in a sufficient manner, simulates the rapid exponential decay of high frequency components. For example, when a discontinuity exists between the initial function and the boundary conditions. Within the class of splitting methods for multi-space dimensional problems, the first order locally one-dimensional method (LOD method, cf. [17]) possesses this damping property. A disadvantage of this method is its low accuracy in time. To increase the accuracy, Lawson and Morris [9] have proposed a local extrapolation of the LOD method to order two which maintains the rapid damping of high frequency components.<sup>1</sup> This locally extrapolated LOD scheme requires in general twice as many operations per step as the basic LOD scheme. However, a numerical experiment [9] on a model heat equation in two space dimensions with a discontinuity between the initial and boundary conditions, has shown that local extrapolation may pay off. In particular, for such problems the Lawson–Morris scheme will perform better than the second order Peaceman–Rachford scheme due to a lack of damping of high frequency components in the latter one.

We advocate an alternative extrapolation of the LOD method, viz. global Richardson extrapolation. This type of Richardson extrapolation, which is classic in the numerical solution of ordinary differential equations [5, p. 81], involves parallel integration with the same basic scheme on different time grids, but completely separated. Consequently, all stability and damping properties of the basic scheme are left invariant by global extrapolation, contrary to local extrapolation. Global extrapolation is easy to implement. For the LOD method it is also less expensive than the Lawson–Morris extrapolation. Global extrapolation to order two requires one and a half-times as many operations per step as the basic scheme. For global extrapolation to order three this factor is equal to  $\frac{7}{4}$  or 2, depending on the implementation.

---

\*Received by the editors March 29, 1983, and in revised form May 17, 1984.

<sup>†</sup>Centre for Mathematics and Computer Science, Foundation Mathematical Centre, Kruislaan 413, 1098 SJ Amsterdam, the Netherlands.

<sup>‡</sup>Fokker, B.V., Dept. RAMS, S018-29, 1117 ZJ Schiphol Oost, the Netherlands.

<sup>1</sup>As a sequel to [9] Gourlay and Morris have published [3] and [4] where they develop new schemes of orders 2, 3 and 4 which also rapidly damp high frequency components. However, in contrast to those of [9], no splitting versions of their new schemes were given and, as far as we know, the question of applying them in a splitting context has not been discussed in the literature.

Recent alternative approaches to increasing the accuracy of splitting methods include the application of defect correction techniques [7], [8], [14]. A common property of these techniques with the local extrapolation technique of Lawson and Morris [9], is that the accuracy is increased by some local procedure. For splitting methods such local procedures always seem to interfere with the requirement of unconditional stability and, particularly, with rapid damping of high frequency components. In this respect, the present approach differs in principle. By global extrapolation the accuracy is increased in a global way and by no means influences the stepwise stability of the solution process. This latter point is our main motive to advocate global extrapolation. Furthermore, the technique is simple and can be applied to any one-step splitting method for time-dependent, multi-space dimensional problems.

In this note we concentrate on the LOD method. Apart from its well-known damping properties, which is an essential requirement in [9], this method also possesses excellent stability properties for nonlinear problems [15]. Global extrapolation does not interfere with these nonlinear stability properties either. Furthermore, the LOD method is not restricted to two space dimensions; it is equally applicable to two- and three-space dimensional problems.

**2. The LOD method.** In this section we briefly recall the LOD method [17]. By following the method of lines approach, the LOD method can be formulated in a very compact way (cf. [6]). Let the initial value problem for the ordinary differential system

$$(2.1) \quad \dot{y}(t) = f(t, y(t)), \quad t > 0, \quad y(0) = y_0$$

represent a semi-discrete version of a given initial-boundary value problem for a partial differential equation. For the moment it will not be necessary to be specific about the partial differential equation and the space discretization. We only assume that the vector function  $f(t, y)$  can be written as

$$(2.2) \quad f(t, y) = \sum_{i=1}^k f_i(t, y),$$

where  $f_i$  corresponds to a one-space dimensional partial differential operator. The following time integration formula:

$$(2.3) \quad \begin{aligned} y_{n+1}^{(0)} &= y_n, \\ y_{n+1}^{(i)} &= y_{n+1}^{(i-1)} + \tau f_i(t_{n+1}^{(i)}, y_{n+1}^{(i)}), \quad i = 1, \dots, k, \\ y_{n+1} &= y_{n+1}^{(k)}, \end{aligned}$$

then defines the LOD step  $y_n \rightarrow y_{n+1}$ . Here,  $y_n$  is the approximation to  $y(t)$ , the exact solution of (2.1), at time  $t = t_n$ , and  $\tau = t_{n+1} - t_n$  is the timestep. Further, we suppose  $t_n \leq t_{n+1}^{(i)} \leq t_{n+1}$ ,  $i = 1, \dots, k$ .

In applications  $k$  is normally equal to 2 or 3, being the space dimension of the partial differential equation. In §4 we shall give a numerical example of a two- and three-space dimensional problem. Note that we formulate the splitting method directly for time-dependent and, possibly, nonlinear problems.

The order of consistency of (2.3) is equal to one for all splitting functions  $f_i$ ,  $i = 1, \dots, k$ , satisfying (2.2). Observe that (2.3) consists of  $k$  consecutive backward Euler steps, each of which applied with a different function  $f_i$ . The computation of the vectors  $y_{n+1}^{(i)}$  from the implicit backward Euler relations can be performed cheaply using a Newton type iteration, because we assumed that  $f_i$  stands for a semi-discrete, one-space dimensional partial differential operator.

The LOD method is known to be unconditionally stable for the linear model problem

$$(2.4) \quad U_t = \sum_{i=1}^k U_{x_i x_i},$$

where the second order derivative has been replaced by the central difference operator. Furthermore, the method possesses excellent damping properties for high frequency solution components (see e.g. [9]). In addition to these linear stability properties, the LOD method can also be shown to be unconditionally stable for nonlinear parabolic problems of the form

$$(2.5) \quad U_t = \sum_{i=1}^k F_i \left( t, x, U, \frac{\partial}{\partial x_i} \left( p_i(t, x) \frac{\partial U}{\partial x_i} \right) \right), \quad x \in \Omega \subset \mathbb{R}^k.$$

This nonlinear stability result can be shown by exploiting the intimate relation with the backward Euler scheme. For details we refer to [15].

To conclude our description of the LOD method we have to recall two possible sources of inaccuracies, viz. nonconstant boundary values and nonconstant inhomogeneous terms [2], [12], [16]. These inaccuracies will also be noticeable for locally and globally extrapolated results (see Experiment 2 of §5). Other splitting methods, such as ADI, also suffer from these phenomena, although to a somewhat lesser extent.

**3. Global Richardson extrapolation.** The LOD method (2.3) may be considered as a particular one-step integration method of order of consistency  $p = 1$  for the ordinary differential system (2.1). Suppose that a  $p$ th order one-step method is applied from  $t_0 = 0$  up to  $t_N = T$ , using a time grid  $G_N$ . This grid does not need to be uniform. It is only assumed that for  $N$  sufficiently large, the minimal and maximal stepsizes  $\tau$  behave like  $O(N^{-1})$ . If this natural assumption is satisfied, we are assured of the existence of an asymptotic expansion in the maximal stepsize,  $\tau_N$  say, for the global error [13]

$$(3.1) \quad \varepsilon_N = y_N - y(t_N).$$

If we let  $f$  be  $M$  times differentiable, in some neighbourhood of the exact solution (2.1), then functions  $e_j, j = p, \dots, M$ , exist, independent of  $\tau_N$ , such that

$$(3.2) \quad \varepsilon_N = \sum_{j=p}^M \tau_N^j e_j(t_N) + O(\tau_N^{M+1}), \quad \tau_N \rightarrow 0.$$

The existence of this asymptotic expansion for  $\varepsilon_N$  forms the basis for global Richardson extrapolation.

The use of this technique for estimating the global error of one-step integration methods for ordinary differential equations is classic (see [5, p. 81] and [13, p. 157]), although it is not very often applied [1], [10], [11]. As far as we know, the possibility of using this technique for increasing the accuracy of low order splitting methods has not yet been discussed in the literature.

Global extrapolation is easy to implement. It involves parallel integration with the same method on different grids  $G_N$ . Let us consider the coherent grids  $G_N, G_{2N}$  and  $G_{3N}$  depicted in Fig. 1.  $G_{2N}$  is obtained from  $G_N$  by halving all stepsizes, etc. Because of this coherence between the grids, expansion (3.2) holds for  $\tau_N, \tau_{2N} = \tau_N/2$  and  $\tau_{3N} = \tau_N/3$ , at all common gridpoints, i.e., on the whole of  $G_N$ . Let  $y_{n,i}$  denote the approximation to  $y(t_n)$  at the grid  $G_{iN}$ . Then, at all common points,

$$(3.3) \quad \varepsilon_{n,i} \equiv y_{n,i} - y(t_n) = \sum_{j=1}^M (\tau_N/i)^j e_j(t_n) + O(\tau_N^{M+1}), \quad \tau_N \rightarrow 0,$$

if  $p = 1$ . Next suppose  $M$  sufficiently large and compute

$$(3.4) \quad \begin{aligned} y_n^{[2]} &\equiv 2y_{n,2} - y_{n,1}, \\ y_n^{[3]} &\equiv \frac{9}{2} y_{n,3} - 4y_{n,2} + \frac{1}{2} y_{n,1}. \end{aligned}$$

Then

$$(3.5) \quad \begin{aligned} y_n^{[2]} &= y(t_n) - \frac{1}{2} \tau_N^2 e_2(t_n) + O(\tau_N^3), \\ y_n^{[3]} &= y(t_n) + \frac{1}{6} \tau_N^3 e_3(t_n) + O(\tau_N^4), \end{aligned}$$

showing that  $y_n^{[2]}$  and  $y_n^{[3]}$  are of order of convergence two and three, respectively.

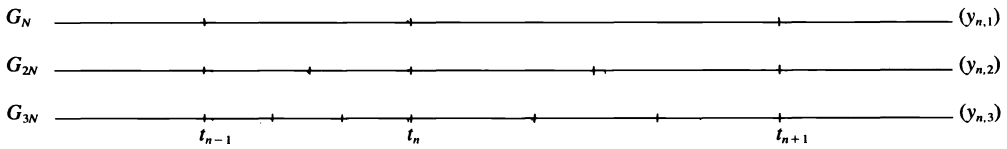


FIG. 1. Three coherent grids.

It is emphasized that the extrapolation is passive, i.e. the integrations are performed independently of each other. This trivially implies that global extrapolation cannot interfere with the stability of the basic scheme.

Observe that it is theoretically possible to extrapolate to arbitrarily high order of convergence. We restrict ourselves to orders two and three, assuming that this is sufficiently high for partial differential equations. When compared with the result  $y_{n,2}$  computed at the grid  $G_{2N}$ , the computation of  $y_n^{[2]}$  requires an additional computational effort of 50%. Global extrapolation to order three requires twice as many operations per step as the basic scheme on  $G_{3N}$ . When using the grids  $G_N$ ,  $G_{2N}$  and  $G_{4N}$ , the corresponding factor is equal to  $\frac{1}{4}$ . We prefer to use  $G_N$ ,  $G_{2N}$  and  $G_{3N}$  in order to avoid a too large difference between the stepsizes.

**4. Comparison with the Lawson–Morris extrapolation scheme.** Lawson and Morris [9] restrict their investigations to constant coefficient, homogeneous linear semi-discrete systems

$$(4.1) \quad \dot{y} = Ay = \sum_{i=1}^k A_i y,$$

where  $1 \leq k \leq 3$ . For this linear problem, the LOD scheme (2.3) reduces to

$$(4.2) \quad y_{n+1} = \prod_{i=k}^1 (I - \tau A_i)^{-1} y_n.$$

The second order, locally extrapolated LOD scheme of [9] is given by

$$(4.3) \quad \begin{aligned} z_0 &= \prod_{i=k}^1 (I - \tau A_i)^{-1} y_n, & z_1 &= \prod_{i=1}^k (I - \tau A_i)^{-1} z_0, \\ z_2 &= \prod_{i=k}^1 (I - 2\tau A_i)^{-1} y_n, & z_3 &= \prod_{i=1}^k (I - 2\tau A_i)^{-1} y_n, \\ y_{n+2} &= 2z_1 - \frac{1}{2}(z_2 + z_3). \end{aligned}$$

This scheme performs the step  $y_n \rightarrow y_{n+2}$  over an interval of length  $2\tau$  by performing four basic LOD steps. Hence, per interval of length  $\tau$ , it requires twice as many operations as the basic scheme. Lawson and Morris have defined their extrapolation in such a way that for the linear model (4.1) associated to (2.4) (i) the scheme is unconditionally stable, and (ii) the damping of the basic scheme is maintained. In fact, the preservation of damping has been their main concern.

*Remark.* The reader should observe the symmetrization in method (4.3), i.e., the order in which the matrices  $A_i$  appear alternates. Lawson and Morris do not motivate their use of symmetrization. However, it benefits the accuracy and stability. We also remark that symmetrization is not necessary for obtaining order two. Further, if the matrices commute, which they do in the stability analysis, we get  $y_{n+2} = 2z_1 - z_2$ , i.e., straightforward local Richardson extrapolation to order two. Hence in this case the extrapolation requires  $\frac{3}{2}$  times as many operations. If the matrices do commute we use this work estimate for scheme (4.3) rather than the factor 2 mentioned above.

The second order, local extrapolation scheme (4.3) can be extended to the general problem cases (2.1)-(2.2). Introduce the formal notation  $y_{n+1} = E[\tau, t_n, y_n, f_1, \dots, f_k]$  for the LOD formula (2.3). The extension then reads

$$\begin{aligned}
 (4.4) \quad z_0 &= E[\tau, t_n, y_n, f_1, \dots, f_k], & z_1 &= E[\tau, t_{n+1}, z_0, f_k, \dots, f_1], \\
 z_2 &= E[2\tau, t_n, y_n, f_1, \dots, f_k], & z_3 &= E[2\tau, t_n, y_n, f_k, \dots, f_1], \\
 y_{n+2} &= 2z_1 - \frac{1}{2}(z_2 + z_3).
 \end{aligned}$$

A straightforward Taylor expansion shows that this algorithm is of second order for any choice of intermediate time points  $t_{n+1}^{(i)}$  in (2.3).

To demonstrate their scheme, Lawson and Morris [9] computed the solution to the problem

$$\begin{aligned}
 (4.5) \quad U_t &= U_{x_1 x_1} + U_{x_2 x_2}, & t > 0, & \quad 0 < x_1, x_2 < 2, \\
 U(0, x_1, x_2) &= \sin\left(\frac{\pi x_2}{2}\right), & 0 \leq x_1, x_2 \leq 2, \\
 U(t, x_1, x_2) &= 0, & t > 0, & \quad x_1, x_2 \text{ on the boundary.}
 \end{aligned}$$

The Fourier solution of this problem is given by

$$(4.6) \quad U(t, x_1, x_2) = \sin\left(\frac{\pi x_2}{2}\right) \sum_{n=1}^{\infty} [1 - (-1)^n] \frac{2}{n\pi} \sin\left(\frac{n\pi x_1}{2}\right) \exp\left(-\frac{\pi^2}{4}(n^2 + 1)t\right).$$

Because of the discontinuity between the initial function and the boundary function, this problem should be integrated with a method which rapidly damps high frequency components.

Following Lawson and Morris [9], we also compute the solution to this problem, at  $t = 1$ , using the first order LOD scheme, the second order Lawson–Morris scheme, the second order ADI scheme of Peaceman–Rachford, and global extrapolation to order two and three applied to the LOD scheme. The spatial discretization is based on the standard 5-point finite difference operator on a uniform grid of stepsize  $h$ . In Table 2 we show the maximum of the absolute errors for some values of  $\tau$  and  $h$ . It should be noted that at  $t = 1$  the theoretical solution has a maximum value of approximately 0.01. We refer to [9] for a plot of this two-dimensional function. The values of  $\tau$  given in the table belong to the finest time grids. In the column “Work” we have expressed the total computational effort per  $\tau$ -interval into the computational effort of the LOD scheme.

TABLE 2  
Maximum absolute errors in solving problem (4.5) at  $t = 1$ .

Method	Work	$\tau = 1/12$			$\tau = 1/24$		
		$h = 0.1$	$h = 0.05$	$h = 0.025$	$h = 0.1$	$h = 0.05$	$h = 0.025$
LOD	1	$5.3_{10^{-3}}$	$5.2_{10^{-3}}$	$5.2_{10^{-3}}$	$2.5_{10^{-3}}$	$2.5_{10^{-3}}$	$2.5_{10^{-3}}$
Peaceman-Rachford	$3/2$	$2.3_{10^{-3}}$	$1.8_{10^{-2}}$	$4.1_{10^{-2}}$	$3.4_{10^{-5}}$	$2.3_{10^{-3}}$	$1.8_{10^{-2}}$
Lawson-Morris	$3/2$	$7.5_{10^{-4}}$	$6.9_{10^{-4}}$	$6.7_{10^{-4}}$	$3.0_{10^{-4}}$	$2.4_{10^{-4}}$	$2.3_{10^{-4}}$
Global extrap. (2)	$3/2$	$8.4_{10^{-4}}$	$9.0_{10^{-4}}$	$9.2_{10^{-4}}$	$1.8_{10^{-4}}$	$2.3_{10^{-4}}$	$2.5_{10^{-4}}$
Global extrap. (3)	2	$5.5_{10^{-5}}$	$1.1_{10^{-4}}$	$1.2_{10^{-4}}$	$5.9_{10^{-5}}$	$3.1_{10^{-6}}$	$1.1_{10^{-5}}$

It can be concluded that for problem (4.5), which serves as a test example for problems with nonsmooth initial data, the third order global extrapolation scheme is to be preferred to the other schemes. For example, the globally extrapolated third order results for  $\tau = 1/12$  are more accurate than the second order extrapolated results for  $\tau = 1/24$ . Note the somewhat awkward error behaviour of the third order results for the three different values of  $h$ . It seems plausible to owe this to interference of space and time errors of nearly equal magnitude. We recall that Table 2 shows full discretization errors, whereas the implemented extrapolations only deal with the time integration. We also emphasize that the very minor increasing error behaviour of the global extrapolation methods for decreasing  $h$  does not occur systematically. See Tables 3a, b for comparison. Finally it should be observed that the ADI scheme yields relatively large errors when  $h$  decreases. This is caused by a lack of damping of high frequency components. The LOD type schemes do not suffer from this phenomenon.

TABLE 3a  
Maximum absolute errors in solving problem (4.7) at  $t = 1$  for  $\gamma = 1/6$ .

Method	Work	$\tau = 1/12$			$\tau = 1/24$		
		$h = 0.1$	$h = 0.05$	$h = 0.025$	$h = 0.1$	$h = 0.05$	$h = 0.025$
LOD	1	$3.7_{10^{-3}}$	$3.4_{10^{-3}}$	$3.4_{10^{-3}}$	$1.9_{10^{-3}}$	$1.7_{10^{-3}}$	$1.7_{10^{-3}}$
Lawson-Morris	$3/2$	$6.3_{10^{-4}}$	$4.0_{10^{-4}}$	$3.4_{10^{-4}}$	$4.1_{10^{-4}}$	$1.8_{10^{-4}}$	$1.2_{10^{-4}}$
Global extrap. (3)	2	$2.5_{10^{-4}}$	$3.7_{10^{-5}}$	$3.0_{10^{-5}}$	$2.9_{10^{-4}}$	$6.9_{10^{-5}}$	$1.3_{10^{-5}}$

TABLE 3b  
Time integration errors at the gridpoint  $(1/2, 1/2, 1/2)$ .

	$\tau = 1/12$	$\tau = 1/24$
LOD	$-3.4_{10^{-3}}$	$-1.6_{10^{-3}}$
Lawson-Morris	$-3.3_{10^{-4}}$	$-1.0_{10^{-4}}$
Global extrap. (3)	$5.0_{10^{-5}}$	$6.0_{10^{-6}}$

As a further illustration we also compute the solution to the following three-dimensional version of problem (4.5):

$$\begin{aligned}
 (4.7) \quad & U_t = \gamma(U_{x_1x_1} + U_{x_2x_2} + U_{x_3x_3}), \quad t > 0, \quad 0 < x_1, x_2, x_3 < 1, \\
 & U(0, x_1, x_2, x_3) = \sin(\pi x_2)\sin(\pi x_3), \quad 0 \leq x_1, x_2, x_3 \leq 1, \\
 & U(t, x_1, x_2, x_3) = 0, \quad t > 0, \quad x_1, x_2, x_3 \text{ on the boundary.}
 \end{aligned}$$

The Fourier solution is given by

$$(4.8) \quad U(t, x_1, x_2, x_3) = \sin(\pi x_2) \sin(\pi x_3) \sum_{n=1}^{\infty} [1 - (-1)^n] \frac{2}{n\pi} \sin(n\pi x_1) \exp(-\gamma\pi^2(n^2 + 2)t).$$

Again we have a discontinuity between the initial and boundary function. The spatial discretization is based on the standard 7-point finite difference operator on a uniform grid of stepsize  $h$ . Table 3a shows the maximum of the absolute errors for some values of  $\tau$  and  $h$  at  $t = 1$  for three LOD schemes using  $\varphi = 1/6$ . For this value of  $\gamma$  the solution (4.8) has a maximum value of approximately 0.01 at  $t = 1$ . The column ‘‘Work’’ has the same meaning as in Table 2.

Again we may conclude that it pays to apply extrapolation. It was found that in all cases the maximum absolute error appeared near the point  $(1/2, 1/2, 1/2)$ . For this grid point the space errors (fully continuous solution — semi-discrete solution) are  $-3.0_{10^{-4}}$ ,  $-7.4_{10^{-5}}$  and  $-1.9_{10^{-5}}$ . For all three methods the corresponding time errors (semi-discrete solution — fully discrete solution) were found to be independent of  $h$ . They are listed in Table 3b. This table shows the performance of the extrapolation procedures more clearly than Table 3a.

**5. Two more numerical experiments.** In addition to the previous experiments, we discuss two more experiments in order to give some more insight into the use of the extrapolation procedures. For this purpose we consider the first initial-boundary value problem for the simple linear problems

$$(5.1) \quad \begin{aligned} U_t &= \frac{x_1 - x_1^2}{4} U_{x_1 x_1} + \frac{x_2 - x_2^2}{4} U_{x_2 x_2}, & t > 0, & \quad 0 < x_1, x_2 < 1, \\ U(t, x_1, x_2) &= 1 - e^{-t}(x_1^2 - x_1)(x_2^2 - x_2), & t \geq 0, & \quad 0 \leq x_1, x_2 \leq 1, \end{aligned}$$

and

$$(5.2) \quad \begin{aligned} U_t &= U_{x_1 x_1} + U_{x_2 x_2} - e^{-t}(x_1^2 + x_2^2 + 4), & t > 0, & \quad 0 < x_1, x_2 < 1, \\ U(t, x_1, x_2) &= 1 + e^{-t}(x_1^2 + x_2^2), & t \geq 0, & \quad 0 \leq x_1, x_2 \leq 1. \end{aligned}$$

For the space discretization we use again standard finite differences on a uniform grid with meshwidth  $h$ . Note that, due to the solutions selected, the space discretization is exact. So we can illustrate the effect of the extrapolations without interference of space discretization errors. Note also that the selected solutions are smooth, i.e., free of high frequency components. This implies that here the ADI scheme could also successfully be applied, possibly in combination with global extrapolation.

We have applied

(i) The first order LOD scheme

$$(5.3) \quad y^* = y_n + \tau f_1(t_n + \tau, y^*), \quad y_{n+1} = y^* + \tau f_2(t_n + \tau, y_{n+1}).$$

(ii) Global extrapolation on this scheme to order three.

(iii) The extension (4.4) of the Lawson–Morris scheme (4.3), where  $E$  now represents (5.3).

(iv) The second order Peaceman–Rachford ADI scheme

$$(5.4) \quad \begin{aligned} y^* &= y_n + \frac{\tau}{2} f_1\left(t_n + \frac{1}{2}\tau, y^*\right) + \frac{\tau}{2} f_2(t_n, y_n), \\ y_{n+1} &= y^* + \frac{\tau}{2} f_1\left(t_n + \frac{1}{2}\tau, y^*\right) + \frac{\tau}{2} f_2(t_{n+1}, y_{n+1}). \end{aligned}$$



- (v) Global extrapolation on this ADI scheme to order four using the grids of Fig. 1. The extrapolation formula to order four reads

$$y_n^{(4)} = \frac{27}{12}y_{n,3} - \frac{4}{3}y_{n,2} + \frac{1}{12}y_{n,1}.$$

*Experiment 1.* This experiment deals with the homogeneous problem (5.1). We emphasize that the solution is constant at the boundary. Space discretization leads to the splitting functions

$$(5.5) \quad f_1(t, y) = A_1y + B_1, \quad f_2(f, y) = A_2y + B_2,$$

where  $B_1$  and  $B_2$  are sparse vectors containing the constant boundary values. The meaning of  $A_1$  and  $A_2$  should be self-evident. Because of the fact that the present semidiscrete problem is of constant coefficient type we in fact really apply the original algorithm of Lawson and Morris, as the extension of their algorithm to the inhomogeneous form is not essential.

Table 4 shows maximum absolute errors at  $t = 1$  for three values of  $\tau$  using  $h = 1/40$ . The  $\tau$ -values correspond to the finest grid. Observe that the LOD scheme (5.3) and the ADI scheme (5.4) were applied using half the value of  $\tau$  given in the table. The column ‘‘Work’’ has the same meaning as in Table 2.

TABLE 4  
Maximum absolute errors in solving problem (5.1) at  $t = 1$ .

Method	Work	$\tau = 1/6$	$\tau = 1/12$	$\tau = 1/24$
LOD (applied with $\tau/2$ )	2	$4.7_{10^{-4}}$	$2.4_{10^{-4}}$	$1.2_{10^{-4}}$
Lawson–Morris	2	$8.0_{10^{-5}}$	$2.3_{10^{-5}}$	$6.2_{10^{-6}}$
Global extrap. LOD (3)	2	$4.9_{10^{-6}}$	$6.9_{10^{-7}}$	$9.2_{10^{-8}}$
ADI (applied with $\tau/2$ )	3	$3.3_{10^{-6}}$	$8.3_{10^{-7}}$	$2.1_{10^{-7}}$
Global extrap. ADI (4)	3	$2.3_{10^{-8}}$	$1.4_{10^{-9}}$	$8.8_{10^{-11}}$

For the present problem it can be concluded that, due to the smooth solution, the homogeneity, and the constant boundary values, all methods perform relatively accurately. In particular, it pays to apply extrapolation. Also observe that the order of convergence of all time integration methods shows up clearly (the space discretization is exact).

*Experiment 2.* The second experiment deals with the inhomogeneous problem (5.2), whose solution has time-dependent boundary values. This experiment serves to illustrate the effect of the extrapolation procedures in the presence of time-dependent boundary values and time-dependent source terms. For all splitting methods these dependencies are known to reduce the accuracy of the time integration [2], [12], [16]. Space discretization yields the splitting functions

$$(5.6) \quad f_1(t, y) = A_1y(t) + B_1(t) + 1/2V(t), \quad f_2(t, y) = A_2y(t) + B_2(t) + 1/2V(t),$$

where  $V(t)$  originates from the source term of (5.2) and  $A_1, A_2$  and  $B_1(t), B_2(t)$  have the same meaning as in Experiment 1. The results for  $h = 1/40$  are given in Table 5. Observe that we now apply an extension of the original Lawson–Morris algorithm (4.3).

It is striking that all LOD methods are significantly less accurate than in the previous experiment, although the solution of problem (5.2) has the same ‘‘smoothness properties’’ as the solution of problem (5.1). For the LOD methods the order of convergence shows up insufficiently for problem (5.2). Here we must conclude that the LOD extrapolation procedures are hardly more efficient than their respective basic schemes. The errors of the

TABLE 5  
Maximum absolute errors in solving problem (5.2) at  $t = 1$ .

Method	Work	$\tau = 1/6$	$\tau = 1/12$	$\tau = 1/24$
LOD (applied with $\tau/2$ )	2	$1.2_{10^{-2}}$	$7.8_{10^{-3}}$	$4.5_{10^{-3}}$
Lawson–Morris extension	2	$2.3_{10^{-2}}$	$1.5_{10^{-2}}$	$8.7_{10^{-3}}$
Global extrap. LOD (3)	2	$9.8_{10^{-3}}$	$4.7_{10^{-3}}$	$2.0_{10^{-3}}$
ADI (applied with $\tau/2$ )	3	$2.6_{10^{-6}}$	$6.6_{10^{-7}}$	$1.6_{10^{-7}}$
Global extrap. ADI (4)	3	$3.8_{10^{-6}}$	$2.2_{10^{-7}}$	$1.3_{10^{-8}}$

Lawson–Morris extension are even larger than the LOD ( $\tau/2$ ) errors. Finally, the ADI schemes perform very satisfactorily for the present problem. Their time integration errors are rather small while their order of convergence in time is clearly visible.

As already remarked at the end of §2 the disappointing behaviour of the LOD algorithms lies in the time dependencies of the boundary values and the source term. ADI algorithms also suffer from these dependencies, although to a lesser extent. The decrease of accuracy can be understood from a study of the error of approximation, i.e., the defect which is obtained by substituting an exact and smooth solution into the numerical scheme. When expanding this error in the time step  $\tau$ , any splitting gives rise to terms which cannot be combined to a higher derivative of the smooth solution, a situation which is not encountered in the application of fully implicit schemes such as backward Euler or implicit midpoint. These terms, finite difference expressions multiplied by  $h^{-2}$ , thus may be relatively large when compared with derivatives of an exact solution. It turns out that these terms are clearly present when the problem has time-dependent boundary values and/or source terms (for details, see [2], [12], [16]). The fact that ADI methods suffer less than LOD from this phenomenon can be explained from the consistency of the intermediate level.

It is also worthwhile to realize that the accuracy of a splitting scheme may depend significantly on the splitting of the source term. To illustrate this we have given Table 6 which shows some results of the LOD method applied to problem (5.2), when using the splitting

$$(5.7) \quad \begin{aligned} f_1(t, y) &= A_1 y(t) + B_1(t) + \alpha V(t), \\ f_2(t, y) &= A_2 y(t) + B_2(t) + (1 - \alpha)V(t), \end{aligned}$$

for  $\alpha = 0, 1/2, 1$ . The correct splitting of  $V(t)$  appears to be the one we have used in Table 5, at least for problem (5.2). It is obvious of course that the differences shown in Table 6 penetrate into the extrapolations. For the ADI method one will find even larger differences in accuracy for  $\alpha = 0, 1/2, 1$ .

Finally, we should like to observe that inaccuracies caused by time-dependent boundary values can be removed by applying the Fairweather–Mitchell boundary value correction [2],

TABLE 6  
Maximum absolute errors in solving problem (5.2) at  $t = 1$  using the LOD method for three different splittings (5.7). The mesh width  $h = 1/40$ .

	$\tau = 1/5$	$\tau = 1/10$	$\tau = 1/20$	$\tau = 1/40$	$\tau = 1/80$
$\alpha = 0$	$9.5_{10^{-2}}$	$6.9_{10^{-2}}$	$4.5_{10^{-2}}$	$2.6_{10^{-2}}$	$1.4_{10^{-2}}$
$\alpha = 1/2$	$1.8_{10^{-2}}$	$1.3_{10^{-2}}$	$8.9_{10^{-3}}$	$5.2_{10^{-3}}$	$2.9_{10^{-3}}$
$\alpha = 1$	$5.9_{10^{-2}}$	$4.3_{10^{-2}}$	$2.8_{10^{-2}}$	$1.6_{10^{-2}}$	$8.5_{10^{-3}}$

TABLE 7  
 Maximum absolute errors in solving problem (5.2) at  $t = 1$  when using the boundary value correction technique and the splitting (5.6).

Method	Work	$\tau = 1/6$	$\tau = 1/12$	$\tau = 1/24$
LOD (applied with $\tau/2$ )	2	$2.2_{10^{-3}}$	$1.3_{10^{-3}}$	$7.0_{10^{-4}}$
Lawson-Morris	2	$2.2_{10^{-3}}$	$1.1_{10^{-3}}$	$5.0_{10^{-4}}$
Global extrap. LOD (3)	2	$1.3_{10^{-3}}$	$4.5_{10^{-4}}$	$1.1_{10^{-4}}$
ADI (applied with $\tau/2$ )	3	$2.7_{10^{-6}}$	$6.6_{10^{-7}}$	$1.6_{10^{-7}}$
Global extrap. ADI (4)	3	$1.3_{10^{-6}}$	$8.3_{10^{-8}}$	$4.7_{10^{-9}}$

[12], [16]. For problem (5.2) this is illustrated by the results shown in Table 7, although the influence of the source term is still visible (this table is to be compared with Table 5). In fact, the gain in accuracy of the extrapolated LOD schemes is still not up to our expectations. The accuracy of the ADI scheme (5.4) is not improved by applying the boundary value correction. On the other hand, the global extrapolation of the ADI scheme is benefited slightly by this correction technique.

#### REFERENCES

- [1] K. DEKKER AND J. G. VERWER, *Estimating the global error of Runge-Kutta approximation*, in Differential-Differenzengleichungen, Anwendungen und numerische Probleme, L. Collatz, G. Meinardus and W. Wetterling, eds., ISNM Series, Vol. 62, Birkhäuser, Basel-Boston-Stuttgart, 1983.
- [2] G. FAIRWEATHER AND A. R. MITCHELL, *A new computational procedure for ADI methods*, SIAM J. Numer. Anal., 4 (1967), pp. 163-170.
- [3] A. R. GOURLAY AND J. LL. MORRIS, *The extrapolation of first order methods for parabolic partial differential equations. II*, SIAM J. Numer. Anal., 17 (1980), pp. 641-655.
- [4] ———, *Linear combinations of generalized Crank-Nicolson schemes*, IMA J. Numer. Anal., 1 (1981), pp. 347-357.
- [5] P. HENRICI, *Discrete Variable Methods for Ordinary Differential Equations*, John Wiley, New York, 1962.
- [6] P. J. VAN DER HOUWEN AND J. G. VERWER, *One-step splitting methods for semi-discrete parabolic equations*, Computing, 22 (1979), pp. 291-309.
- [7] P. J. VAN DER HOUWEN, *Multistep splitting methods of high order for initial value problems*, SIAM J. Numer. Anal., 17 (1980), pp. 410-427.
- [8] ———, *Iterated splitting methods of high order for time dependent partial differential equations*, SIAM J. Numer. Anal., 21(1984), pp. 635-656.
- [9] J. D. LAWSON AND J. LL. MORRIS, *The extrapolation of first order methods for parabolic partial differential equations, I*, SIAM J. Numer. Anal. 15 (1978), pp. 1212-1224.
- [10] F. G. LETHER, *The use of Richardson extrapolation in one-step methods with variable stepsize*, Math. Comp. 20, (1966), pp. 379-385.
- [11] L. F. SHAMPINE AND H. A. WATTS, *Global error estimation for ordinary differential equations*, ACM Trans. Math. Software, 2 (1976), pp. 172-186.
- [12] B. P. SOMMEIJER, P. J. VAN DER HOUWEN AND J. G. VERWER, *On the treatment of time-dependent boundary conditions in splitting methods for parabolic differential equations*, Internat. J. Numer. Meth. Engrg., 17 (1981), pp. 335-346.
- [13] H. J. STEITTE, *Analysis of Discretization Methods for Ordinary Differential Equations*, Springer-Verlag, Berlin, 1973.
- [14] J. G. VERWER, *The application of iterated defect correction to the LOD method for parabolic equations*, BIT, 19 (1979), pp. 384-394.
- [15] ———, *Contractivity of locally one-dimensional splitting methods*, Numer. Math, 44 (1984), pp. 247-259.
- [16] YE. G. D'YAKONOV, *Some difference schemes for solving boundary problems*, USSR Comput. Math. Math. Phys., 1 (1963), pp. 55-77.
- [17] N. N. YANENKO, *The Method of Fractional Steps*, Springer-Verlag, Berlin, 1971.

## A UNIFICATION OF SOME SOFTWARE RELIABILITY MODELS\*

NAFTALI LANGBERG† AND NOZER D. SINGPURWALLA‡

**Abstract.** In this paper we show how several models used to describe the reliability of computer software can be comprehensively viewed by adopting a Bayesian point of view. We first provide an alternative motivation for a commonly used model, the Jelinski–Moranda model, using notions from shock models. We then show that some alternate models proposed in the literature can be derived by assigning specific prior distributions for the parameters of the above model. We also obtain other structural results such as stochastic inequalities and association, and discuss how these can be interpreted.

**Key words.** software reliability, Bayesian statistics, shock models, association, reliability theory

**1. Introduction, review of models, motivation, and summary.** The increasing demand for computers and computing services has brought about a great need for the study and solution of many computer related problems. In particular, the problem of estimating the reliability of computer software has, over the last few years, been receiving a great deal of attention.

The software segment of a computer system involves instructions or codes used to program the hardware system. Let  $N^*$  be the total number of distinct “input types” to the software system;  $N^*$  is assumed to be large, conceptually infinite. By an input type we mean a specific type of a job, data set, or function, which the software system is required to undertake. Let  $N \leq N^*$  be the number of input types which result in the inability of the software system to perform its desired function;  $N$  is assumed *unknown*. Such input types lead to what will be termed as *software failures*. Software failures are either due to errors in the logic of the instructions, errors in the coding of the instructions (the program), or an input that is incompatible with the design of the software system. We assume that the processing of an input (if successful) is instantaneous; this implies that there is no queueing of inputs at the system. We also allow for the possibility that the same type of input can arrive at the software system over and over again. That is, as is typical, the software could be called upon to perform the same type of job more than once. When a software failure occurs, a diagnosis is made of the cause of the failure, and one of the following two actions is taken:

- (a) an error(s) in either the logic or the coding, or both, is (are) corrected, so that now  $N$  is reduced by one (or more); this input type presents no future problem; or
- (b) if it is determined that the input is incompatible with the software, then it is eliminated from further consideration by an appropriate modification of the software.

Given  $t_1, \dots, t_k$ ,  $k \leq N$ , the times *between* failures of the software system, a problem is to estimate  $(N - k)$ , the number of input types which would lead to future software failures. By “time” we mean here “execution time.”

One of the earliest, and certainly the most referenced, models for describing the stochastic failure of software, is that proposed by Jelinski and Moranda (1972)—henceforth J-M. Here, attention is focussed on the  $N$  input types which lead to software failures; these inputs are viewed as faults (bugs) in the program. The following assumptions are made:

---

\*Received by the editors August 2, 1982, and in revised form February 13, 1984. This research was supported by the Office of Naval Research under contract N00014-77-C-0263, Project NR-042-372, and by the Army Research Office under grant DAAG 29-84-K-0160.

†Tel Aviv University, Tel Aviv, Israel.

‡Department of Operations Research and Department of Statistics, George Washington University, Washington, DC 20052.

(i) The failure rate of the software at any point in time is proportional to the residual number of faults in the program; the program begins life with  $N$  faults.

(ii) Each of the  $N$  faults contributes an *equal* amount, say  $\Lambda$  (unknown), to the failure rate.

(iii)  $t_1, \dots, t_N$ , the times between successive failures of the program, are judged independent, given  $\Lambda$  and  $N$ . Thus the conditional density of  $T_i$ ,  $i = 1, \dots, N$ , is

$$(1.1) \quad f(t_i | N, \Lambda) = \Lambda(N - i + 1) \exp\{-\Lambda(N - i + 1)t_i\},$$

with  $N$  and  $\Lambda$  being the unknown parameters.

Given  $t_1, \dots, t_k$ ,  $k \leq N$ , Jelinski and Moranda, among others, apply the method of maximum likelihood to estimate  $N$  and  $\Lambda$ .

To date, it appears that there are two issues that are critical of this model. These have been emphasized by Littlewood (1981); they are:

(i) the "bug counting" framework from which the model is derived is regarded as being naive—in particular, assumption (ii) above, which states that every fault has the same effect on the overall failure rate, has been challenged;

(ii) the standard "sample theory" approach of maximum likelihood estimation (MLE) of the parameters of this model has often produced misleading answers (Forman and Singpurwalla (1977), (1979)).

In response to (i) above, alternate models have been proposed. Those by Littlewood and Verrall (1973), henceforth L-V, and Goel and Okumoto (1979) have proved to be of practical value.

Arguing that different portions of the program code are exercised with varying frequencies, and that there is some uncertainty in the removal of a fault, L-V whilst retaining the assumption that  $T_i$ ,  $i = 1, 2, \dots$ , are conditionally independent with density

$$(1.2) \quad f(t_i | \Lambda_i) = \Lambda_i \exp(-\Lambda_i t_i)$$

require that the  $\Lambda_i$ 's be independent with density

$$(1.3) \quad f(\lambda_i | \psi(i), \alpha) = \frac{\psi(i) \{\psi(i) \lambda_i\}^{\alpha-1} \exp\{-\psi(i) \lambda_i\}}{\Gamma(\alpha)}.$$

To describe "reliability growth," which is expected to occur from the debugging process, the parameter  $\psi(i)$  is taken to be an increasing function of  $i$ ,  $\beta_0 + \beta_1 i$ . This choice of  $\psi(i)$  ensures that the sequence  $\{\Lambda_i\}$  stochastically increases in  $i$ . In L-V, the parameters  $\beta_0$  and  $\beta_1$  are estimated using maximum likelihood, whereas  $\alpha$  is estimated via a Bayesian argument. Littlewood (1981) uses some real life data on software failures to show that the L-V model reasonably well describes his data, and uses this as empirical evidence in support of the model.

Goel and Okumoto (1979) consider  $M(t)$ , the cumulative number of software failures in time  $t$ , and make some other assumptions to show that  $\{M(t); t \geq 0\}$  is a nonhomogeneous Poisson process with a mean value function  $a(1 - e^{-bt})$ ;  $a$  is the expected number of faults to be eventually detected, and  $b$  is a constant of proportionality for which no physical meaning has been given. By contrast with the J-M model, this formulation treats  $N$  as a random variable with expectation  $a$ , and the  $T_i$  now depend on  $\sum_{j=1}^{i-1} T_j$ ,  $i = 1, 2, \dots$ . Using some real life software failure data, Goel and Okumoto obtain the maximum likelihood estimators of  $a$  and  $b$ , and show that the estimates of reliability using their model are conservative as compared to those given by the J-M model. This, they argue, is a desirable feature.

**1.1. Motivation and summary.** Our initial motivation for undertaking the work reported

here grew out of a desire to respond to the second of the two issues critical of the J-M model. In so doing, we were led to a careful examination of the likelihood function for  $N$  and interpreting its behavior. This in turn led us to conclude that a Bayesian point of view is the natural one to consider here (see Meinhold and Singpurwalla (1983)). In the sequel, we determine that besides coherent inference (Lindley (1972, p. 3)), we also have a way to pull together the software reliability models presented above.

These and other matters are developed according to the following outline:

In §2 we present an alternate derivation of the J-M model using notions from the theory of shock models. We also obtain the posterior distributions for the parameters of this model for a certain choice of prior distributions. In §3 we show that the prior distributions of §2, regarded as special cases, result in the L-V model and the model by Goel and Okumoto. In §4 we derive certain properties, such as “stochastic inequalities” and “association,” of the times between software failures, and discuss the relevance of the above in software reliability.

The unification of models discussed in §3 enables us to view the major software reliability models, some of which have proved to be useful, in a comprehensive manner, and this together with the analysis of §4 is the main contribution of this paper.

**2. A shock model for software failures and inference for its parameters.** Before proceeding to the problem of inference for the parameters of the J-M model (1.1), we shall first consider the following alternative interpretation of the model.

To do this suppose that the inputs arrive at the software system according to the postulates of a Poisson process with intensity function  $\omega$ . Then, given  $N^*$  and  $N$ , the probability that the software encounters no failures in a time interval  $[0, t)$  is given by

$$(2.1) \quad \bar{F}(t|N, N^*) = \sum_{j=0}^{\infty} \left( \frac{e^{-\omega t} (\omega t)^j}{j!} \right) \left( \frac{N^* - N}{N^*} \right)^j.$$

As a justification for the above, note that the first term inside the summation sign denotes the probability that  $j$  inputs (shocks) are received in time  $t$ , and the second term inside the summation sign denotes the probability that all  $j$  inputs do not lead to a failure of the software. Arguments of this type form the basis of the theory of shock models and wear processes, which have played an important role in reliability theory (see Barlow and Proschan (1975, p. 52).

It is easy to verify that (2.1), when simplified, leads to

$$(2.2) \quad \bar{F}(t|N, N^*) = e^{-\omega N t / N^*},$$

implying that the time to first failure of the software, say  $T_1$ , has an exponential distribution with a scale parameter  $\omega N / N^*$ . Following the error correction policy described under (a) and (b) of §1, we note that  $T_i$ , the time between the  $(i-1)$ th and the  $i$ th failure of the software,  $i \leq N$ , has survival function

$$(2.3) \quad \bar{F}_i(t|N, N^*) = P\{T_i \geq t|N, N^*\} = \exp\left(-\frac{\omega(N-i+1)t}{N^*}\right), \quad t \geq 0.$$

In order to verify that the J-M model is a special case of the model described above, we note that for the J-M model,  $T_i$  has survival function

$$(2.4) \quad \bar{F}_i(t|N, \Lambda) = \exp(-\Lambda(N-i+1)t), \quad t \geq 0,$$

where  $N$  denotes the number of errors in the program, and  $\Lambda$  is an unknown constant. Clearly, (2.3) and (2.4) are alike if we set  $\Lambda = \omega / N^*$ .

The scenario described above can be viewed as being an alternative to the “bug-counting” scenario considered by Jelinski and Moranda. Mathematically, and especially from the point

of view of inference, the J-M model and the model derived above are the same, even though their motivations are different and their parameters have different interpretations. Since the mathematical form of the J-M model is familiar to those working with the modelling of software failures, we shall in the subsequent text use the form (2.4), and treat  $\Lambda$  and  $N$  as the unknown parameters.

Bayesian inference for the unknown parameters  $N$  and  $\Lambda$  involves assigning a prior distribution to the pair  $(N, \Lambda)$ , and then obtaining the resultant posterior distribution using  $T_1, \dots, T_k$  and the Bayes theorem. Our uncertainty in knowledge about  $N$  and  $\Lambda$  is summarized by the posterior distribution.

In §2.1, we obtain the posterior distribution of the pair  $(N, \Lambda)$  for various choices of their prior distribution. The parameters  $N$  and  $\Lambda$  refer to *tangible* quantities, for which a software expert must have judgments and perhaps even data, and these should be incorporated into our choice of the prior distribution. Perhaps a detailed discussion of how to elicit the software experts' judgments about  $N$  and  $\Lambda$  would be helpful here. However, we have refrained from including this in the interest of brevity, and because this aspect does not represent the main thrust of our paper.

To start our Bayesian analysis, we shall need the following assumption.

A1. Let the prior distribution of  $(N, \Lambda)$  be  $\mathcal{F}$ . In what follows, we shall consider special cases of  $\mathcal{F}$ . In what follows, we shall consider special cases of  $\mathcal{F}$  and obtain the resulting posterior distributions. These can be further exploited by a user.

**2.1. Determination of posterior distributions.** We shall consider three cases, Case 1 requiring the assumption that  $N^*$ , the total number of distinct input types, is conceptually infinite.

*Case 1.* Assume that  $\pi$ , the prior distribution for  $N$ , is Poisson with parameter  $\theta$ , and that  $\Lambda$  is degenerate at a known  $\lambda$ .

Let  $t_1, \dots, t_k$  be the observed times between failures, and let  $\underline{t}^{(k)} = (t_1, \dots, t_k)$ . Then given the observed outcome  $\underline{t}^{(k)}$ , the likelihood for  $N = q$ ,  $q \geq k$  is (for any unspecified but *non-informative* stopping rule)

$$L(N = q | \underline{t}^{(k)}) \propto \frac{q!}{(q - k)!} \lambda^k \exp \left[ -\lambda \sum_{j=1}^k (q - j + 1)t_j \right],$$

so that

$$L(N = q | \underline{t}^{(k)}) \pi(N = q) \propto \frac{e^{-\theta} \theta^q \lambda^k}{(q - k)!} \exp \left[ -\lambda \sum_{j=1}^k (q - j + 1)t_j \right].$$

Summing  $q$  from  $k$  to  $\infty$ , we obtain the posterior probability of  $N$  as

$$P\{N = q | t_1, \dots, t_k\} = \frac{\theta^{q-k}}{(q - k)!} \left( \exp \left[ -\lambda \sum_{j=1}^k t_j \right] \right)^{q-k} \theta \exp \left[ -\lambda \sum_{j=1}^k t_j \right].$$

If we let  $\gamma(t_1, \dots, t_k) = \theta \exp[-\lambda \sum_{j=1}^k t_j]$ , then it follows from the above that the posterior distribution of  $(N - k)$  is a Poisson distribution with mean  $\gamma(t_1, \dots, t_k)$ .

*Case 2.* Assume that  $N$  is degenerate at a known value, say  $n$ , and that  $\Lambda$  has a gamma distribution with scale parameter  $\mu$  and shape parameter  $\alpha$ . We denote this latter distribution as  $\Lambda \sim \mathcal{G}(\mu, \alpha)$ .

Given the observed outcome  $\underline{t}^{(k)}$ , the likelihood of  $\lambda$  for  $k \leq n$  is (for any unspecified noninformative stopping rule)

$$L(\lambda | \underline{t}^{(k)}) \propto \frac{n!}{(n - k)!} \lambda^k \exp \left[ -\lambda \sum_{j=1}^k (n - j + 1)t_j \right],$$

so that

$$L(\lambda | \mathbf{t}^{(k)}) d\mathcal{G}(\mu, \alpha) \propto \frac{n!}{(n-k)!} \frac{\mu^\alpha}{\Gamma(\alpha)} \lambda^{k+\alpha+1} \exp \left[ -\lambda \left( \mu + \sum_{j=1}^k (n-j+1)t_j \right) \right].$$

Integrating over  $\lambda$ , we obtain the joint density of the observed outcome as

$$f(\underline{t}^{(k)}) \propto \frac{n!}{(n-k)!} \frac{\mu^\alpha}{\Gamma(\alpha)} \Gamma(k+\alpha) \left[ \mu + \sum_{j=1}^k (n-j+1)t_j \right]^{-(k+\alpha)},$$

a multivariate Pareto, and Bayes theorem gives us the result that the posterior density of  $\Lambda$  is  $\mathcal{G}(\mu + \sum_{j=1}^k (n-j+1)t_j, k+\alpha)$ .

Case 3. Assume that  $N$  has any specified prior distribution, and that  $\Lambda \approx \mathcal{G}(\mu, \alpha)$ , independent of the distribution of  $N$ .

Given  $\underline{t}^{(k)}$ , and following the arguments used in Cases 1 and 2, we see that for  $q \geq k$ , and any unspecified noninformative stopping rule, the joint density of of  $\underline{t}^{(k)}$ ,  $N = q$ , and  $\lambda$  is

$$f(\underline{t}^{(k)}, N = q, \lambda) \propto \frac{\mu^\alpha}{\Gamma(\alpha)} \frac{q!}{(q-k)!} \lambda^{\alpha+k-1} \exp \left[ -\lambda \left( \mu + \sum_{j=1}^k (q-j+1)t_j \right) \right] P\{N = q\}.$$

Summing  $q$  from  $k$  to  $\infty$ , and integrating over  $\lambda$ , we obtain the joint density of the observed outcome as

$$f(\underline{t}^{(k)}) \propto \frac{\mu^\alpha \Gamma(\alpha+k)}{\Gamma(\alpha)} \left\{ \sum_{q=k}^{\infty} \frac{q!}{(q-k)!} P(N = q) \left( \mu + \sum_{j=1}^k (q-j+1)t_j \right)^{-(\alpha+k)} \right\},$$

which is a mixture of multivariate Paretos.

Using Bayes' theorem, the joint posterior probability of  $N$  and  $\Lambda$  is

$$\begin{aligned} P(N = q, \Lambda = \lambda | t_1, \dots, t_k) &= \frac{\lambda^{\alpha+k-1} e^{-\lambda(\mu + \sum_{j=1}^k (q-j+1)t_j)} \Gamma(\alpha+k)}{\Gamma(\alpha+k)} \left( \mu + \sum_{j=1}^k (q-j+1)t_j \right)^{-(\alpha+k)} \\ &\times \frac{q!}{(q-k)!} \frac{P(N = q) \left( \mu + \sum_{j=1}^k (q-j+1)t_j \right)^{-(\alpha+k)}}{\sum_{q=k}^{\infty} \frac{q!}{(q-k)!} \left( \mu + \sum_{j=1}^k (q-j+1)t_j \right)^{-(\alpha+k)} P(N = q)}. \end{aligned}$$

It now follows that the posterior probability of  $\Lambda$ , given that  $N = q$ , is  $\mathcal{G}(\mu + \sum_{j=1}^k (q-j+1)t_j, k+\alpha)$ , and the posterior probability of  $N = q$ , for  $q \geq k$ , is

$$P\{N = q | t_1, \dots, t_k\} = \frac{\frac{q!}{(q-k)!} \left( \mu + \sum_{j=1}^k (q-j+1)t_j \right)^{-(\alpha+k)} P(N = q)}{\sum_{r=k}^{\infty} \frac{r!}{(r-k)!} \left( \mu + \sum_{j=1}^k (r-j+1)t_j \right)^{-(\alpha+k)} P(N = r)}.$$

**3. Model unification.** Let  $M(t)$  denote the number of times that the software fails in a time interval  $[0, t)$ . Using the assumptions A2, A3, and A4 given below, we shall show in Theorem 3.3 that  $\{M(t), t \geq 0\}$  is a nonhomogeneous Poisson process with  $EM(t) = \theta(1 - e^{-\lambda t})$ , the constants  $\theta$  and  $\lambda$  are defined in A3 and A4, respectively. This is precisely the model considered by Goel and Okumoto (1979).



A2. For  $k = 1, 2, \dots$ , the conditional random variable  $(T_k|N, \Lambda)$  is such that

$$(T_k|N, \Lambda) = \begin{cases} \infty, & k > N, \\ (\Lambda(N - k + 1))^{-1}U_k, & k \leq N, \end{cases}$$

where  $U_1, U_2, \dots$  is an i.i.d. sequence of exponential random variables with mean 1.

A3. The prior distribution for  $N$  is Poisson with a known parameter  $\theta$ .

A4. The parameter  $\Lambda$  is degenerate at a known  $\lambda$ ; that is,  $P\{\Lambda = \lambda\} = 1$ .

Note that these assumptions describe Case 1 of §2.1. Thus, the model by Goel and Okumoto is a generalization of the J-M model obtained by assuming  $\Lambda$  known, and assigning a special prior to  $N$ .

Just as Case 1 helps us describe the Goel and Okumoto model, Case 2 helps us to describe the L-V model. Thus the L-V model is a generalization of the J-M model by assuming  $N$  known and assigning a specific prior to  $\Lambda$ . The model in Case 3 seems to be a combination of the Goel and Okumoto and L-V models, by assuming both  $N$  and  $\Lambda$  to be random variables.

To prove Theorem 3.5, we need assumption A5, which is a generalization of assumption A2, obtained by considering a nondecreasing positive real valued function  $h$  defined on  $(0, \infty)$ , and a sequence of i.i.d. random variables  $V_1, V_2, \dots$ , with mean 1.

A5. For  $k = 1, 2, \dots$ , the conditional random variable  $(T_k|N, \Lambda)$  is such that

$$(T_k|N, \Lambda) = \begin{cases} \infty, & k > N, \\ h(\Lambda(N - k + 1))V_k, & k \leq N. \end{cases}$$

Note that A5 reduces to A2 if  $V_1$  is an exponential random variable and  $h(x) = x^{-1}$ .

To begin our unifying effort, we need the following well-known lemma.

LEMMA 3.1. Let  $\tau_{n:1} < \tau_{n:2} < \dots < \tau_{n:n}$  be the order statistics of a sample of size  $n$  taken from an exponential distribution with mean 1. Furthermore, let  $U_1, U_2, \dots$  be independent and exponentially distributed random variables with mean 1. Then, for  $k = 1, \dots, n$  and  $t \geq 0$ ,

$$P\left\{\sum_{q=1}^k (n - q + 1)^{-1}U_q \leq t\right\} = P\{\tau_{n:k} \leq t\}.$$

*Proof.* Let  $\tau_{n:0} \equiv 0$ . The proof follows from the fact that the two random vectors  $[(n - q + 1)^{-1}U_q, q = 1, \dots, k]$  and  $[\tau_{n:q} - \tau_{n:q-1}, q = 1, \dots, k]$  are stochastically equal (Barlow and Proschan (1975, p. 59)).  $\square$

In an unpublished technical paper, Stefanski (1981) has further exploited the above lemma to point out some connections between the J-M model and the order statistics property of point processes.

In the theorem below we show that under A2, the number of times that the software fails between any two specified time points has a distribution that is a product of certain binomial terms.

THEOREM 3.2. Let  $0 \equiv t_0 < t_1 < \dots < t_k$ ;  $P_r = 1 - \exp(-\Lambda(t_r - t_{r-1}))$ ,  $l_r \in \{0, 1, \dots\}$ , and  $S_r = \sum_{j=1}^r l_j$ ,  $r = 1, \dots, k$  and  $S_0 = 0$ . Then, under A2, for  $S_k \leq N$ ,

$$P\{M(t_r) - M(t_{r-1}) = l_r, r = 1, \dots, k\} = \prod_{j=1}^k \binom{N - S_{j-1}}{l_j} P_j^{l_j} (1 - P_j)^{N - S_j}.$$

*Proof.* Throughout the proof, let  $M(t, N)$  denote the number of times that the software fails in time  $(0, t)$ , given  $N$ . Then, from the lack of memory property of the exponential random variables, we have for  $r = 1, \dots, k$

$$P\{(M(t_r, N) - M(t_{r-1}, N)) = l_r | M(t_j, N) = S_j, j = 0, \dots, r-1\} \\ = P\{M(t_r - t_{r-1}, N - S_{r-1}) = l_r\}.$$

Thus, to obtain the desired result, it suffices to show that for  $k=0, \dots, N$ , and  $t \geq 0$

$$P\{M(t, N) = k\} = \binom{N}{k} (1 - e^{-\Lambda t})^k e^{-\Lambda t(N-k)}.$$

We next verify that the above equation is true. Let  $S_q = \sum_{j=1}^q T_j$ , and let  $f_{S_j}$  denote the density function of  $S_q$ ,  $q = 1, \dots, N$ . Note that for  $t > 0$ , and  $k = 1, \dots, N-1$ ,

$$P\{S_k \leq t < S_{k+1}\} = [\Lambda(N - k)]^{-1} f_{S_{k+1}}(t)$$

and

$$P\{M(t, N) = k\} = \begin{cases} P\{S_1 > t\}, & k = 0, \\ P\{S_k \leq t < S_{k+1}\}, & 0 < k < N, \\ P\{S_N \leq t\}, & k = N. \end{cases}$$

Since  $S_k$  and  $\sum_{q=1}^k [\Lambda(N - q + 1)]^{-1} U_q$  are stochastically equal for  $k = 1, \dots, N$ , the desired result follows from Lemma 3.1.  $\square$

**THEOREM 3.3.** *Under A1, A2, A3, A4, the notation of Theorem 3.2, and*

$$\gamma_j \stackrel{\text{def}}{=} \theta P_j \prod_{q=1}^{j-1} (1 - P_q), \quad j = 1, \dots, k,$$

then

$$P\{(M(t_r) - M(t_{r-1})) = l_r, r = 1, \dots, k\} = \prod_{r=1}^k e^{-\gamma_r} \gamma_r^{l_r} (l_r!)^{-1}.$$

*Proof.* By Theorem 3.2,

$$P\{(M(t_r) - M(t_{r-1})) = l_r, r = 1, \dots, k\} \\ = \sum_{n=S_k}^{\infty} P\{(M(t_r) - M(t_{r-1})) = l_r, r = 1, \dots, k | N = n\} P\{N = n\} \\ = e^{-\theta} \prod_{r=1}^k \frac{(\gamma_r)^{l_r}}{l_r!} \sum_{n=S_k}^{\infty} \left[ \theta \prod_{j=1}^k (1 - P_j) \right]^{n-S_k} [(n - S_k)!]^{-1} \\ = \prod_{r=1}^k e^{-\gamma_r} \frac{(\gamma_r)^{l_r}}{l_r!}$$

since

$$\sum_{r=1}^k P_r \prod_{q=1}^{r-1} (1 - P_q) = 1 - \prod_{q=1}^k (1 - P_q).$$

Clearly, under A2–A4,

$$P\{(M(t_r) - M(t_{r-1})) = l_r, r = 1, \dots, k\} = \int_0^{\infty} \prod_{r=1}^k e^{-\gamma_r} \frac{(\gamma_r)^{l_r}}{l_r!} dP\{\Lambda \leq \lambda\}.$$

In particular, for  $t \geq 0$ ,  $l \in \{0, 1, \dots\}$ ,

$$P\{M(t) = l\} = \int_0^\infty \exp[-\theta(1 - e^{-\lambda t})] \frac{[\theta(1 - e^{-\lambda t})]^l}{l!} dP\{\Lambda \leq \lambda\},$$

and

$$EM(t) = \theta(1 - Ee^{-\Lambda t}) = \theta(1 - e^{-\lambda t}).$$

A key feature of L-V is that the times between failures of the software,  $T_1, T_2, \dots$ , have a decreasing failure rate (DFR). Even though the above authors motivate this DFR feature with an interesting subjective argument, they deduce special cases of this property by using physical arguments. We show (Theorem 3.5) that the DFR feature follows, in a more general way, within the framework of our assumptions. A second feature of the L-V paper is that  $T_1, T_2, \dots$  are stochastically increasing. In Theorem 3.4 below, we obtain a stochastic relationship between the two unconditional random variables  $T_k$  and  $T_{k+r}$ , for all positive integers  $k$  and  $r$ , which retains this second feature.

**THEOREM 3.4.** *Under A1, A3, and A5  $T_k$  is stochastically less than  $T_{k+r}$ , for all positive integers  $k$  and  $r$ .*

*Proof.* Let  $t \geq 0$ , and  $l$  be a positive integer. Then

$$P\{T_l > t\} = P\{N < l\} + \sum_{q=l}^\infty \frac{e^{-\theta} \theta^q}{q!} \int_0^\infty P\{V_1 > t[h(\lambda(q - l + 1))]^{-1}\} dP[\Lambda \leq \lambda | \theta].$$

Thus,  $P\{T_l > t\} \leq P\{T_{l+1} > t\}$ ,  $t > 0$ , and the desired result follows.  $\square$

Note that in the above theorem we can choose A3, the assumption of a Poisson prior distribution, without any loss of generality.

**THEOREM 3.5.** *Under A1, A3, A5, and the assumption that  $V_1$  has a DFR,  $T_k$ ,  $k = 1, 2, \dots$ , also has a DFR.*

*Proof.* Let  $t \geq 0$ ; then

$$P\{T_k > t\} = P\{N < k\} + \int_0^\infty \int_0^\infty P\{h(\lambda(n - k + 1))V_1 > t\} dF(n, \lambda).$$

Since a mixture of DFR distributions in DFR (Barlow and Proschan (1975, p. 103), the desired result follows.  $\square$

**4. Association of the times between software failures and its implications.** In §2 we have assumed that  $T_1, \dots, T_n$ , the times between software failures, are conditionally independent. That such an assumption need not always be true has been pointed out by Crow and Singpurwalla (1983). As a first step towards weakening this assumption, we shall now prove the association of  $[T_1, \dots, T_k]$ ,  $k = 1, \dots, 2$ . Association as a notion of positive dependence is well known in reliability theory, and is summarized by Barlow and Proschan (1975, p. 29). We first prove the following lemma, which is of interest in its own right.

**LEMMA 4.1.** *Let  $\mathbf{T}$  and  $\mathbf{S}$  be some random vectors defined on some probability space. Assume that*

- (i) *the conditional random vector  $\mathbf{T}|\mathbf{S}$  is associated;*
- (ii) *the conditional random vector  $\mathbf{T}|\mathbf{S}$  is stochastically nondecreasing (nonincreasing) in  $\mathbf{S}$ ; and*
- (iii) *the random vector  $\mathbf{S}$  is associated.*

*Then the random vector  $\mathbf{T}$  is associated.*

*Proof.* Assume that  $\mathbf{T}$  is a  $k$ -dimensional random vector, and let  $f_1$  and  $f_2$  be two bounded real valued nondecreasing functions defined on the  $k$ th Euclidean space. First, note that

$$\text{Cov}\{f_1(\mathbf{T}), f_2(\mathbf{T})\} = E \text{Cov}\{f_1(\mathbf{T}), f_2(\mathbf{T}) | \mathbf{S}\} + \text{Cov}\{E\{f_1(\mathbf{T}) | \mathbf{S}\}, E\{f_2(\mathbf{T}) | \mathbf{S}\}\}.$$

By (i),  $\text{Cov}\{f_1(\mathbf{T}), f_2(\mathbf{T}) | \mathbf{S}\} \geq 0$ . By (ii),  $E\{f_1(\mathbf{T}) | \mathbf{S}\}$  and  $E\{f_2(\mathbf{T}) | \mathbf{S}\}$  are nondecreasing (nonincreasing) in  $\mathbf{S}$ . Thus by (iii)  $\text{Cov}\{E\{f_1(\mathbf{T}) | \mathbf{S}\}, E\{f_2(\mathbf{T}) | \mathbf{S}\}\} \geq 0$ . The result of the lemma follows.  $\square$

**THEOREM 4.2.** *Assume that A1, A3, and A5 hold, and that the pair  $(N, \Lambda)$  is associated, Then for every positive integer  $k$ ,  $\mathbf{T}_k = [T_1, \dots, T_k]$  is associated.*

*Proof.* Under A5,  $(\mathbf{T}_k | N, \Lambda)$  has independent components, and is therefore associated. Since  $(N, \Lambda)$  is associated, the result of the theorem can be proved using Lemma 4.1, if we can show that  $(\mathbf{T}_k | N, \Lambda)$  is stochastically nonincreasing in  $(N, \Lambda)$ .

Since  $(\mathbf{T}_k | N, \Lambda)$  has independent components, the preceding statement can be proved if we can show that for each positive integer  $k$ ,  $(T_k | N, \Lambda)$  is nonincreasing in  $(N, \Lambda)$ .

To prove the latter statement, we note that for  $k = 1, 2, \dots$  and  $t \geq 0$ ,

$$P\{T_k > t | N, \Lambda\} = \begin{cases} 1, & k > N, \\ P\{V_1 > th(\Lambda(N - k + 1))\}, & k \leq N. \end{cases}$$

Thus,  $(T_k | N, \Lambda)$  is stochastically nonincreasing in  $(N, \Lambda)$ .  $\square$

Note that if  $N$  and  $\Lambda$  are independent random variables, and if A1, A3, and A5 hold, then  $T_1, \dots, T_k$  are associated for  $k = 1, 2, \dots$ .

By way of a corollary to Theorem 4.2, we have the following inequalities which are useful and have an interesting interpretation.

**COROLLARY 4.3.** *Assume that the conditions of Theorem 4.2 hold. Let  $T_0 \stackrel{\text{def}}{=} \infty$ ,  $t_0 \stackrel{\text{def}}{=} 0$ , and let  $r$  and  $k$  be positive integers such that  $1 \leq r < k$ . Then for all  $t_1, \dots, t_k$ ,  $t_k \geq 0$ ,*

$$P\{T_r > t_r, \dots, T_k > t_k | T_0 > t_0, \dots, T_{r-1} > t_{r-1}\} \geq P\{T_r > t_r, \dots, T_k > t_k\},$$

and

$$P\{T_r \leq t_r, \dots, T_k \leq t_k | T_0 \leq t_0, \dots, T_{r-1} \leq t_{r-1}\} \geq P\{T_r \leq t_r, \dots, T_k \leq t_k\}.$$

*Proof.* Let  $f_1(\mathbf{x}) = \prod_{q=r}^k I(x_q = t_q)$ ,  $f_2(\mathbf{x}) = \prod_{q=1}^r I(x_q > t_q)$ ,  $f_3(\mathbf{x}) = \prod_{q=r}^k I(x_q \leq t_q)$ ,  $f_4(\mathbf{x}) = \prod_{q=1}^r I(x_q \leq t_q)$ , where  $\mathbf{x} = [x_1, \dots, x_k]$ , and  $I$  is an indicator function. The results of the corollary now follow, since by Theorem 4.2,

$$\text{Cov}\{f_1(\mathbf{T}_k), f_2(\mathbf{T}_k)\} \geq 0 \quad \text{and} \quad \text{Cov}\{f_3(\mathbf{T}_k), f_4(\mathbf{T}_k)\} \geq 0. \quad \square$$

One way to interpret the inequalities of Corollary 4.3 is that statements about the uncertainties of the future times between failures,  $T_r, \dots, T_k$ , can be improved by incorporating knowledge about the previous observed times between failures,  $T_0, \dots, T_{r-1}$ . This is reasonable because the association of  $\mathbf{T}_k = [T_1, \dots, T_k]$  for every positive integer  $k$  implies a kind of positive dependence between the components of  $\mathbf{T}_k$ . It is useful to remark that a similar interpretation would have been possible had we been able to establish a negative dependence between the components of  $\mathbf{T}_k$ . What is important here is that positive or negative correlation conveys ‘‘information’’ among the variables. We suspect that inequalities of the kind described by Corollary 4.3 may be useful for preposterior analysis.

**Acknowledgments.** We would like to acknowledge comments on a preliminary version of this paper by Chris Dale, Ralph Evans, Thomas Mazzuchi, Ray Shafer, and Len Stefansky. Professor Richard E. Barlow has made substantial comments which have led us to rethink matters and arrive at the present version.

## REFERENCES

- [1] R. F. BARLOW AND F. PROSCHAN (1975), *Statistical Theory of Reliability and Life Testing*, Holt, Rinehart and Winston, New York.
- [2] L. H. CROW AND N. D. SINGPURWALLA (1984), *An empirically developed Fourier series model for describing software failures*, IEEE Trans. Reliability, R-33, pp. 176–183.
- [3] E. H. FORMAN AND N. D. SINGPURWALLA (1977), *An empirical stopping rule for debugging and testing computer software*, J. Amer. Statist. Assoc., 72, pp. 750–757.
- [4] ——— (1979), *Optimal time intervals for testing hypotheses on computer software errors*, IEEE Trans. Reliability, R-28, pp. 250–253.
- [5] A. L. GOEL AND K. OKUMOTO (1979), *Time-dependent error-detection rate model for software reliability and other performance measures*, IEEE Trans. Reliability, R-28, pp. 206–211.
- [6] Z. JELINSKI AND P. B. MORANDA (1972), *Software reliability research*, in Statistical Computer Performance Evaluation, W. Freiberger, ed., Academic Press, New York, pp. 485–502.
- [7] D. V. LINDLEY (1972), *Bayesian Statistics, A Review*, CBMS Regional Conference Series in Applied Mathematics 2, Society for Industrial and Applied Mathematics, Philadelphia.
- [8] B. LITTLEWOOD (1981), *A critique of the Jelinski-Moranda model for software reliability*, Proc. Ann. Rel. and Maintainability Symp., pp. 357–364.
- [9] B. LITTLEWOOD AND J. L. VERRALL (1973), *A Bayesian reliability growth model for computer software*, Record IEEE Symposium on Computer Software Reliability, IEEE, New York, pp. 70–77.
- [10] R. J. MEINHOLD AND N. D. SINGPURWALLA (1983), *Bayesian analysis of a commonly used model for describing software failures*, The Statistician, 32(1&2), pp. 168–173.
- [11] L. A. STEFANSKY (1981), *A note on a well-known software reliability model*, unpublished report, Division of Mathematics, U.S. Army Research Office, Research Triangle Park, NC.

**CORRIGENDUM:  
BLOCK PRECONDITIONING FOR THE CONJUGATE  
GRADIENT METHOD\***

P. CONCUS†, G. H. GOLUB‡ AND G. MEURANTS§

In Theorem 4 and Theorem 5 change “*negative*” to “*nonpositive*”.

---

\* This Journal, 6 (1985), pp. 220–252.

† Lawrence Berkeley Laboratory and Department of Mathematics, University of California, Berkeley, California 94720.

‡ Computer Science Department, Stanford University, Stanford, California 94305.

§ Commissariat à l’Energie Atomique, Limeil 94190 Villeneuve-Saint-Georges, France, and Computer Science Department, Stanford University, Stanford, California 94305.

## CONTINUATION-CONJUGATE GRADIENT METHODS FOR THE LEAST SQUARES SOLUTION OF NONLINEAR BOUNDARY VALUE PROBLEMS\*

R. GLOWINSKI†, H. B. KELLER‡ AND L. REINHART§

**Abstract.** We discuss in this paper a new combination of methods for solving nonlinear boundary value problems containing a parameter. Methods of the continuation type are combined with least squares formulations, preconditioned conjugate gradient algorithms and finite element approximations.

We can compute branches of solutions with limit points, bifurcation points, etc.

Several numerical tests illustrate the possibilities of the methods discussed in the present paper; these include the Bratu problem in one and two dimensions, one-dimensional bifurcation and perturbed bifurcation problems, the driven cavity problem for the Navier–Stokes equations.

**Key words.** nonlinear boundary value problems, bifurcation, continuation methods, nonlinear least squares, conjugate gradient, finite elements, Navier–Stokes equations, biharmonic solvers

**1. Introduction.** We present in this paper a powerful combination of techniques that are used to solve a variety of nonlinear boundary value problems containing a parameter. Indeed the resulting method can be employed to study a large class of nonlinear eigenvalue problems. The individual techniques include: arclength or pseudo-arclength continuation, least squares formulation in an appropriate Hilbert space setting, a conjugate gradient iterative method for solving the least squares problem and finite element approximations to yield a finite dimensional problem for computation.

In § 2 the solution techniques are described in some detail. Specifically in § 2.1 the last squares formulation of a broad class of nonlinear problems, say in the form

$$(1.1) \quad AU = T(U),$$

are formulated in an appropriate Hilbert space setting. Then in § 2.2 a conjugate gradient iterative solution technique for solving such least squares problems is presented. In § 2.3 a pseudo-arc length continuation method for nonlinear eigenvalue problems in the form

$$(1.2) \quad Lu = G(u, \lambda)$$

is discussed. This involves adjoining a scalar linear constraint, say

$$(1.3) \quad l(u, \lambda, s) = 0,$$

and with  $U \equiv \{u, \lambda\}$  the previous least squares and conjugate gradient techniques can be applied to the system (1.2), (1.3). One big advantage of our specific continuation method is that simple limit or fold points of the original problem (1.2) are just regular points for our reformulation in the form (1.1). The entire procedure thus enables us to determine large arcs of branches of solutions of (1.2) with no special precautions or change of methods near limit points.

---

\* Received by the editors November 22, 1982, and in revised form June 12, 1984.

† Paris VI University, LA 189, Tour 55.65, 75230 Paris Cedex 05 France, and INRIA, Domaine de Voluceau, Rocquencourt, 78153 Le Chesnay Cedex, France.

‡ Applied Mathematics, California Institute of Technology, Pasadena, California 91125. The research of this author was supported in part by the U.S. Department of Energy under contract EY-76-S-03-0767, Project Agreement 12, and by the Army Research Office under contract DAAG 29-78-C-0011.

§ INRIA, Domaine de Voluceau, Rocquencourt, 78153 Le Chesnay Cedex, France.

These techniques, as described in § 2, apply to the analytical problem. However they go over extremely well when various discrete approximations are applied to yield computational methods of great power and practicality. We illustrate this by considering several nonlinear boundary value problems of some difficulty and current interest. In each of these problems the discretization is obtained by some finite element formulation. The well-known Bratu problem on a square domain is treated in § 3. Several ordinary differential equation examples displaying bifurcation and the effects of perturbed bifurcation are treated in § 4. We show how to use perturbed bifurcation and continuation to obtain the bifurcating solutions. Finally in § 5 the Navier–Stokes equations are solved for the driven cavity problem.

Actually the techniques described in this paper have also been applied to the solution of nonlinear boundary value problems, more complicated than those considered in the following sections. Among these problems, we shall mention the *Von Karman equations for nonlinear plates* and the computation of the multiple solutions of the *full potential equation* modelling *transonic flows for compressible inviscid fluids*.

**2. Solution techniques.** We introduce in this section the methods we shall apply in §§ 3, 4, 5, to the solution of quite general nonlinear boundary value problems. They include *least squares*, *conjugate gradient* and *arc length continuation* methods.

Let  $V$  be a *Hilbert space* (real for simplicity) equipped with the scalar product  $(\cdot, \cdot)$  and the corresponding norm  $\|\cdot\|$ . We denote by  $V'$  the *dual space* of  $V$ , by  $\langle \cdot, \cdot \rangle$  the *duality pairing* between  $V$  and  $V'$ , and by  $\|\cdot\|_*$  the corresponding *dual norm*, i.e.

$$(2.1) \quad \|f\|_* = \sup_{v \in V - \{0\}} \frac{|\langle f, v \rangle|}{\|v\|} \quad \forall f \in V'.$$

The problem that we consider is to find  $u \in V$  such that

$$(2.2) \quad S(u) = 0,$$

where  $S$  is a nonlinear operator from  $V$  to  $V'$ .

**2.1. Least squares formulation of problem (2.2).** A least squares formulation of (2.2) is obtained by observing that any solution of (2.2) is also a *global minimizer* over  $V$  of the functional  $J: V \rightarrow \mathbb{R}$  defined by

$$(2.3) \quad J(v) = \frac{1}{2} \|S(v)\|_*^2.$$

Hence a least squares formulation of (2.2) is:

*Find  $u \in V$  such that*

$$(2.4) \quad J(u) \leq J(v) \quad \forall v \in V.$$

In practice we proceed as follows. Let  $A$  be the *duality isomorphism* corresponding to  $(\cdot, \cdot)$  and  $\langle \cdot, \cdot \rangle$ . That is  $\forall v \in V, Av \in V'$  satisfies

$$(2.5) \quad \langle Av, w \rangle = (v, w) \quad \forall w \in V,$$

$$(2.6) \quad \|v\| = \|Av\|_* \quad (\text{or equivalently } \|f\|_* = \|A^{-1}f\|, \forall f \in V').$$

It follows that

$$(2.7) \quad J(v) = \frac{1}{2} \langle A\xi, \xi \rangle \quad (= \frac{1}{2} \|\xi\|^2),$$

where  $\xi$  is a (nonlinear) function of  $v$  obtained via the solution of the *well-posed linear problem*

$$(2.8) \quad A\xi = S(v).$$



We observe that (2.4) has the structure of an *optimal control* problem, where

- (i)  $v$  is the *control vector*,
- (ii)  $\xi$  is the *state vector*,
- (iii) (2.8) is the *state equation*,
- (iv)  $J$  is the *cost function*.

As a final remark we observe that any solution of the minimization problem (2.4) for which  $J$  vanishes is also a solution of the original problem (2.2).

**2.2. Solution by a conjugate gradient algorithm.** We suppose from now on that  $S$  is differentiable implying in turn the differentiability of  $J$  over  $V$ . We denote by  $S'$  and  $J'$  the Fréchet derivatives of  $S$  and  $J$  respectively.

From the differentiability of  $J$  it is quite natural to solve the minimization problem (2.4) by a conjugate gradient algorithm; among the possible conjugate gradient algorithms we have selected the Polak–Ribière variant (cf. Polak [1]) whose very good performance has been discussed by Powell [2] (see also Shanno [28]). The Polak–Ribière method applied to the solution of (2.4) provides the following algorithm.

*Step 0: Initialization.* For some given

$$(2.9) \quad u^0 \in V,$$

compute  $g^0 \in V$  as the solution of

$$(2.10) \quad Ag^0 = J'(u^0),$$

and set

$$(2.11) \quad z^0 = g^0.$$

Then, for  $n \geq 0$ , with  $u^n, g^n, z^n$  known, compute  $u^{n+1}, g^{n+1}, z^{n+1}$  by:

*Step 1: Descent.* Compute:

$$(2.12) \quad \rho_n = \arg \min_{\rho \in \mathbb{R}} J(u^n - \rho z^n),$$

and then set

$$(2.13) \quad u^{n+1} = u^n - \rho_n z^n.$$

*Step 2: New descent direction.* Define  $g^{n+1} \in V$  as the solution of

$$(2.14) \quad Ag^{n+1} = J'(u^{n+1});$$

then compute

$$(2.15) \quad \gamma_n = \frac{\langle A(g^{n+1} - g^n), g^{n+1} \rangle}{\langle Ag^n, g^n \rangle} \left( = \frac{\langle g^{n+1} - g^n, g^{n+1} \rangle}{\langle g^n, g^n \rangle} \right)$$

and set

$$(2.16) \quad z^{n+1} = g^{n+1} + \gamma_n z^n.$$

Set  $n = n + 1$  and return to Step 1.

The two costly steps (because they need some auxiliary computations) of algorithm (2.9)–(2.16) are:

- (i) The solution of the one-dimensional minimization problem (2.12) to obtain  $\rho_n$ . We have done the corresponding line search by *dichotomy* and *parabolic interpolation*, using  $\rho_{n-1}$  as starting value<sup>1</sup> (see [3] for more details). We recall that each evaluation

<sup>1</sup> If the nonlinearity is *polynomial* we can use faster methods.

of  $J(v)$ , for a given argument  $v$ , requires the solution of the linear problem (2.8) to obtain the corresponding  $\xi$ .

(ii) The calculation of  $g^{n+1}$  from  $u^{n+1}$  which requires the solution of two linear problems associated with  $A$  (namely (2.8) with  $v = u^{n+1}$  and (2.14)).

*Calculation of  $J'(u^n)$  and  $g^n$ :* Owing to the importance of Step 2, let us detail the calculation of  $J'(u^n)$  and  $g^n$ .

Let  $v \in V$ ; then  $J'(v)$  may be defined by

$$(2.17) \quad \langle J'(v), w \rangle = \lim_{\substack{t \rightarrow 0 \\ t \neq 0}} \frac{J(v + tw) - J(v)}{t} \quad \forall w \in V.$$

We obtain from (2.7), (2.8), (2.17) that

$$(2.18) \quad \langle J'(v), w \rangle = \langle A\xi, \eta \rangle$$

where  $\xi$  and  $\eta$  are the solutions of (2.8) and

$$(2.19) \quad A\eta = S'(v) \cdot w,$$

respectively. Since  $A$  is *self-adjoint* (from (2.5)) we also have from (2.18), (2.19) that

$$(2.20) \quad \langle J'(v), w \rangle = \langle A\xi, \eta \rangle = \langle A\eta, \xi \rangle = \langle S'(v) \cdot w, \xi \rangle.$$

Therefore  $J'(v) \in V'$  may be identified with the *linear functional*

$$(2.21) \quad w \rightarrow \langle S'(v) \cdot w, \xi \rangle.$$

It follows then from (2.14), (2.20), (2.21) that  $g^n$  is the solution of the following *linear variational problem*:

$$(2.22) \quad \text{Find } g^n \in V \text{ such that} \\ \langle Ag^n, w \rangle = \langle S'(u^n) \cdot w, \xi^n \rangle \quad \forall w \in V,$$

where  $\xi^n$  is the solution of (2.8) corresponding to  $v = u^n$ .

*Remark 2.1.* It is clear from the above observations that an efficient solver for linear problems related to operator  $A$  (in fact to a *finite-dimensional* approximation of  $A$ ) will be a fundamental tool for the solution of problem (2.2) by the conjugate gradient algorithm (2.9)–(2.16).

*Remark 2.2.* The fact that  $J'(v)$  is known through (2.20) is not at all a drawback if a *Galerkin* or a *finite element* method is used to approximate (2.2). Indeed we only need to know the value of  $\langle J'(v), w \rangle$  for  $w$  belonging to a *basis* of the *finite-dimensional subspace* of  $V$  corresponding to the Galerkin or finite element approximation under consideration.

*Convergence of algorithm (2.9)–(2.16):* We introduce the concept of *regular solution* of problem (2.2) by the following definition.

DEFINITION 2.1. A solution  $u$  of (2.2) is said to be *regular* if the operator  $S'(u)$  ( $\in \mathcal{L}(V, V')$ ) is an isomorphism from  $V$  onto  $V'$ .

Using a modification of the finite-dimensional techniques of Polak [1], it has been proved in Reinhart [3] that if problem (2.2) has a *finite number* of solutions and if these solutions are *regular*, then the conjugate gradient algorithm (2.9)–(2.16) converges to a solution of (2.2), depending upon the initial iterate  $u^0$  in (2.9). This convergence result requires that  $u^0$  is well chosen, as in Newton's method. Hence the role that continuation methods may play is apparent.

**2.3. Arc length continuation methods.** Consider now the solution of nonlinear problems depending upon a real parameter  $\lambda$ ; we would like to follow in the space  $V \times \mathbb{R}$  branches of solutions  $\{u(\lambda), \lambda\}$  when  $\lambda$  belongs to a compact interval of  $\mathbb{R}$ .

These nonlinear eigenvalue problems can be written as follows

$$(2.23) \quad S(u, \lambda) = 0, \quad \lambda \in \mathbb{R}, \quad u \in V.$$

Equation (2.23) reduces quite often to

$$(2.24) \quad Lu = G(u, \lambda), \quad \lambda \in \mathbb{R}, \quad u \in V,$$

where  $L: V \rightarrow V'$  is a linear elliptic operator, and where  $G$  is a *nonlinear Fredholm operator* (see e.g. Berger [4] for the definition of Fredholm operators).

A classical approach is to use  $\lambda$  as the parameter defining arcs of solutions. If for  $\lambda = \lambda_0$  problem (2.23) has a unique solution  $u = u_0$  and if that solution is *isolated*, that is

$$(2.25) \quad S_u^0 = \frac{\partial S}{\partial u}(u_0, \lambda_0) \text{ is an isomorphism from } V \text{ onto } V',$$

and if  $\{u, \lambda\} \rightarrow S(u, \lambda)$  is  $C^1$  in some ball around  $\{u_0, \lambda_0\}$ , then the *implicit function theorem* implies the existence of a smooth arc of regular solutions  $u = u(\lambda)$  for  $|\lambda - \lambda_0| < \rho$ . Therefore, for  $\lambda$  given *sufficiently close to*  $\lambda_0$  we may solve problem (2.23) just as problem (2.2). These procedures, however, may fail or encounter difficulties (slow convergence for example) close to a *nonisolated* solution.

To overcome these difficulties we replace problem (2.23) by the following system

$$(2.26) \quad S(u, \lambda) = 0,$$

$$(2.27) \quad I(u, \lambda, s) = 0,$$

where  $I: V \times \mathbb{R} \times \mathbb{R} \rightarrow \mathbb{R}$  is chosen such that  $s$  is some *arc length* (or a convenient *approximation* to it) on the solution branch. We look then for a solution  $\{u(s), \lambda(s)\}$ ,  $s$  being given (but not  $\lambda$ ). If in addition to  $\{u_0, \lambda_0\}$  we know a tangent vector to the path  $\{\dot{u}(s_0), \dot{\lambda}(s_0)\}$  (where  $\dot{v}$  denotes the derivative of  $v$  with respect to  $s$ ) satisfying:

$$(2.28a) \quad S_u(u_0, \lambda_0)\dot{u}(s_0) + S_\lambda(u_0, \lambda_0)\dot{\lambda}(s_0) = 0,$$

$$(2.28b) \quad \|\dot{u}(s_0)\|^2 + |\dot{\lambda}(s_0)|^2 = 1,$$

then we can use

$$(2.29) \quad I(u, \lambda, s) \equiv (\dot{u}(s_0), u(s) - u(s_0)) + \dot{\lambda}(s_0)(\lambda(s) - \lambda(s_0)) - (s - s_0) = 0,$$

for  $|s - s_0|$  sufficiently small.

Let us define  $U \in V \times \mathbb{R}$  by  $U = \{u, \lambda\}$ ; then problem (2.26), (2.27) can be written as

$$(2.30) \quad T_s(U) = 0,$$

where

$$(2.31) \quad T_s(U) = \begin{pmatrix} T_{1s}(U) \\ T_{2s}(U) \end{pmatrix}$$

with

$$(2.32) \quad T_{1s}(U) = S(u, \lambda), \quad T_{2s}(U) = I(u, \lambda, s).$$

The main interest of this new formulation is that the ordinary *limit points* of (2.26) become *regular* solutions of (2.30) (see H. B. Keller [5], [6] for more details).

Using the notation of § 2.1 a least squares formulation of (2.30) (generalizing (2.4)) is given by:

$$(2.33) \quad \text{Find } U(s) = \{u(s), \lambda(s)\} \text{ such that} \\ J_s(U(s)) \leq J_s(W) \quad \forall W = \{w, \mu\} \in V \times \mathbb{R},$$

with

$$(2.34) \quad J_s(W) = \frac{1}{2} \langle A\tilde{w}, \tilde{w} \rangle + \frac{1}{2} |\tilde{\mu}|^2$$

where, in (2.34),  $\tilde{w}$  and  $\tilde{\mu}$  are (nonlinear) functions of  $\{w, \mu\}$  via the solution of the linear problems

$$(2.35) \quad A\tilde{w} = T_{1s}(w, \mu),$$

$$(2.36) \quad \tilde{\mu} = T_{2s}(w, \mu),$$

respectively.

We consider now a conjugate gradient algorithm (in fact, a variation of algorithm (2.9)–(2.16)) to solve the least squares problem (2.33); this algorithm is defined as follows.

*Step 0: Initialization.* For some given

$$(2.37) \quad U^0 = \{u^0, \lambda^0\}$$

compute  $G^0 = \{g_u^0, g_\lambda^0\} \in V \times \mathbb{R}$  as the solution of

$$(2.38) \quad Ag_u^0 = \frac{\partial J_s}{\partial u}(U^0),$$

$$(2.39) \quad g_\lambda^0 = \frac{\partial J_s}{\partial \lambda}(U^0),$$

and set

$$(2.40) \quad Z^0 = G^0.$$

Then for  $n \geq 0$ , with  $U^n, G^n, Z^n$  known, compute  $U^{n+1}, G^{n+1}, Z^{n+1}$  as follows.

*Step 1: Descent.* Compute

$$(2.41) \quad \rho_n = \text{Arg Min}_{\rho \in \mathbb{R}} J_s(U^n - \rho Z^n),$$

$$(2.42) \quad U^{n+1} = U^n - \rho_n Z^n$$

(i.e.  $u^{n+1} = u^n - \rho_n z_u^n, \lambda^{n+1} = \lambda^n - \rho_n z_\lambda^n$ ).

*Step 2: Calculation of the new descent direction.* Define  $G^{n+1} = \{g_u^{n+1}, g_\lambda^{n+1}\} \in V \times \mathbb{R}$  as the solution of

$$(2.43) \quad Ag_u^{n+1} = \frac{\partial J_s}{\partial u}(U^{n+1}),$$

$$(2.44) \quad g_\lambda^{n+1} = \frac{\partial J_s}{\partial \lambda}(U^{n+1}),$$

then compute

$$\begin{aligned}
 \gamma_n &= \frac{\langle A(g_u^{n+1} - g_u^n), g_u^{n+1} \rangle + (g_\lambda^{n+1} - g_\lambda^n) g_\lambda^{n+1}}{\langle A g_u^n, g_u^n \rangle + |g_\lambda^n|^2} \\
 (2.45) \qquad &= \frac{(g_u^{n+1} - g_u^n, g_u^{n+1}) + (g_\lambda^{n+1} - g_\lambda^n) g_\lambda^{n+1}}{\|g_u^n\|^2 + |g_\lambda^n|^2},
 \end{aligned}$$

and set

$$(2.46) \qquad Z^{n+1} = G^{n+1} + \gamma_n Z^n.$$

Set  $n \leftarrow n + 1$  and return to Step 1.

The various comments given in § 2.2, concerning algorithm (2.9)–(2.16), still hold for algorithm (2.37)–(2.46). In particular as we pointed out in Remark 2.1 is the importance of efficient methods for solving linear problems related to operator  $A$ . This remark still holds, indeed, since in the context of § 2.3,  $A$  is replaced by the *block-diagonal* isomorphism  $\mathcal{A}: V \times \mathbb{R} \rightarrow V' \times \mathbb{R}$  defined by

$$\mathcal{A} = \begin{pmatrix} A & 0 \\ 0 & 1 \end{pmatrix}.$$

We do not go into the details of the calculation of  $\partial J_s / \partial u$ ,  $\partial J_s / \partial \lambda$  since it is just a trivial modification of the calculation done in § 2.2 to obtain  $J'$ .

A crucial step in the continuation method is a “good” *initialization* choice in (2.37). The obvious choice

$$(2.47) \qquad \{u^0, \lambda^0\} = \{u(s_0), \lambda(s_0)\}$$

is naive and a better choice is provided by using the tangent of (2.28) in the *extrapolation*:

$$\begin{aligned}
 (2.48) \qquad u^0 &= u(s_0) + (s - s_0) \dot{u}(s_0), \\
 \lambda^0 &= \lambda(s_0) + (s - s_0) \dot{\lambda}(s_0).
 \end{aligned}$$

This results in much faster convergence, especially close to the limit points. This initialization technique leads to the so-called *continuation method with incremental load* and is of order 2 (see Deuffhard [38]). We shall return to this initialization problem in § 2.4.

*Convergence of algorithm (2.37)–(2.46).* The fundamental advantage of the arc length continuation approach is that it provides an efficient solution method in the neighborhood of the so-called *limit* (or *fold*) *points* of problem (2.23). A precise formulation of the concept of simple limit points is given by the following definition.

DEFINITION 2.2. Let  $\{u_0, \lambda_0\} \in V \times \mathbb{R}$  be the solution of problem (2.23). We say that  $\{u_0, \lambda_0\}$  is a *simple limit point* if:

$$(2.49) \qquad \dim \text{Ker} \left( \frac{\partial S}{\partial u}(u_0, \lambda_0) \right) = \text{codim Range} \left( \frac{\partial S}{\partial u}(u_0, \lambda_0) \right) = 1,$$

$$(2.50) \qquad \frac{\partial S}{\partial \lambda}(u_0, \lambda_0) \notin \text{Range} \left( \frac{\partial S}{\partial u}(u_0, \lambda_0) \right).$$

We show in Fig. 3.2 an example of such a limit point (located on a solution curve of  $S(u, \lambda) = 0$  where  $\lambda_0 = \lambda_{CR}$ ).

The main justification of arc length continuation methods follows from the next proposition.

PROPOSITION 2.1. *Any simple limit point solution of problem (2.23) is a regular solution of problem (2.30).*

For a proof, see Keller [39], Decker-Keller [40].

From a practical point of view, Proposition 2.1 is of fundamental importance for the following reasons.

(i) Since simple limit points for problem (2.23) are regular points for problem (2.30), the conjugate gradient algorithm can be used to compute these limit points via the least squares formulation (2.33). This property is a direct consequence of the convergence properties of the conjugate gradient algorithm mentioned in § 2.2 and discussed in details in [3].

(ii) Using a perturbation technique, bifurcation points can be approximated by simple limit points. Then the solution methods described in the present § 2 can be applied; several examples of such situations will be discussed in § 4.

**2.4. Implementation of the arc length continuation method.** To help potential users of continuation methods discussed in the previous sections, we summarize here the essentials of these methods. We solve the nonlinear problem (2.23) via the solution of a family (parametrized by  $s$ ) of nonlinear system (2.26), (2.27). In practice we approximate (2.26), (2.27) by the discrete family of nonlinear systems described below, where  $\Delta s$  is an *arc length step, positive or negative* (possibly varying with  $n$ ) and where  $u^n \approx u(n \Delta s)$ ,  $\lambda^n \approx \lambda(n \Delta s)$ :

*Initialization.* We suppose that we know a solution  $\{u^0, \lambda^0\}$  of (2.23); we take it as origin of the arc of solutions, i.e.  $u^0 = u(0)$ ,  $\lambda^0 = \lambda(0)$ . We suppose also that we know the tangent  $(\dot{u}(0), \dot{\lambda}(0))$  satisfying (2.28a, b) (or at least an approximation to it; see Remark 2.3).

*Continuation.* Then for  $n \geq 0$ , with  $u^n, \lambda^n$  known and also  $u^{n-1}, \lambda^{n-1}$  (resp.  $u(0), \lambda(0)$  if  $n = 0$ ), we obtain  $\{u^{n+1}, \lambda^{n+1}\} \in V \times \mathbb{R}$  as the solution of:

$$(2.51) \quad S(u^{n+1}, \lambda^{n+1}) = 0,$$

and

$$(2.52) \quad (u^1 - u^0, \dot{u}(0)) + (\lambda^1 - \lambda^0)\dot{\lambda}(0) = \Delta s \quad \text{if } n = 0,$$

$$(2.53) \quad \left( u^{n+1} - u^n, \frac{u^n - u^{n-1}}{\Delta s} \right) + (\lambda^{n+1} - \lambda^n) \left( \frac{\lambda^n - \lambda^{n-1}}{\Delta s} \right) = \Delta s \quad \text{if } n \geq 1.$$

*Remark 2.3.* It may occur that obtaining  $\{\dot{u}(0), \dot{\lambda}(0)\}$  is by itself a complicated problem; however obtaining a second solution of (2.23), close to  $\{u^0, \lambda^0\}$ , may be easy (using the nonlinear least squares-conjugate gradient methods of §§ 2.1, 2.2, for example). This supposes that we are sufficiently far from a singular point. Let us denote this second solution by  $\{u^{-1}, \lambda^{-1}\}$ ; to approximate  $\{\dot{u}(0), \dot{\lambda}(0)\}$  we compute first  $\Delta s^0$  by

$$(2.54) \quad (\Delta s^0)^2 = \|u^0 - u^{-1}\|^2 + |\lambda^0 - \lambda^{-1}|^2,$$

and approximate  $\dot{u}(0), \dot{\lambda}(0)$  by

$$(2.55) \quad \frac{u^0 - u^{-1}}{\Delta s^0}, \quad \frac{\lambda^0 - \lambda^{-1}}{\Delta s^0},$$

respectively. The sign of  $\Delta s^0$  depends upon the orientation chosen for the arc of solutions and of the relative positions of  $\{u^0, \lambda^0\}$  and  $\{u^{-1}, \lambda^{-1}\}$  on it.

*Remark 2.4.* Relations (2.52), (2.53) employ difference quotients to replace or approximate the tangents used in (2.28b) and (2.29). From this idea we can derive

many other schemes for the approximation of (2.26), (2.27). Methods for the automatic adjustments of  $\Delta s$  are important but we do not discuss them here; see Rheinboldt [41]. A least squares conjugate gradient method for solving in  $V \times \mathbb{R}$  systems very close to (2.51)–(2.53) has been discussed in § 2.3. To start this algorithm we have used  $\{2u^n - u^{n-1}, 2\lambda^h - \lambda^{n-1}\}$  as an initial guess to compute  $\{u^{n+1}, \lambda^{n+1}\}$ . This is just another use of the difference approximation to the tangent, but now in (2.48).

### 3. Application to the solution of the Bratu problem.

**3.1. Formulation of the problem, properties of its solutions.** As a first test problem for the solution techniques discussed in § 2 we consider the numerical solution of a modified *Bratu problem*, i.e. find a solution  $u$  of the nonlinear boundary value problem:

$$(3.1) \quad \begin{aligned} -\Delta u &= \lambda e^u + f \quad \text{in } \Omega, \\ u &= 0 \quad \text{on } \partial\Omega. \end{aligned}$$

Here  $\Omega$  is a *bounded* domain of  $\mathbb{R}^N$  and  $\partial\Omega$  its boundary. We denote by  $x = \{x_i\}_{i=1}^N$  the generic point of  $\mathbb{R}^N$  and define  $dx$  by  $dx = dx_1 \cdots dx_N$ . The (quite classical) Sobolev–Hilbert space

$$(3.2) \quad H_0^1(\Omega) = \left\{ v \left| v \in L^2(\Omega), \frac{\partial v}{\partial x_i} \in L^2(\Omega), \forall i = 1, \dots, N, v = 0 \text{ on } \partial\Omega \right. \right\},$$

equipped with the scalar product

$$(3.3) \quad (u, v)_{H_0^1(\Omega)} = \int_{\Omega} \nabla u \cdot \nabla v \, dx$$

and the corresponding form

$$(3.4) \quad \|v\|_{H_0^1(\Omega)} = \left( \int_{\Omega} |\nabla v|^2 \, dx \right)^{1/2},$$

provides a functional framework well suited to the solution of (3.1) by variational methods, and most particularly by those discussed in § 2.

For simplicity we consider only situations for which  $f$  is a *nonnegative constant* ( $f \equiv 0$  in the Bratu case). We suppose also that  $\lambda \geq 0$ , since problem (3.1) has a unique solution in  $H_0^1(\Omega)$  if  $\lambda \leq 0$ ; such a result can be proved using *monotonicity* methods, like those discussed in e.g. Lions [7], and based on the fact that the operator

$$v \rightarrow -\Delta v - \lambda e^v - f$$

is *monotone* over  $H_0^1(\Omega)$  if  $\lambda < 0$ . If  $\lambda > 0$ , problem (3.1) and closely related nonlinear problems, have been considered by many authors; with regard to recent publications let us mention among others Crandall–Rabinowitz [8], [9], Amann [10], Mignot–Puel [11], Mignot–Murat–Puel [12], Keller–Cohen [30], Keener–Keller [31]; in particular we find in [12] an interesting discussion showing relationships between (3.1) and *combustion phenomena* and in [30] a relationship to joule-heating in conductors. The following has been proved for  $\lambda > 0$ :

*There exists a critical value of  $\lambda$ , say  $\lambda^* > 0$ , such that:*

- (i) *If  $\lambda > \lambda^*$ , then problem (3.1) has no solution.*
- (ii) *If  $\lambda \in ]0, \lambda^*]$  (resp.  $\lambda \in ]0, \lambda^*[$ ), then problem (3.1) has at least one solution (resp. two solutions) belonging to  $H_0^1(\Omega) \cap W^{2,p}(\Omega) \forall p \geq 1$ , where*

$$W^{2,p}(\Omega) = \left\{ v \left| v, \frac{\partial v}{\partial x_i}, \frac{\partial^2 v}{\partial x_i \partial x_j} \in L^p(\Omega), \forall 1 \leq i, j \leq N \right. \right\}.$$

(iii) If  $\lambda = \lambda^*$  there exists a unique  $u^* \in H_0^1(\Omega) \cap W^{2,p}(\Omega)$ ,  $\forall p \geq 1$ ; moreover  $\{u^*, \lambda^*\}$  is a simple limit point for the equation

$$S(u, \lambda) = 0,$$

where the operator  $S$  is defined over  $H_0^1(\Omega) \times \mathbb{R}$  by

$$S(v, \mu) = -\Delta v - \mu e^v - f.$$

In the above theoretical references it is also proved that these solutions which are not limit points are regular solutions. It follows from all these properties that the solution techniques discussed in § 2 can be applied to the solution of (3.1) if  $\lambda > 0$ . Their application to the computational solution of (3.1) requires, however, a *finite dimensional approximation* of this last problem; such an approximation—by *finite element methods*—is considered in the following sections. Problem (3.1) has been investigated numerically by, among others, Kikuchi [13], Simpson [14], Moore–Spence [15], Chan–Keller [16] (by arc length continuation and multigrid finite difference methods), Reinhart [3], to which we refer for more details and further references.

**3.2. Finite element approximation of the Bratu problem.**

**3.2.1. Variational formulation of the Bratu problem. Triangulation of  $\Omega$ . Fundamental discrete spaces.** A variational formulation of the Bratu problem (3.1), well suited to finite element approximations and to the solution techniques of § 2, is given by

(3.5) Find  $\{u, \lambda\} \in H_0^1(\Omega) \times \mathbb{R}$  such that

$$\int_{\Omega} \nabla u \cdot \nabla v \, dx = \int_{\Omega} (\lambda e^u + f)v \, dx \quad \forall v \in H_0^1(\Omega).$$

We describe only the approximation of problem (3.1) for  $N = 2$  (the one-dimensional case,  $N = 1$ , is much simpler); we suppose also for simplicity that  $\Omega$  is a polygonal domain of  $\mathbb{R}^2$ . We consider now a standard family of finite element triangulations  $\{\mathcal{T}_h\}_h$  of  $\Omega$ , i.e. for a given  $h$ ,  $\mathcal{T}_h$  is a finite collection of (*closed*) subtriangles,  $T$ , of  $\Omega$ , such that

- (i)  $\cup_{T \in \mathcal{T}_h} T = \bar{\Omega}$ ,
- (ii)  $\forall T, T' \in \mathcal{T}_h, T \neq T'$ , we have either
  - (\*)  $T \cap T' = \emptyset$ ,
  - (\*\*) or  $T, T'$  have only one vertex in common,
  - (\*\*\*) or  $T, T'$  have only a whole edge in common,
- (iii)  $h$  is the maximal length of the edges of the  $T \in \mathcal{T}_h$ .

An example of such a triangulation is shown in Fig. 3.1, for  $\Omega = ]0, 1[ \times ]0, 1[$ .

We approximate  $H_0^1(\Omega)$  by the finite-dimensional space:

(3.6)  $V_{0h} = \{v_h \mid v_h \in C^0(\bar{\Omega}), v_h|_T \in P_1 \quad \forall T \in \mathcal{T}_h, v_h = 0 \text{ on } \partial\Omega\}$

where  $P_1$  is the space of polynomials in  $x_1, x_2$  of degree  $\leq 1$ . It follows that  $\dim V_{0h} = N_{0h}$  where  $N_{0h}$  is the number of vertices of  $\mathcal{T}_h$  interior to  $\Omega$  and  $V_{0h} \subset H_0^1(\Omega)$ .

**3.2.2. Formulation of the approximate problems.** As an approximate problem it is quite natural to take:

(3.7) Find  $\{u_h, \lambda\} \in V_{0h} \times \mathbb{R}$  such that

$$\int_{\Omega} \nabla u_h \cdot \nabla v_h \, dx = \int_{\Omega} (\lambda e^{u_h} + f)v_h \, dx \quad \forall v_h \in V_{0h}.$$



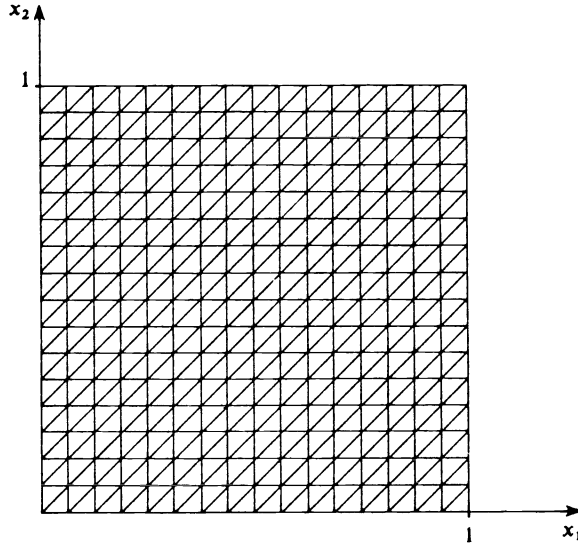


FIG. 3.1. Finite element triangulation for the Bratu problem.

Problem (3.7) is equivalent to a system of nonlinear equations in  $\mathbb{R}^{N_{0h}+1}$ . To obtain this system we suppose that the set  $\Sigma_{0h}$  of the vertices of  $\mathcal{T}_h$  has been ordered so that

$$(3.8) \quad \Sigma_{0h} = \{P_i\}_{i=1}^{N_{0h}},$$

and that to each  $P_i$  of  $\Sigma_{0h}$  we have associated the function  $w_i$  satisfying

$$(3.9) \quad w_i \in V_{0h}, w_i(P_j) = \delta_{ij} \quad \forall 1 \leq i, j \leq N_{0h}.$$

The set  $B_{0h} = \{w_i\}_{i=1}^{N_{0h}}$  is a basis of the vector space  $V_{0h}$  and we clearly have the important relation,  $\forall v_h \in V_{0h}$ :

$$(3.10) \quad v_h = \sum_{i=1}^{N_{0h}} v_h(P_i) w_i.$$

Using (3.10) in (3.7), we get the nonlinear system:

$$(3.11) \quad \sum_{j=1}^{N_{0h}} \left( \int_{\Omega} \nabla w_i \cdot \nabla w_j dx \right) u_h(P_j) = \int_{\Omega} \left( \lambda \exp \left( \sum_{j=1}^{N_{0h}} u_h(P_j) w_j \right) + f \right) w_i dx, \quad 1 \leq i \leq N_{0h}.$$

Here the unknown vector is  $\{ \{u_h(P_i)\}_{i=1}^{N_{0h}}, \lambda \} \in \mathbb{R}^{N_{0h} \times \mathbb{R}}$ .

Since  $\nabla w_i, \nabla w_j$  are piecewise constant functions, the calculation of the left-hand side is an easy task. The integrals occurring on the right-hand side of (3.11) can be calculated exactly. However in order to reduce the computational work, we evaluate these integrals approximately. Two possibilities are as follows:

(i) Calculate  $\int_{\Omega} e^{u_h} w_i dx$  using the *two-dimensional Simpson rule* on each triangle  $T \in \mathcal{T}_h$ , i.e.

$$(3.12) \quad \int_T \phi(x) dx \approx \frac{1}{3} \text{measure}(T) \sum_{j=1}^3 \phi(m_{jT}),$$

where  $m_{1T}, m_{2T}, m_{3T}$  are the midpoints of the three edges of  $T$ . Formula (3.12) is exact

if  $\phi \in P_2$  ( $P_2 =$  space of polynomials of degree  $\leq 2$ ). We need to apply Simpson's rule only on those triangles of  $\mathcal{T}_h$  with  $P_i$  as a common vertex.

(ii) Apply to  $\int_{\Omega} e^{u_h} w_i dx$  the *two-dimensional trapezoidal rule*, i.e.

$$(3.13) \quad \int_T \phi(x) dx \approx \frac{1}{3} \text{measure}(T) \sum_{j=1}^3 \phi(P_{jT}),$$

where  $P_{jT}$ ,  $j = 1, 2, 3$  are the vertices of  $T$ ; formula (3.13) is exact if  $\phi \in P_1$ . If  $\mathcal{T}_h$  is a regular triangulation, like the one of Fig. 3.1, using (3.13) to calculate the right-hand sides of (3.11) we recover classical finite difference schemes for the discretization of (3.1).

**3.3. Numerical solution of the discrete Bratu problem by arc length continuation methods.** We now apply the continuation methods of §§ 2.3, 2.4 to solve the discrete Bratu problem (3.7). This leads to the following algorithm:

(a) *Initialization.* Set

$$(3.14) \quad \lambda^0 = 0.$$

The corresponding  $u_h^0$  is the unique solution of the following discrete linear Dirichlet problem (given in variational form).

Find  $u_h^0 \in V_{0h}$  such that

$$(3.15) \quad \int_{\Omega} \nabla u_h^0 \cdot \nabla v_h dx = \int_{\Omega} f v_h dx \quad \forall v_h \in V_{0h}.$$

This is equivalent to a linear system (obtained by setting  $\lambda = 0$  in (3.11)) whose matrix is *symmetric* and *positive definite*. We take  $\{u_h^0, 0\}$  as the point on the arc of solutions  $\{u_h(s), \lambda(s)\}$  for which  $s = 0$ . Denote  $dX(s)/ds$  by  $\dot{X}(s)$  for  $X = u_h$  or  $X = \lambda$ . Then by differentiation of (3.7), with respect to  $s$ , we obtain at  $s = 0$ :

$$(3.16) \quad \int_{\Omega} \nabla \dot{u}_h(0) \cdot \nabla v_h dx = \dot{\lambda}(0) \int_{\Omega} e^{u_h^0} v_h dx \quad \forall v_h \in V_{0h},$$

$$\dot{u}_h(0) \in V_{0h}.$$

We also require as a definition of  $s$ :

$$(3.17) \quad \int_{\Omega} |\nabla \dot{u}_h(0)|^2 dx + \dot{\lambda}^2(0) = 1.$$

Define  $\hat{u}_h$  as the solution of

$$(3.18) \quad \hat{u}_h \in V_{0h},$$

$$\int_{\Omega} \nabla \hat{u}_h \cdot \nabla v_h dx = \int_{\Omega} e^{u_h^0} v_h dx \quad \forall v_h \in V_{0h}.$$

Then from (3.16)-(3.18) we have

$$(3.19) \quad \dot{u}_h(0) = \dot{\lambda}(0) \hat{u}_h,$$

$$(3.20) \quad \dot{\lambda}^2(0) = \left( 1 + \int_{\Omega} |\nabla \hat{u}_h|^2 dx \right)^{-1}.$$

Since we are interested in solving the Bratu problem for  $\lambda > 0$  and we have set  $\lambda(0) = 0$

we must orient the solution arc in such a way that  $d\lambda/ds (= \dot{\lambda}) \geq 0$ . Thus (3.20) yields:

$$(3.21) \quad \dot{\lambda}(0) = \left( 1 + \int_{\Omega} |\nabla \hat{u}_h|^2 dx \right)^{-1/2}.$$

(b) *Continuation.* With  $\Delta s (> 0)$  as a fixed step in arc length, we define for  $n \geq 0$  an approximation  $\{u_h^{n+1}, \lambda^{n+1}\} (\in V_{0h} \times \mathbb{R})$  of  $\{u_h((n+1)\Delta s), \lambda((n+1)\Delta s)\}$  as the solution of the following nonlinear variational system:

Find  $\{u_h^{n+1}, \lambda^{n+1}\} \in V_{0h} = \mathbb{R}$  such that

$$(3.22a) \quad \int_{\Omega} \nabla u_h^{n+1} \cdot \nabla v_h dx = \int_{\Omega} (\lambda^{n+1} e^{u_h^{n+1}} + f)v_h dx \quad \forall v_h \in V_{0h},$$

$$(3.22b) \quad \int_{\Omega} \nabla(u_h^1 - u_h^0) \cdot \nabla \dot{u}_h(0) dx + (\lambda^1 - \lambda^0)\dot{\lambda}(0) = \Delta s \quad \text{if } n = 0,$$

$$(3.22c) \quad \int_{\Omega} \nabla(u_h^{n+1} - u_h^n) \cdot \nabla \left( \frac{u_h^n - u_h^{n-1}}{\Delta s} \right) dx + (\lambda^{n+1} - \lambda^n) \left( \frac{\lambda^n - \lambda^{n-1}}{\Delta s} \right) = \Delta s \quad \text{if } n \geq 1.$$

To solve the nonlinear system (3.22), we use the *nonlinear least squares conjugate gradient* techniques of § 2.3. We give a detailed description of the operations involved in the solution process for this first application.

A convenient nonlinear least squares formulation of (3.22) is:

$$(3.23) \quad \text{Find } \{u_h^{n+1}, \lambda^{n+1}\} \in V_{0h} \times \mathbb{R} \text{ such that}$$

$$J_{n+1}(u_h^{n+1}, \lambda^{n+1}) \leq J_{n+1}(w_h, \mu) \quad \forall \{w_h, \mu\} \in V_{0h} \times \mathbb{R}.$$

Here the functional  $J_{n+1}(\cdot, \cdot)$  is defined by

$$J_{n+1}(w_h, \mu) = \frac{1}{2} \int_{\Omega} |\nabla \tilde{w}_h|^2 dx + \frac{1}{2} |\tilde{\mu}|^2$$

where  $\tilde{w}_h$  and  $\tilde{\mu}$  are nonlinear functions of  $\{w_h, \mu\}$  obtained as the solutions of the *linear* problems:

$$(3.24) \quad \tilde{w}_h \in V_{0h},$$

$$\int_{\Omega} \nabla \tilde{w}_h \cdot \nabla v_h dx = \int_{\Omega} \nabla w_h \cdot \nabla v_h dx - \int_{\Omega} (\mu e^{w_h} + f)v_h dx \quad \forall v_h \in V_{0h},$$

$$(3.25) \quad \tilde{\mu} = \int_{\Omega} \nabla(w_h - u_h^n) \cdot \nabla \left( \frac{u_h^n - u_h^{n-1}}{\Delta s} \right) dx + (\mu - \lambda^n) \left( \frac{\lambda^n - \lambda^{n-1}}{\Delta s} \right) + \Delta s.$$

In this particular case, the conjugate gradient algorithm (2.37)-(2.46) reduces to:

*Step 0: Initialization.* For a given

$$(3.26) \quad \{u_h^0, \lambda^0\} \in V_{0h} \times \mathbb{R},$$

compute  $\{g_u^0, g_\lambda^0\} \in V_{0h} \times \mathbb{R}$  as the solution of:

$$(3.27) \quad \int_{\Omega} \nabla g_u^0 \cdot \nabla v_h dx = \left\langle \frac{\partial J_{n+1}}{\partial u_h}(u_h^0, \lambda^0), v_h \right\rangle \quad \forall v_h \in V_{0h},$$

$$(3.28) \quad g_\lambda^0 = \frac{\partial J_{n+1}}{\partial \lambda}(u_h^0, \lambda^0).$$

Then set

$$(3.29) \quad \{z_u^0, z_\lambda^0\} = \{g_u^0, g_\lambda^0\}$$

and for  $m \geq 0$ , assuming that  $\{u_h^m, \lambda^m\}$ ,  $\{g_u^m, g_\lambda^m\}$ ,  $\{z_u^m, z_\lambda^m\}$  are known, compute  $\{u_h^{m+1}, \lambda^{m+1}\}$ ,  $\{g_u^{m+1}, g_\lambda^{m+1}\}$ ,  $\{z_u^{m+1}, z_\lambda^{m+1}\}$  by:

Step 1: *Descent.* Find  $\rho_m \in \mathbb{R}$  such that,  $\forall \rho \in \mathbb{R}$ :

$$(3.30) \quad J_{n+1}(u_h^m - \rho_m z_u^m, \lambda^m - \rho_m z_\lambda^m) \leq J_{n+1}(u_h^m - \rho z_u^m, \lambda^m - \rho z_\lambda^m).$$

Then set

$$(3.31) \quad u_h^{m+1} = u_h^m - \rho_m z_u^m, \lambda^{m+1} = \lambda^m - \rho_m z_\lambda^m.$$

Step 2: *Calculate new descent direction.* Compute  $\{g_u^{m+1}, g_\lambda^{m+1}\} \in V_{0h} \times \mathbb{R}$  as the solution of:

$$(3.32) \quad \int_\Omega \nabla g_u^{m+1} \cdot \nabla v_h \, dx = \left\langle \frac{\partial J_{n+1}}{\partial u_h}(u_h^{m+1}, \lambda^{m+1}), v_h \right\rangle \quad \forall v_h \in V_{0h},$$

$$(3.33) \quad g_\lambda^{m+1} = \frac{\partial J_{n+1}}{\partial \lambda}(u_h^{m+1}, \lambda^{m+1}).$$

Evaluate

$$(3.34) \quad \gamma_m = \frac{\int_\Omega \nabla(g_u^{m+1} - g_u^m) \cdot \nabla g_u^{m+1} \, dx + (g_\lambda^{m+1} - g_\lambda^m) g_\lambda^{m+1}}{\int_\Omega |\nabla g_u^{m+1}|^2 \, dx + |g_\lambda^{m+1}|^2}$$

and

$$(3.35) \quad z_u^{m+1} = g_u^{m+1} + \gamma_m z_u^m, z_\lambda^{m+1} = g_\lambda^{m+1} + \gamma_m z_\lambda^m.$$

Then set  $m = m + 1$  and return to step 1.

As in § 2.4 we can use  $\{2u_h^n - u_h^{n-1}, 2\lambda^n - \lambda^{n-1}\}$  in (3.26). The partial derivatives  $\partial J_{n+1}/\partial u_h, \partial J_{n+1}/\partial \lambda$  (occurring in (3.32), (3.33)) can be evaluated using the derivative calculation technique of § 2.2. At  $\{w_h, \mu\}$  this gives:

$$(3.36) \quad \left\langle \frac{\partial J_{n+1}}{\partial u_h}(w_h, \mu), v_h \right\rangle = \int_\Omega \nabla \tilde{w}_h \cdot \nabla v_h \, dx - \mu \int_\Omega e^{w_h} \tilde{w}_h v_h \, dx + \tilde{\mu} \int_\Omega \nabla \left( \frac{u_h^n - u_h^{n-1}}{\Delta s} \right) \cdot \nabla v_h \, dx \quad \forall v_h \in V_{0h},$$

$$(3.37) \quad \frac{\partial J_{n+1}}{\partial \lambda}(w_h, \mu) = \tilde{\mu} \left( \frac{\lambda^n - \lambda^{n-1}}{\Delta s} \right) - \int_\Omega e^{w_h} \tilde{w}_h \, dx.$$

Here of course  $\{\tilde{w}_h, \tilde{\mu}\}$  are obtained from  $\{w_h, \mu\}$ , through the solution of (3.24), (3.25).

As convergence test we took

$$(3.38) \quad J_{n+1}(u_h^{m+1}, \lambda^{m+1}) \leq \varepsilon.$$

For the examples we used  $\varepsilon = 10^{-6}$ .

**3.4. Numerical examples.** We have employed the above indicated procedures to solve problem (3.1) in three specific cases:

- A.  $f \equiv 0, \quad \Omega \equiv ]0, 1[;$
- B.  $f \equiv 1, \quad \Omega \equiv ]0, 1[;$
- C.  $f \equiv 0, \quad \Omega \equiv ]0, 1[ \times ]0, 1[.$

Cases A and B have been discretized by *one-dimensional finite elements*, using a space discretization step  $h = 0.1$ . Case C has been approximated using a triangulation  $\mathcal{T}_h$  as shown in Fig. 3.1, consisting of 512 triangles. The unknowns are the values taken by the approximate solution  $u_h$  at the interior nodes of  $\mathcal{T}_h$ ; we have 225 such nodes. The continuation algorithm described in § 3.3 has been applied with fixed  $\Delta s = 0.1$ .

We show in Fig. 3.2 the variation of  $u_h(0.5, 0.5)$  (maximal value of  $u_h$ ) as a function of  $\lambda$  for case C. The numerical results agree very well with those of Kikuchi [13], obtained by quite different methods.

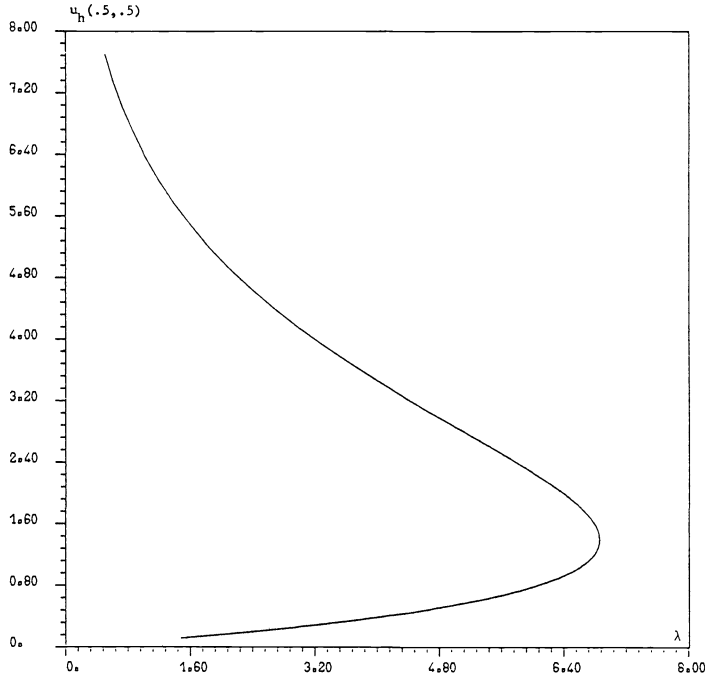


FIG. 3.2. Solution  $u_h$  at  $\{x, y\} = \{0.5, 0.5\}$  of the Bratu problem on the unit square (case C).

With  $\Delta s = 0.1$ , the solution of the above three test problems never required more than 3 to 4 iterations of the conjugate gradient algorithm (3.26), (3.35) to obtain  $\{u_h^{n+1}, \lambda^{n+1}\}$  from  $\{u_h^n, \lambda^n\}$  and  $\{u_h^{n-1}, \lambda^{n-1}\}$  via the solution of the least squares problem (3.23). This efficiency is partly due to the good initialization of algorithm (3.26)–(3.35) provided by the initial guess  $\{2u_h^n - u_h^{n-1}, 2\lambda^n - \lambda^{n-1}\}$  and partly due to the small step size of  $\Delta s$ . Using the above methods there were no difficulties close to and at the limit point.

We point out that each iteration of the conjugate gradient algorithm (3.26)–(3.35), requires the solution of several discrete linear systems with a *fixed* coefficient matrix independent of  $n$  and  $m$ ; since this matrix is symmetric and positive definite we use *only one* Cholesky factorization, taking into account the sparsity of the matrix. The solution procedure is thus quite efficient.

#### 4. Applications to the solution of bifurcation problems via perturbed bifurcations.

**4.1. Synopsis. Generalities.** In this section we discuss the numerical treatment of *nonlinear second order boundary value problems* whose branches of solutions exhibit *bifurcation*. To do this we perturb the original problem into a new one whose branches

of solutions do not bifurcate but instead have *limit or fold points*. The solution branches can then be computed by the continuation methods of § 2.3. The process is based on the use of a simple perturbation method related to the concept of *perturbed bifurcation*, see Keener-Keller [32], Matkowsky-Reiss [33]. By continuing from the perturbed to the unperturbed problem we can recover the bifurcating solution branches.

**4.2. First example: A nonlinear Dirichlet problem.** With  $\Omega$  a bounded domain of  $\mathbb{R}^N$  ( $N \geq 1$ ), we consider the solution in  $V = H_0^1(\Omega)$ , of the nonlinear Dirichlet problem:

$$(4.1) \quad \begin{aligned} -\Delta u &= \lambda u^2 + \delta && \text{in } \Omega, \\ u &= 0 && \text{on } \partial\Omega. \end{aligned}$$

Here  $\delta \in \mathbb{R}$  is the perturbing parameter. The nonperturbed problem ( $\delta = 0$ ) has two solution branches for  $\lambda \geq 0$ :

- (i) the trivial branch  $\{u, \lambda\} = \{0, \lambda\}, \forall \lambda \in \mathbb{R}$ .
- (ii) a nontrivial branch which never crosses the trivial one (see Fig. 4.1). By symmetry with respect to  $\{u, \lambda\} \rightarrow \{-u, -\lambda\}$  about  $\{0, 0\}$  we easily obtain the unperturbed solutions of (4.1) corresponding to  $\lambda \leq 0$ . Thus in general the unperturbed problem with  $\delta = 0$ , has three disjoint solution branches:  $u \equiv 0, \lambda \in \mathbb{R}$  and two ‘‘hyperbolic’’ branches, one for  $\lambda > 0$ , and one for  $\lambda < 0$ . For the perturbed problem,  $\delta \neq 0$ , it follows from Mignot-Puel [11] that only two distinct branches exist. One of these is a perturbation of one of the hyperbolic branches and it contains only regular solutions. The other is formed from the perturbed trivial branch joining the other hyperbolic branch and it contains one simple fold point. Figure 4.1 shows some of these solution branches for  $\delta \geq 0$ .

This problem with  $\delta = 0$  is not, technically, a bifurcation problem. Rather we may say that it exhibits bifurcation at  $\lambda = +\infty$ . However it furnishes a clear example of our perturbation techniques. We now describe our procedure for computing the nontrivial branch of solutions of the unperturbed problem (4.1) with  $\lambda \geq 0$ . With fixed  $\delta > 0$ , ‘‘sufficiently small’’, we solve (4.1) by a continuation method, as described in § 2.3, using  $\{u, \lambda\} = \{u_\delta^0, 0\}$  as starting point. Here  $u_\delta^0 \in H_0^1(\Omega)$  is the solution of

$$(4.2) \quad \begin{aligned} -\Delta u_\delta^0 &= \delta && \text{in } \Omega, \\ u_\delta^0 &= 0 && \text{on } \partial\Omega. \end{aligned}$$

For  $\delta > 0$  and sufficiently small, the upper part,  $C_\delta^+$  of the branch of solutions of the perturbed problem, away from the limit point  $\{u_{\delta_c}, \lambda_{\delta_c}\}$  is a good approximation of the nontrivial branch of solutions of the nonperturbed problem. We take two distinct points on  $C_\delta^+$ , say  $\{u_{\delta_1}, \lambda_{\delta_1}\}$  and  $\{u_{\delta_2}, \lambda_{\delta_2}\}$  and compute the nontrivial solutions of the unperturbed problem corresponding to values  $\lambda = \lambda_{\delta_1}$  and  $\lambda = \lambda_{\delta_2}$ . These solutions can be obtained using simply the least squares conjugate gradient method of §§ 2.1, 2.2 (i.e. without continuation), taking  $u_{\delta_1}$  and  $u_{\delta_2}$  as starting points. If necessary, however, continuation with respect to  $\delta$  can be used to reach the value  $\delta = 0$ .

Once two distinct solutions (sufficiently close to each other) on the nontrivial branch of solutions of the unperturbed problem have been obtained, we can use continuation, again, to compute the whole unperturbed branch. Fig. 4.1 illustrates the indicated process. Indeed the curves in this figure are the results of computation using  $\Omega \equiv ]0, 1[$  in (4.1). We discuss these calculations below where also the influence of step size  $\Delta s$  and other factors are considered.

The above technique has been applied to compute the nontrivial solutions of more complicated nonlinear boundary value problems. We discuss some such examples in

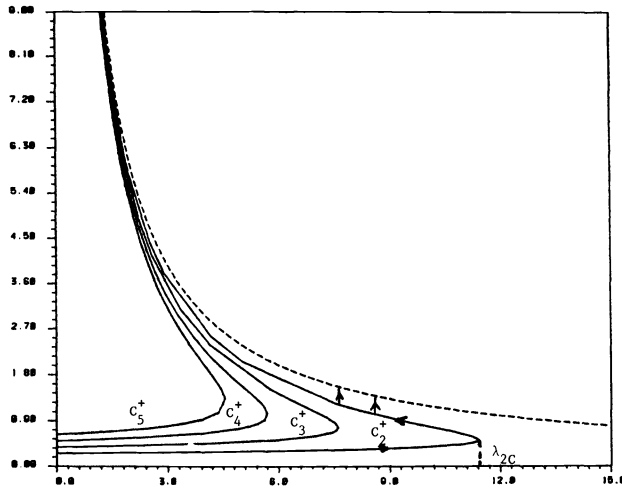


FIG. 4.1

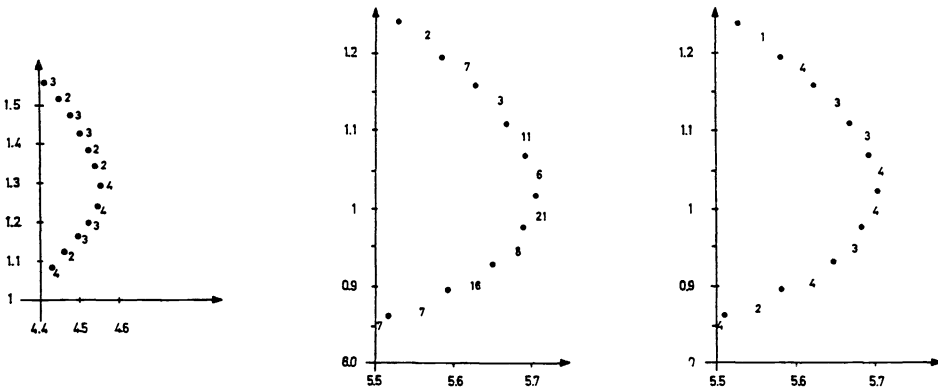
§§ 4.3 and 4.4. The *Von Karman equations* for nonlinear plates are treated in Reinhart [3], [17].

*Computational Results and Tests.* The methods of § 2 were applied to (4.1) with  $\Omega \equiv ]0, 1[$  for  $\delta = 5, 4, 3, 2, 0$ . The nontrivial branch of solutions corresponding to  $\delta = 0$  is obtained by the method indicated on Fig. 4.1. We have used  $h = 0.1$  and  $\Delta s = 0.1$  for the space discretization and the continuation algorithm, respectively. The numerical results are shown on Fig. 4.1, where we have plotted  $\max_{x \in [0,1]} u_h(x) = u_h(.5)$  versus  $\lambda$ .

The computed results agree with those obtained elsewhere by other methods.

Using (4.1) with  $\delta = 5$  as a test problem we show in Fig. 4.2(a) the number of conjugate gradient iterations necessary to solve the least squares problem encountered at each step of the continuation process. The convergence test is as indicated in (3.38), with  $\epsilon = 10^{-6}$ .

If one takes  $\gamma_n = 0$  in algorithm (2.37)-(2.46) (instead of  $\gamma_n$  given by (2.45)) we recover a *steepest descent* algorithm for solving the least squares problem (2.33). In the particular case of (4.1) with  $\delta = 4$  we have done a comparison between the performances of the steepest descent and conjugate gradient algorithms when applied to the continuation solution. The computed results are summarized on Fig. 4.2(b)



(a) Conjugate gradient ( $\delta = 5$ ).

(b) Steepest descent ( $\delta = 4$ ).

(c) Conjugate gradient ( $\delta = 4$ ).

FIG. 4.2

(steepest descent) and 4.2(c) (conjugate gradient). They show clearly the superiority of the conjugate variant in the neighborhood of the limit point. Note that the steepest descent case “oscillates”, each step taking more or less iterates depending upon the previous rate of convergence. This effect is less apparent in the conjugate gradient case. Finally we note some effects of the size of  $\Delta s$  upon the convergence of our continuation method, *particularly in the neighborhood of the limit point.*

(a) If  $\Delta s$  is too large, the algorithm does not converge close to the limit point. We can explain this behavior by the fact that the initial guess at the solution, provided by  $\{2u_h^n - u_h^{n-1}, 2\lambda^n - \lambda^{n-1}\}$  or  $\{u_h^n, \lambda^n\}$  is too far from the branch of solutions.

(b) The smaller the  $\Delta s$ , the smaller is the number of iterations close to the limit points. However if we are sufficiently far from the limit point the number of iterations is quite small and essentially independent of  $\Delta s$ .

(c) The smaller the  $\Delta s$ , the better is the approximation to the location of the limit point.

In conclusion, we should use large  $\Delta s$  if we are sufficiently far from the limit point, and decrease  $\Delta s$  if we are close to the limit point (further details concerning the choice of  $\Delta s$  may be found in [3], [5], in Rheinboldt [41] and in Perozzi [34]).

**4.3. Bifurcation from a trivial branch.**

**4.3.1. Synopsis. Generalities.** In this section we study simple nonlinear eigenvalue problems with bifurcations from the trivial branch. In the perturbed form, these problems are:

*Find  $u \in H_0^1(\Omega)$  such that*

$$(4.3) \quad \begin{aligned} -\Delta u &= \lambda u + f(u, \lambda) + \delta \quad \text{in } \Omega, \\ u &= 0 \quad \text{on } \partial\Omega. \end{aligned}$$

Here  $f$  satisfies:

$$(4.4) \quad f(0, \lambda) = 0 \quad \forall \lambda \in \mathbb{R}$$

and

$$(4.5) \quad f'_u(0, \lambda) = 0 \quad \forall \lambda \in \mathbb{R}.$$

For  $\delta = 0$  we note that  $\{u, \lambda\} \equiv \{0, \lambda\}$  is a solution of (4.3) for all  $\lambda \in \mathbb{R}$ . This is the trivial branch and we seek nontrivial branches bifurcating from it. The linearized problem about  $u = 0$ , reduces to:

$$(4.6) \quad \begin{aligned} -\Delta w &= \lambda w \quad \text{in } \Omega, \\ w &= 0 \quad \text{on } \partial\Omega. \end{aligned}$$

It is well known (i.e. Crandall-Rabinowitz [18] or Keller-Langford [36]) that if  $\lambda_i$  is an eigenvalue of *multiplicity one* of (4.6), then the pair  $\{0, \lambda_i\}$  is a simple *bifurcation point* for solutions of the unperturbed problem:  $\delta \equiv 0$  in (4.3). If  $w_i$  is a corresponding eigenfunction then it is also well known (i.e. Keller-Langford [36] or Brezzi-Rappaz-Raviart [19]) that the bifurcation is symmetric for  $\alpha_i = 0$  and asymmetric or transverse for  $\alpha_i \neq 0$  where:

$$(4.7) \quad \alpha_i \equiv \int_{\Omega} w_i^3(x) \int_{\Omega} G(x, \xi) f''_{uu}(0, \lambda_i) d\xi dx.$$

Here  $G(x, \xi)$  is the Green’s function for  $(-\Delta)$  on  $\Omega$ . These cases are illustrated by the curves for  $\delta = 0$  in Figs. 4.3 and 4.4, respectively. If  $\delta \neq 0$  we have the local behavior



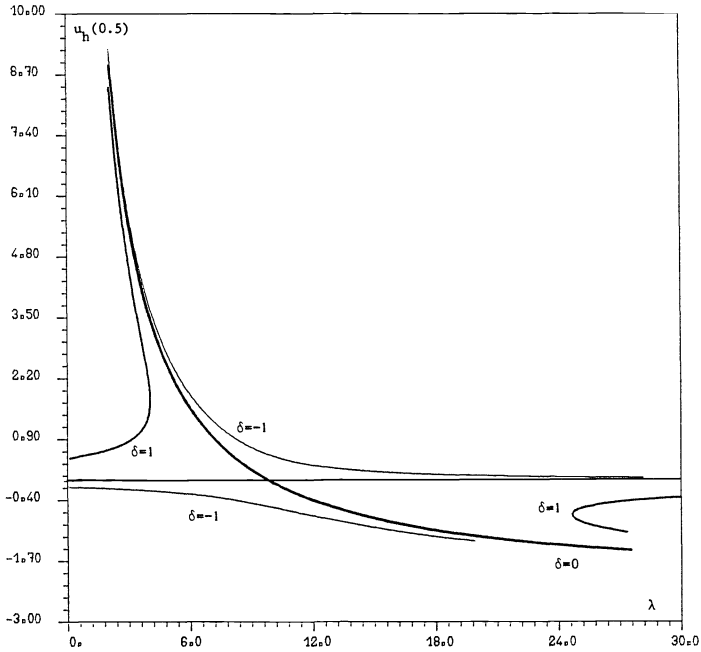


FIG. 4.3.  $-u'' = \lambda u + \lambda u^2/2 + \delta$  in  $]0, 1[$ ,  $u(0) = u(1) = 0$ .

indicated on Figs. 4.3 and 4.4, for the solutions of the perturbed problem (4.3). These configurations are called perturbed bifurcation in Keener-Keller [32] or imperfect bifurcations in Matkowsky-Reiss [33]. We shall study in particular the cases of (4.3) in which

(4.8) 
$$f(u, \lambda) \equiv \frac{\lambda}{2} u^2,$$

and

(4.9) 
$$f(u, \lambda) \equiv -u^3.$$

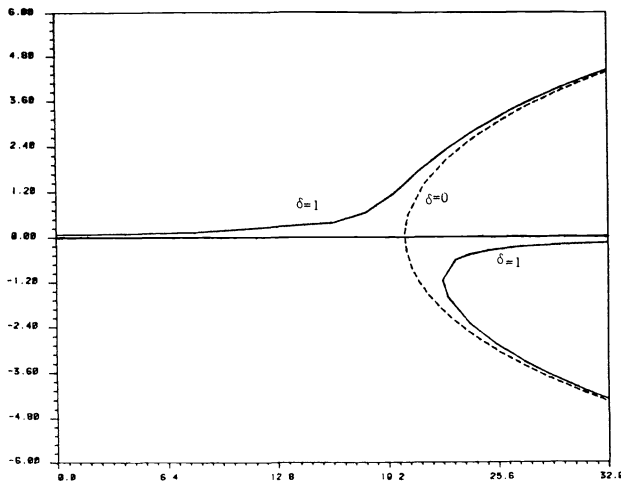


FIG. 4.4.  $-\Delta u = \lambda u - u^3 + \delta$  in  $]0, 1[ \times ]0, 1[$  ( $= \Omega$ ),  $u = 0$  on  $\partial\Omega$ .

It easily follows from (4.6) and (4.7) that for  $\Omega \equiv ]0, 1[$ :

$$(4.10a) \quad \alpha_i = 0, \forall i = 1, 2, \dots \quad \text{when } f \text{ is as in (4.9),}$$

$$(4.10b) \quad \alpha_i = 0 \text{ for } i = 2, 4, 6, \dots \quad \text{when } f \text{ is as in (4.8),}$$

$$\alpha_i \neq 0 \text{ for } i = 1, 3, 5, \dots$$

Similarly for  $\Omega \equiv ]0, 1[ \times ]0, 1[$  it follows that for the smallest eigenvalue  $\lambda_{11}$ :

$$(4.11) \quad \alpha_{11} \begin{cases} = 0 & \text{for (4.9),} \\ \neq 0 & \text{for (4.8).} \end{cases}$$

Using *finite element approximations* and the *continuation methods* discussed in § 4.2, we have computed approximate solutions of the perturbed and unperturbed problems near the *first eigenvalue* of the linearized problems. In Fig. 4.3 we show results for  $\Omega = ]0, 1[$  and  $f(u, \lambda) = \lambda u^2/2$ .

Both perturbed and unperturbed asymmetric bifurcation phenomena at the first eigenvalue of the linearized approximate problem are illustrated. For  $f(u, \lambda) = -u^3$  and  $\Omega \equiv ]0, 1[ \times ]0, 1[$  we have symmetric bifurcation phenomenon ( $\alpha_{11} = 0$ ) at the first eigenvalue of the linearized problem. The results are shown on Fig. 4.4 for both perturbed and unperturbed cases. For more details about the numerical procedure we refer to [3]. We refer also to [17] where it is shown (theoretically and computationally) that the solutions of Von Karman equations for nonlinear plates have the same qualitative behavior as observed here for  $f(u) = -u^3$  (for the first eigenvalue of the linearized problem).

**4.4. Bifurcation from a nontrivial branch.**

**4.4.1. Formulation and properties of the solutions.** We discuss in this section the solution of the nonlinear boundary value problem of *Neumann type*:

*Find  $\{u, \lambda\} \in H^1(0, 1) \times \mathbb{R}$  such that:*

$$(4.12) \quad -\frac{u''}{\pi^2} + u = \lambda e^u \text{ on } ]0, 1[,$$

$$u'(0) = u'(1) = 0.$$

Problem (4.12) has a branch of solutions  $\{u, \lambda\}$  with  $u = \text{const.}$  on  $]0, 1[$ . This constant is any root of

$$(4.13a) \quad u = \lambda e^u.$$

Alternatively each  $u \in \mathbb{R}$  is a root of (4.13a) for the value

$$(4.13b) \quad \lambda = u e^{-u}.$$

The “almost trivial” solution branch of (4.12) given by (4.13) is shown in Fig. 4.5a.

To find solutions bifurcating from the nontrivial branch, we note that the linearized form of (4.12) about the nontrivial branch is

$$(4.14) \quad -\frac{w''}{\pi^2} + w = \lambda e^u w \quad \text{in } ]0, 1[,$$

$$w'(0) = w'(1) = 0.$$

It easily follows that  $\lambda e^u$  must have one of the values

$$(4.15) \quad \lambda e^u = 1 + k^2, \quad k = 0, 1, \dots$$

Since the relation  $u = \lambda e^u$  holds, we find that the set of "singular" points  $\{u, \lambda\} = \{u_k, \lambda_k\}$  is a *discrete* set defined for  $k = 0, 1, \dots$ , by:

$$(4.16) \quad u_k = 1 + k^2, \quad \lambda_k = (1 + k^2) e^{-(1+k^2)}.$$

We can take as eigenfunctions  $w_k$  in (4.14):

$$(4.17) \quad w_k(x) = \cos k\pi x, \quad k = 0, 1, \dots.$$

The first "singular" point (obtained by taking  $k = 0$  in (4.16)) is  $\{1, e^{-1}\}$ .

We can show that it is a *simple limit point* for problem (4.12). This reduces to showing that  $\pi^2 e$  is not in the range of:  $-d^2/dx^2 \phi$  subject to  $\phi'(0) = \phi'(1) = 0$ . But this follows since  $\pi^2 e \neq 0$ .

On the other hand  $\{u_1, \lambda_1\} = \{2, 2e^{-2}\}$  is a *simple bifurcation* point and it can be proved (using e.g., [36] or [19]) that the bifurcation at  $\{u_1, \lambda_1\}$  is a *symmetric* one. All points  $\{u_k, \lambda_k\}$  for  $k > 1$  are also bifurcation points.

**4.4.2. Numerical results.** To compute the nonconstant solutions of (4.12) we use that combination of *finite element approximation* and *continuation techniques* already used in the previous sections. To avoid difficulties close to the bifurcation points during the continuation process we introduce a *perturbation* of the problem in the *boundary* conditions to get

$$(4.18) \quad \begin{aligned} -\frac{u''}{\pi^2} + u &= \lambda e^u \quad \text{on } ]0, 1[, \\ -u'(0) &= u'(1) = \delta. \end{aligned}$$

We observe that if  $\{u, \lambda\}$  is a solution of (4.18) and if  $u^*$  is defined by  $u^*(x) \equiv u(1-x)$ , then  $\{u^*, \lambda\}$  is a solution with  $\delta$  replaced by  $-\delta$ . This property of course holds if  $\delta = 0$ . It holds also for the approximate problems.

With  $N$  a positive integer and  $h = 1/N$ , we define  $x_i$  by

$$x_i = ih, \quad i = 0, \dots, N,$$

and we use piecewise linear elements and the trapezoidal rule in our variational formulation. The resulting system of  $N+2$  nonlinear equations is identical to the standard finite difference formulation of the Neumann problem (4.18). Our solution algorithm is but a trivial modification of that described in § 3.

Using the continuation strategy summarized in Fig. 4.1 we have computed branches of solutions of the perturbed problem (4.18) and also the nonconstant

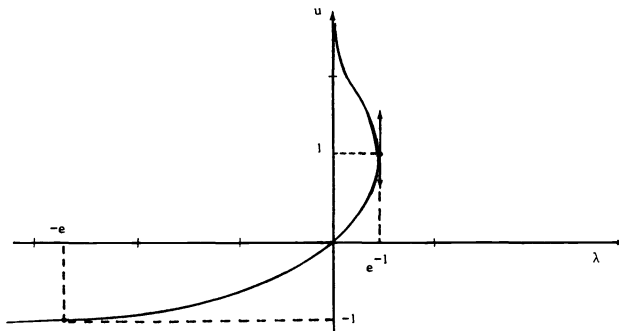


FIG. 4.5a. Constant solutions,  $u = \lambda e^u$ , of (4.12).

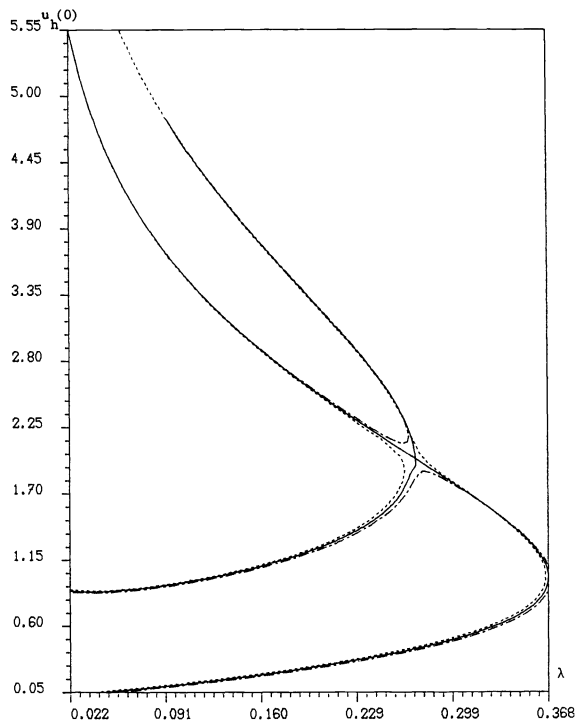


FIG. 4.5b.  $-u''/\pi^2 + u = \lambda e^u$ .  $-u'(0) = u'(1) = \delta$ . ----  $\delta = 0.01$ , —  $\delta = 0$ , -·-  $\delta = -0.01$ .

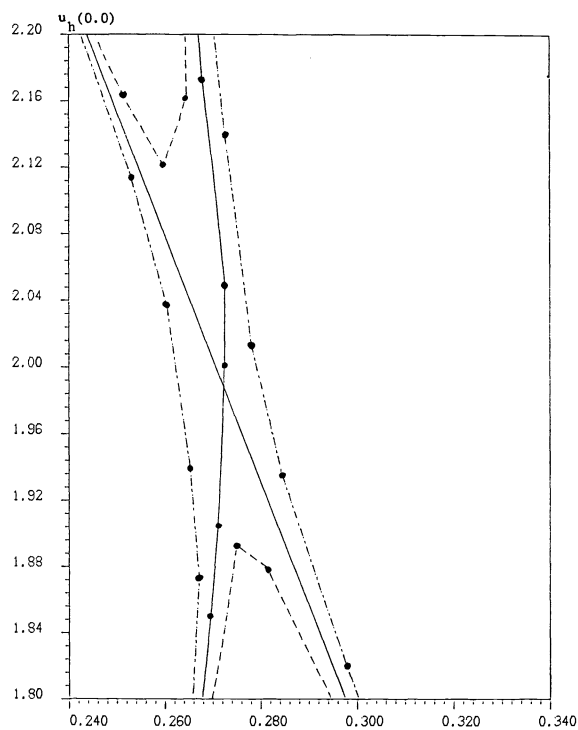


FIG. 4.6.  $-u''/\pi^2 + u = \lambda e^u$ ,  $u'(0) = -u'(1) = \delta$ . ---  $\delta = 0.01$ , —  $\delta = 0$ , -·-  $\delta = -0.01$ .

solutions of the unperturbed problem (4.12). The following results have been obtained using  $N = 20$  (i.e.  $h = 0.05$ ) for the approximate problems, and  $\delta = 0.01$  as perturbation parameter.

The variation of  $u_h(0)$  with  $\lambda$  is shown in Fig. 4.5b. (Clearly the behavior of  $u_h(1)$  is also described by this figure). Since the first bifurcation is *symmetric*, the tangent to the branch of nonconstant solutions of this bifurcation point has to be vertical; it is so with good precision. Actually, using smaller  $\Delta s$  and amplifying the vertical variations, we have shown in Fig. 4.6 the variations of  $u_h(0)$  and the above property of vertical tangent appears even more clearly.

## 5. Application to the Navier–Stokes equations for incompressible viscous fluids.

**5.1. Formulation of the Navier–Stokes equations.** Let  $\Omega$  be a domain of  $\mathbb{R}^N$  ( $N = 2, 3$  in practice) and  $\Gamma$  be its boundary. The *steady* flows of an incompressible and viscous Newtonian fluid, in  $\Omega$ , are modelled by the *Navier–Stokes equations*:

$$(5.1) \quad -\nu \Delta \mathbf{u} + (\mathbf{u} \cdot \nabla) \mathbf{u} + \nabla p = \mathbf{f} \quad \text{in } \Omega,$$

$$(5.2) \quad \nabla \cdot \mathbf{u} = 0 \quad \text{in } \Omega \quad (\text{incompressibility condition}).$$

In (5.1), (5.2):

$\mathbf{u} = \{u_i\}_{i=1}^N$  is the *flow velocity*,

$p$  is the *pressure*,

$\nu$  is a *viscosity* parameter,

$\mathbf{f}$  is a *density of external forces*,

$(\mathbf{u} \cdot \nabla) \mathbf{u}$  is a symbolic notation for the vector-function  $\{u_i \partial u_i / \partial x_j\}_{i=1}^N$ .

Typical *boundary conditions* associated with (5.1), (5.2) are

$$(5.3) \quad \mathbf{u} = \mathbf{u}_\beta \quad \text{on } \Gamma,$$

where  $\mathbf{u}_\beta$  is a given function defined over  $\Gamma$  and satisfying (from the incompressibility condition (5.2))

$$(5.4) \quad \int_{\Gamma} \mathbf{u}_\beta \cdot \mathbf{n} \, d\Gamma = 0$$

where  $\mathbf{n}$  is the outward normal unit vector on  $\Gamma$ .

The Navier–Stokes equations for incompressible viscous fluids have motivated a countless number of papers, reports, books, conferences, workshops, from both the theoretical and numerical points of view. Concentrating on books only, we mention, among others: Lions [7], Ladyzhenskaya [20], Temam [21], Girault–Raviart [22], Rautmann [23], Thomasset [24], Glowinski [42, Chapt. 7]; we refer also to the numerous references contained in these books.

It follows in particular from [7], [20], [21] that if  $\mathbf{f}$  and  $\mathbf{u}_\beta$  are sufficiently smooth, then problem (5.1), (5.2), (5.3) has a solution  $\{\mathbf{u}, p\}$  belonging to  $(H^1(\Omega))^N \times (L^2(\Omega)/\mathbb{R})$  (the pressure  $p$  is clearly determined only to within an arbitrary constant). If we suppose in addition that  $\nu$  is sufficiently large (or equivalently—if  $\nu$  is given—that  $\mathbf{f}$  and  $\mathbf{u}_\beta$  are sufficiently small), then problem (5.1)–(5.3) has unique solution in  $(H^1(\Omega))^N \times (L^2(\Omega)/\mathbb{R})$ .

**5.2. Stream function–vorticity formulation of the Navier–Stokes equations.** We suppose from now on that  $\Omega$  is a bounded domain of  $\mathbb{R}^2$ . We also assume for simplicity that  $\Omega$  is *simply connected* (see e.g. Glowinski–Pironneau [25] for the case where  $\Omega$  is multiply connected). With  $\Gamma$  the boundary of  $\Omega$ , let  $\mathbf{n}, \mathbf{s}$  be respectively the unit vector

of the outward normal at  $\Omega$  on  $\Gamma$  and the unit vector of the corresponding *oriented tangent*.

There exists from (5.2) a stream function  $\psi$  (determined only to within an arbitrary constant) such that

$$(5.5) \quad u_1 = \frac{\partial \psi}{\partial x_2}, \quad u_2 = -\frac{\partial \psi}{\partial x_1},$$

and it follows from (5.1), (5.5) that  $\psi$  satisfies the following (well-known) *nonlinear biharmonic equation*

$$(5.6) \quad \nu \Delta^2 \psi + \frac{\partial \psi}{\partial x_1} \frac{\partial}{\partial x_2} \Delta \psi - \frac{\partial \psi}{\partial x_2} \frac{\partial}{\partial x_1} \Delta \psi = \frac{\partial f_2}{\partial x_1} - \frac{\partial f_1}{\partial x_2} \quad \text{in } \Omega.$$

Concerning the *boundary conditions* we have

$$(5.7) \quad \frac{\partial \psi}{\partial s} = \mathbf{u}_\beta \cdot \mathbf{n} \quad \text{on } \Gamma.$$

Since  $\int_\Gamma \mathbf{u}_\beta \cdot \mathbf{n} \, d\Gamma = 0$ , (5.7) implies that

$$(5.8) \quad \psi(M) = \int_{M_0 M} \mathbf{u}_\beta \cdot \mathbf{n} \, d\Gamma \quad \forall M \in \Gamma,$$

where  $M_0 \in \Gamma$  ( $M_0$  can be arbitrarily chosen and we have prescribed  $\psi(M_0) = 0$ ). We also have

$$(5.9) \quad \frac{\partial \psi}{\partial n} = -\mathbf{s} \cdot \mathbf{u}_\beta \quad \text{on } \Gamma.$$

Actually (5.6), (5.8), (5.9) is a particular case of the more general family of nonlinear biharmonic problems

$$(5.10) \quad \nu \Delta^2 \psi + \frac{\partial \psi}{\partial x_1} \frac{\partial}{\partial x_2} \Delta \psi - \frac{\partial \psi}{\partial x_2} \frac{\partial}{\partial x_1} \Delta \psi = f \quad \text{in } \Omega,$$

$$(5.11) \quad \psi = g_1 \quad \text{on } \Gamma,$$

$$(5.12) \quad \frac{\partial \psi}{\partial n} = g_2 \quad \text{on } \Gamma.$$

An equivalent formulation of (5.10)–(5.12) as a nonlinear system of coupled *second order* elliptic equations is

$$(5.13) \quad -\nu \Delta \omega + \frac{\partial \omega}{\partial x_1} \frac{\partial \psi}{\partial x_2} - \frac{\partial \omega}{\partial x_2} \frac{\partial \psi}{\partial x_1} = f \quad \text{in } \Omega,$$

$$(5.14) \quad -\Delta \psi = \omega \quad \text{in } \Omega,$$

with the boundary conditions (5.11), (5.12). In (5.13), (5.14),  $\omega$  is the *voracity* of the flow.

**5.3. Variational formulations.** We suppose that  $g = \{g_1, g_2\}$  is sufficiently smooth (see [25] for the precise requirement), so that there exists  $\psi_0$  such that,  $\psi_0|_\Gamma = g_1$ ,  $(\partial \psi_0 / \partial n)|_\Gamma = g_2$ . Let us define  $V_g$  by

$$(5.15) \quad V_g = \left\{ \phi \mid \phi \in H^2(\Omega), \phi|_\Gamma = g_1, \frac{\partial \psi}{\partial n} \Big|_\Gamma = g_2 \right\};$$

then  $V_g$  is a *nonempty, closed, affine* subspace of  $H^2(\Omega)$ , where

$$H^2(\Omega) = \left\{ \phi \left| \phi, \frac{\partial \phi}{\partial x_i}, \frac{\partial^2 \phi}{\partial x_i \partial x_j} \in L^2(\Omega), \forall i, j \right. \right\}.$$

In particular  $V_0 = \{ \phi \mid \phi \in H^2(\Omega), \phi|_{\Gamma} = (\partial \phi / \partial n)|_{\Gamma} = 0 \} \equiv H_0^2(\Omega)$  is a *closed* subspace of  $H^2(\Omega)$ . We recall— $\Omega$  being bounded—that  $\phi \rightarrow (\int_{\Omega} |\Delta \phi|^2 dx)^{1/2}$  is a norm on  $V_0$  equivalent to the  $H^2$ -norm.

A *variational formulation* of (5.10)–(5.12) is then:

(5.16) Find  $\psi \in V_g$  such that  $\forall \phi \in V_0$

$$\nu \int_{\Omega} \Delta \psi \Delta \phi dx + \int_{\Omega} \Delta \psi \left( \frac{\partial \psi}{\partial x_2} \frac{\partial \phi}{\partial x_1} - \frac{\partial \psi}{\partial x_1} \frac{\partial \phi}{\partial x_2} \right) dx = \int_{\Omega} f \phi dx.$$

To obtain a variational formulation of (5.11)–(5.14) seems to be more complicated; in fact, introducing  $\theta \in L^2(\Omega)$  such that  $\theta = -\Delta \phi$  and using (5.14), it follows from (5.16) that the pair  $\{ \omega, \psi \}$  satisfies

(5.17)  $\{ \omega, \psi \} \in W_g$ ,  
and  $\forall \{ \theta, \phi \} \in W_0$  we have

$$\nu \int_{\Omega} \omega \theta dx + \int_{\Omega} \omega \left( \frac{\partial \psi}{\partial x_1} \frac{\partial \phi}{\partial x_2} - \frac{\partial \psi}{\partial x_2} \frac{\partial \phi}{\partial x_1} \right) dx = \int_{\Omega} f \phi dx,$$

where

(5.18)  $W_0 = \{ \{ \theta, \phi \} \mid \theta \in L^2(\Omega), \phi \in V_0, -\Delta \phi = \theta \text{ in } \Omega \},$

(5.19)  $W_g = \{ \{ \theta, \phi \} \mid \theta \in L^2(\Omega), \phi \in V_g, -\Delta \phi = \theta \text{ in } \Omega \}.$

Conversely if a pair  $\{ \omega, \psi \}$  satisfies (5.17), then  $\{ \omega, \psi \}$  is also a solution of the nonlinear boundary value problem (5.11)–(5.14) (and  $\psi$  a solution of (5.10)–(5.12)).

The variational formulation (5.17) of problem (5.11)–(5.14) contains *second order* derivatives in the definition of  $W_0$  and  $W_g$ ; having in view the approximation of (5.11)–(5.14) by simple finite element methods, it is of great interest to have a variational formulation of (5.11)–(5.14) containing first order derivatives, only. Such a goal is easily achieved since  $W_0$  and  $W_g$  have the alternative definitions

(5.20)  $W_0 = \left\{ \{ \theta, \phi \} \in L^2(\Omega) \times H_0^1(\Omega), \int_{\Omega} \nabla \phi \cdot \nabla q dx = \int_{\Omega} \theta q dx, \forall q \in H^1(\Omega) \right\},$

(5.21)  $W_g = \left\{ \{ \theta, \phi \} \in L^2(\Omega) \times H^1(\Omega), \phi = g_1 \text{ on } \Gamma, \right.$   
 $\left. \int_{\Omega} \nabla \phi \cdot \nabla q dx = \int_{\Omega} \theta q dx + \int_{\Gamma} g_2 q d\Gamma, \forall q \in H^1(\Omega) \right\},$

respectively. The equivalence between (5.18), (5.19) and (5.20), (5.21) follows easily from the *Green's formula*

$$\int_{\Gamma} \frac{\partial \phi}{\partial n} q d\Gamma = \int_{\Omega} \Delta \phi q dx + \int_{\Omega} \nabla \phi \cdot \nabla q dx \quad \forall q \in H^1(\Omega), \quad \forall \phi \in H^2(\Omega),$$

and the assumption that  $\Gamma (= \partial \Omega)$  is *sufficiently smooth* (or  $\Omega$  *convex*).

A variational formulation such as (5.17), (5.20), (5.21) is usually known as a *mixed variational formulation*.

**5.4. Continuation solution of problem (5.10)–(5.12).**

**5.4.1. Synopsis.** We apply now the solution methods of § 2 to the nonlinear boundary value problem (5.10)–(5.12). As parameter  $\lambda$  we choose  $\lambda = 1/\nu$ ;  $\lambda$  is directly proportional to the *Reynolds number* if we fix the boundary conditions as  $\lambda$  varies.

We consider first (in § 5.4.2) the solution of (5.10)–(5.12) via the variational formulation (5.16); the solution of (5.10)–(5.12) via (5.17) will be discussed in § 5.4.3. A mixed finite element implementation will be discussed in § 5.5.

**5.4.2. Solution of (5.10)–(5.12) via the variational formulation (5.16).** The space  $V_0 (= H_0^2(\Omega))$  which plays a fundamental role in the sequel is equipped with the inner product

$$\{v, w\} \rightarrow \int_{\Omega} \Delta v \Delta w \, dx$$

and the corresponding norm  $v \rightarrow (\int_{\Omega} |\Delta v|^2 \, dx)^{1/2}$ .

Taking  $\lambda$  as parameter the problem to be solved is

$$(5.22) \quad \Delta^2 \psi = \lambda \left( \frac{\partial \psi}{\partial x_2} \frac{\partial}{\partial x_1} \Delta \psi - \frac{\partial \psi}{\partial x_1} \frac{\partial}{\partial x_2} \Delta \psi \right) + \lambda f \quad \text{in } \Omega,$$

$$(5.23) \quad \psi = g_1 \quad \text{on } \Gamma,$$

$$(5.24) \quad \frac{\partial \psi}{\partial n} = g_2 \quad \text{on } \Gamma.$$

A variational formulation of (5.22)–(5.24) is given by:

$$(5.25) \quad \text{Find } \psi \in V_g \text{ such that } \forall \phi \in V_0$$

$$\int_{\Omega} \Delta \psi \Delta \phi \, dx = \lambda \int_{\Omega} \Delta \psi \left( \frac{\partial \psi}{\partial x_1} \frac{\partial \phi}{\partial x_2} - \frac{\partial \psi}{\partial x_2} \frac{\partial \phi}{\partial x_1} \right) dx + \lambda \int_{\Omega} f \phi \, dx.$$

*Description of the continuation procedure.* In the particular case of problem (5.22)–(5.24) the continuation techniques of §§ 2.3, 2.4 lead to the following algorithm:

(a) *Initialization.*

$$(5.26) \quad \text{Take } \lambda^0 = 0;$$

the corresponding  $\psi^0$  is the *unique* solution of the following *linear* variational problem:

$$(5.27) \quad \text{Find } \psi^0 \in V_g \text{ such that}$$

$$\int_{\Omega} \Delta \psi^0 \Delta \phi \, dx = 0 \quad \forall \phi \in V_0.$$

Problem (5.27) is in fact equivalent to the linear biharmonic problem

$$(5.28) \quad \Delta^2 \psi^0 = 0 \quad \text{in } \Omega,$$

$$\psi^0 = g_1 \quad \text{on } \Gamma, \quad \frac{\partial \psi^0}{\partial n} = g_2 \quad \text{on } \Gamma.$$

We take  $\{\psi^0, 0\}$  as the origin of the arc of solutions passing through it, and define the arc length  $s$  by

$$(5.29) \quad (\delta s)^2 = \int_{\Omega} |\Delta \delta \psi|^2 \, dx + (\delta \lambda)^2.$$



Denote  $dX/ds$  by  $\dot{X}$ ; by differentiation of (5.25) with respect to  $s$ , we obtain at  $s=0$

$$(5.30) \quad \int_{\Omega} \Delta \dot{\psi}(0) \Delta \phi \, dx = \dot{\lambda}(0) \int_{\Omega} \Delta \psi^0 \left( \frac{\partial \psi^0}{\partial x_1} \frac{\partial \phi}{\partial x_2} - \frac{\partial \psi^0}{\partial x_2} \frac{\partial \phi}{\partial x_1} \right) dx \\ + \dot{\lambda}(0) \int_{\Omega} f \phi \, dx \quad \forall \phi \in V_0, \quad \dot{\psi}(0) \in V_0.$$

We have also by definition of  $s$

$$(5.31) \quad \int_{\Omega} |\Delta \dot{\psi}(0)|^2 \, dx + \dot{\lambda}^2(0) = 1.$$

Define  $\hat{\psi}$  as the solution of the following problem

$$(5.32) \quad \hat{\psi} \in V_0, \\ \int_{\Omega} \Delta \hat{\psi} \Delta \phi \, dx = \int_{\Omega} \Delta \psi^0 \left( \frac{\partial \psi^0}{\partial x_1} \frac{\partial \phi}{\partial x_2} - \frac{\partial \psi^0}{\partial x_2} \frac{\partial \phi}{\partial x_1} \right) dx + \int_{\Omega} f \phi \, dx \quad \forall \phi \in V_0.$$

We clearly have from (5.30)–(5.32), that

$$(5.33) \quad \dot{\psi}(0) = \dot{\lambda}(0) \hat{\psi},$$

$$(5.34) \quad \dot{\lambda}(0) = \left( 1 + \int_{\Omega} |\Delta \hat{\psi}|^2 \, dx \right)^{-1/2}.$$

b) *Continuation.* With  $\Delta s (>0)$  an increment of arc length, we define for  $n \geq 0$  an approximation  $\{\psi^{n+1}, \lambda^{n+1}\} (\in V_g \times \mathbb{R})$  of  $\{\psi((n+1)\Delta s), \lambda((n+1)\Delta s)\}$  as the solution of the following nonlinear variational system:

Find  $\{\psi^{n+1}, \lambda^{n+1}\} \in V_g \times \mathbb{R}$  such that

$$(5.35a) \quad \int_{\Omega} \Delta \psi^{n+1} \Delta \phi \, dx = \lambda^{n+1} \int_{\Omega} \Delta \psi^{n+1} \left( \frac{\partial \psi^{n+1}}{\partial x_1} \frac{\partial \phi}{\partial x_2} - \frac{\partial \psi^{n+1}}{\partial x_2} \frac{\partial \phi}{\partial x_1} \right) dx \\ + \lambda^{n+1} \int_{\Omega} f \phi \, dx \quad \forall \phi \in V_0,$$

$$(5.35b) \quad \int_{\Omega} \Delta(\psi^1 - \psi^0) \Delta \dot{\psi}(0) \, dx + (\lambda^1 - \lambda^0) \dot{\lambda}(0) = \Delta s \quad \text{if } n=0,$$

$$(5.35c) \quad \int_{\Omega} \Delta(\psi^{n+1} - \psi^n) \Delta \left( \frac{\psi^n - \psi^{n-1}}{\Delta s} \right) dx + (\lambda^{n+1} - \lambda^n) \left( \frac{\lambda^n - \lambda^{n-1}}{\Delta s} \right) = \Delta s \quad \text{if } n \geq 1.$$

With  $V_0$  equipped with the inner product

$$\{v, w\} \rightarrow \int_{\Omega} \Delta v \Delta w \, dx$$

a convenient nonlinear least squares formulation of (5.35) is then:

$$(5.36) \quad \text{Find } \{\psi^{n+1}, \lambda^{n+1}\} \in V_g \times \mathbb{R} \text{ such that} \\ J_{n+1}(\psi^{n+1}, \lambda^{n+1}) \leq J_{n+1}(\chi, \mu) \quad \forall \{\chi, \mu\} \in V_g \times \mathbb{R}.$$

Here the functional  $J_{n+1}(\cdot, \cdot)$  is defined by

$$(5.37) \quad J_{n+1}(\chi, \mu) = \frac{1}{2} \int_{\Omega} |\Delta \tilde{\chi}|^2 \, dx + \frac{1}{2} |\tilde{\mu}|^2;$$

where  $\tilde{\chi}$  and  $\tilde{\mu}$  are nonlinear functions of  $\{\chi, \mu\}$ , obtained as the solutions of the *linear* problems, respectively:

$$(5.38) \quad \tilde{\chi} \in V_0, \quad \forall \phi \in V_0 \text{ we have}$$

$$\int_{\Omega} \Delta \tilde{\chi} \Delta \phi \, dx = \int_{\Omega} \Delta \chi \Delta \phi \, dx - \mu \int_{\Omega} \Delta \chi \left( \frac{\partial \chi}{\partial x_1} \frac{\partial \phi}{\partial x_2} - \frac{\partial \chi}{\partial x_2} \frac{\partial \phi}{\partial x_1} \right) dx - \mu \int_{\Omega} f \phi \, dx,$$

$$(5.39) \quad \tilde{\mu} = \int_{\Omega} \Delta(\chi - \psi^n) \Delta \left( \frac{\psi^n - \psi^{n-1}}{\Delta s} \right) dx + (\mu - \lambda^n) \left( \frac{\lambda^n - \lambda^{n-1}}{\Delta s} \right) - \Delta s.$$

Problem (5.38) is a biharmonic problem.

The least squares problem (5.36) can be solved by the conjugate gradient algorithm described in § 2.3; since the nonlinearity in (5.35a) is *quadratic* one should verify that each iteration requires the solution of “only” 3 *linear biharmonic problems* of the following type:

$$(5.40) \quad \Delta^2 w = f \quad \text{in } \Omega$$

$$w = g_1 \quad \text{on } \Gamma, \quad \frac{\partial w}{\partial n} = g_2 \quad \text{on } \Gamma.$$

*Finite element* solvers for (5.40) will be discussed in § 5.5; they are founded on the mixed variational formulation (5.17).

More details about the conjugate gradient solution of (5.36) are given in [43].

**5.4.3. Solution of (5.10)–(5.12) via the mixed variational formulation (5.17).** Using  $\lambda = 1/\nu$  as parameter, the nonlinear mixed variational problem (5.17) becomes

$$(5.41) \quad \text{Find } \{\omega, \psi\} \in W_g \text{ such that } \forall \{\theta, \phi\} \in W_0 \text{ we have}$$

$$\int_{\Omega} \omega \theta \, dx + \lambda \int_{\Omega} \omega \left( \frac{\partial \psi}{\partial x_1} \frac{\partial \phi}{\partial x_2} - \frac{\partial \psi}{\partial x_2} \frac{\partial \phi}{\partial x_1} \right) dx = \lambda \int_{\Omega} f \phi \, dx,$$

with  $W_0$  and  $W_g$  still defined by (5.20), (5.21), respectively.

*Description of the continuation procedure.* We clearly have from § 5.4.2.

(a) *Initialization.*

$$(5.42) \quad \text{Take } \lambda^0 = 0;$$

then  $\{w^0, \psi^0\}$  is the unique solution of the following linear mixed variational problem (equivalent to (5.27)):

$$(5.43) \quad \text{Find } \{\omega^0, \psi^0\} \in W_g \text{ such that}$$

$$\int_{\Omega} \omega^0 \theta \, dx = 0 \quad \forall \{\theta, \phi\} \in W_0.$$

We take then  $\{\{\omega^0, \psi^0\}, 0\}$  as the origin of the arc of solutions passing through it and define the arc length  $s$  by

$$(5.44) \quad (\delta s)^2 = \int_{\Omega} (\delta \omega)^2 \, dx + (\delta \lambda)^2.$$

By differentiation of (5.41) with respect to  $s$ , we obtain at  $s=0$

$$(5.45) \quad \int_{\Omega} \dot{\omega}(0)\theta \, dx = \dot{\lambda}(0) \int_{\Omega} \omega^0 \left( \frac{\partial \psi^0}{\partial x_2} \frac{\partial \phi}{\partial x_1} - \frac{\partial \psi^0}{\partial x_1} \frac{\partial \phi}{\partial x_2} \right) dx \\ + \dot{\lambda}(0) \int_{\Omega} f\phi \, dx \quad \forall \{\theta, \phi\} \in W_0, \quad \{\dot{\omega}(0), \dot{\psi}(0)\} \in W_0.$$

Since we have

$$\int_{\Omega} |\dot{\omega}(0)|^2 \, dx + \dot{\lambda}^2(0) = 1,$$

we obtain from (5.45) that

$$\dot{\lambda}(0) = \left( 1 + \int_{\Omega} |\dot{\omega}|^2 \, dx \right)^{-1/2}, \quad \{\dot{\omega}(0), \dot{\psi}(0)\} = \dot{\lambda}(0) \{\hat{\omega}, \hat{\psi}\},$$

where  $\{\hat{\omega}, \hat{\psi}\}$  is the solution of the linear mixed variational problem

$$(5.46) \quad \{\hat{\omega}, \hat{\psi}\} \in W_0, \\ \int_{\Omega} \hat{\omega}\theta \, dx = \int_{\Omega} \omega^0 \left( \frac{\partial \psi^0}{\partial x_2} \frac{\partial \phi}{\partial x_1} - \frac{\partial \psi^0}{\partial x_1} \frac{\partial \phi}{\partial x_2} \right) dx + \int_{\Omega} f\phi \, dx \quad \forall \{\theta, \phi\} \in W_0.$$

(b) *Continuation.* With  $\Delta s (>0)$  an elementary arc length we define for  $n \geq 0$  an approximation  $\{\{\omega^{n+1}, \psi^{n+1}\}, \lambda^{n+1}\} \in W_g \times \mathbb{R}$  of  $\{\{\omega((n+1)\Delta s), \psi((n+1)\Delta s)\}, \lambda((n+1)\Delta s)\}$  as the solution of the following nonlinear mixed variational system:

Find  $\{\{\omega^{n+1}, \psi^{n+1}\}, \lambda^{n+1}\} \in W_g \times \mathbb{R}$  such that

$$(5.47a) \quad \int_{\Omega} \omega^{n+1}\theta \, dx = \lambda^{n+1} \int_{\Omega} \omega^{n+1} \left( \frac{\partial \psi^{n+1}}{\partial x_2} \frac{\partial \phi}{\partial x_1} - \frac{\partial \psi^{n+1}}{\partial x_1} \frac{\partial \phi}{\partial x_2} \right) dx \\ + \lambda^{n+1} \int_{\Omega} f\phi \, dx \quad \forall \{\theta, \phi\} \in W_0,$$

$$(5.47b) \quad \int_{\Omega} (\omega^1 - \omega^0)\dot{\omega}(0) \, dx + (\lambda^1 - \lambda^0)\dot{\lambda}(0) = \Delta s \quad \text{if } n=0,$$

$$(5.47c) \quad \int_{\Omega} (\omega^{n+1} - \omega^n) \left( \frac{\omega^n - \omega^{n-1}}{\Delta s} \right) dx + (\lambda^{n+1} - \lambda^n) \left( \frac{\lambda^n - \lambda^{n-1}}{\Delta s} \right) = \Delta s \quad \text{if } n \geq 1.$$

The space  $W_0$  can be equipped with the inner product

$$\{ \{\theta_1, \phi_1\}, \{\theta_2, \phi_2\} \} \rightarrow \int_{\Omega} \theta_1 \theta_2 \, dx.$$

A convenient least squares formulation of (5.47) is then

$$(5.48) \quad \text{Find } \{\{\omega^{n+1}, \psi^{n+1}\}, \lambda^{n+1}\} \in W_g \times \mathbb{R} \text{ such that} \\ j_{n+1}(\{\omega^{n+1}, \psi^{n+1}\}, \lambda^{n+1}) \leq j_{n+1}(\{\eta, \chi\}, \mu) \quad \forall \{\{\eta, \chi\}, \mu\} \in W_g \times \mathbb{R},$$

where in (5.48) the functional  $j_{n+1}(\cdot, \cdot)$  is defined by

$$(5.49) \quad j_{n+1}(\{\eta, \chi\}, \mu) = \frac{1}{2} \int_{\Omega} |\tilde{\eta}|^2 \, dx + \frac{1}{2} |\tilde{\mu}|^2.$$

In (5.49),  $\{\tilde{\eta}, \tilde{\chi}\}$  and  $\tilde{\mu}$  are nonlinear functions of  $\eta, \chi, \mu$  obtained as the solutions of the linear problems

$$\begin{aligned} & \{\tilde{\eta}, \tilde{\chi}\} \in W_0, \\ (5.50) \quad & \forall \{\theta, \phi\} \in W_0 \text{ we have} \\ & \int_{\Omega} \tilde{\eta} \theta \, dx = \int_{\Omega} \eta \theta \, dx - \mu \int_{\Omega} \eta \left( \frac{\partial \chi}{\partial x_2} \frac{\partial \phi}{\partial x_1} - \frac{\partial \chi}{\partial x_1} \frac{\partial \phi}{\partial x_2} \right) \, dx - \mu \int_{\Omega} f \phi \, dx, \\ (5.51) \quad & \tilde{\mu} = \int_{\Omega} (\eta - \omega^n) \left( \frac{\omega^n - \omega^{n-1}}{\Delta s} \right) \, dx + (\mu - \lambda^n) \left( \frac{\lambda^n - \lambda^{n-1}}{\Delta s} \right) - \Delta s, \end{aligned}$$

respectively; problem (5.50) is equivalent to the biharmonic problem (5.38).

The conjugate gradient algorithm (2.37)–(2.46) can be applied, again, in this context, each iteration requiring, as in § 5.4.2, the solution of 3 linear biharmonic problems (see [43] for more details).

*Remark 5.1.* The main motivation of the mixed variational formulation discussed in §§ 5.3 and 5.4.3 is that it provides a convenient framework for the approximation of linear and nonlinear biharmonic problems, by very simple finite element methods like those discussed in the following section. Another application is discussed in [17]; it concerns the Von Karman equations for nonlinear plates.

**5.5. Finite element approximation.**

**5.5.1. Triangulation of  $\Omega$ . Fundamental discrete spaces.** We suppose for simplicity that  $\Omega$  is a polygonal domain of  $\mathbb{R}^2$ . With  $\mathcal{T}_h$  a triangulation of  $\Omega$  obeying the conditions given in § 3.2.1 we define the following finite-dimensional functional spaces:

$$\begin{aligned} (5.52) \quad & H_h^1 = \{v_h \mid v_h \in C^0(\bar{\Omega}), v_h|_T \in P_k, \forall T \in \mathcal{T}_h\}, \\ (5.53) \quad & H_{0h}^1 = H_h^1 \cap H_0^1(\Omega) \quad (= \{v_h \mid v_h \in H_h^1, v_h|_{\Gamma} = 0\}) \end{aligned}$$

with  $P_k \equiv$  space of polynomials in  $x_1, x_2$  of degree  $\leq k$ ;  $H_h^1$  and  $H_{0h}^1$  approximate  $H^1(\Omega)$  and  $H_0^1(\Omega)$ , respectively.

We approximate then the spaces  $W_0$  and  $W_g$  (defined by (5.20) and (5.21), respectively) by

$$\begin{aligned} (5.54) \quad & W_{0h} = \left\{ \{\theta_h, \phi_h\} \in H_h^1 \times H_{0h}^1, \int_{\Omega} \nabla \phi_h \cdot \nabla q_h \, dx = \int_{\Omega} \theta_h q_h \, dx, \forall q_h \in H_h^1 \right\}, \\ (5.55) \quad & W_{gh} = \left\{ \{\theta_h, \phi_h\} \in H_h^1 \times H_n^1, \phi_h = g_{1h} \text{ on } \Gamma, \right. \\ & \left. \int_{\Omega} \nabla \phi_h \cdot \nabla q_h \, dx = \int_{\Omega} \theta_h q_h \, dx + \int_{\Gamma} g_{2h} q_h \, d\Gamma, \forall q_h \in H_h^1 \right\}. \end{aligned}$$

Here  $g_{1h}$  and  $g_{2h}$  are convenient approximations to  $g_1$  and  $g_2$ , respectively. We observe that  $W_{0h} \not\subset W_0$ ; similarly  $W_{gh} \not\subset W_g$ , even in the simple case where  $g_{1h} = g_1, g_{2h} = g_2$ .

**5.5.2. Approximation of the Navier–Stokes equations via the  $\{\omega, \psi\}$  formulation.** Using  $\lambda = 1/\nu$  as parameter, a mixed variational formulation of the Navier–Stokes equations was given in § 5.4.3 by (5.41). We approximate (5.41) by:

$$(5.56) \quad \text{Find } \{\omega_h, \psi_h\} \in W_{gh} \text{ such that } \forall \{\theta_h, \phi_h\} \in W_{0h} \text{ we have}$$

$$\int_{\Omega} \omega_h \theta_h \, dx + \lambda \int_{\Omega} \omega_h \left( \frac{\partial \psi_h}{\partial x_1} \frac{\partial \phi_h}{\partial x_2} - \frac{\partial \psi_h}{\partial x_2} \frac{\partial \phi_h}{\partial x_1} \right) \, dx = \lambda \int_{\Omega} f_h \phi_h \, dx,$$

with  $f_h$  a convenient approximation to  $f$ . We refer to Girault–Raviart [22] for the convergence properties of  $\{\omega_h, \psi_h\}$  as  $h \rightarrow 0$ .

Concentrating on the numerical solution of problem (5.56) by *continuation least squares methods* we easily adapt the algorithms of §§ 2.3 and 5.4.3 to the solution of the approximate problem (5.56) (see Reinhart [3] for more details on the solution of (5.56) by the methods of the present paper).

In fact applying the discrete analogues of the methods described in § 5.4.3 to the solution of (5.56) requires an efficient solver for the various *discrete linear biharmonic problems* coming from the mixed finite element approximation. Such a solver is particularly required by the conjugate gradient algorithm in solving the least squares problem encountered at each step of the continuation process (we have to solve 3 linear biharmonic problems at each iteration).

### 5.5.3. On the solution of the discrete linear biharmonic problems.

**5.5.3.1. Generalities. Synopsis.** A careful examination of the algorithms discussed in § 5.4.3 shows that the *discrete linear biharmonic problems* to be solved are in fact mixed finite element approximations of biharmonic problems of the following class:

$$(5.57) \quad \begin{aligned} \Delta^2 \psi &= f_0 - \frac{\partial f_1}{\partial x_1} - \frac{\partial f_2}{\partial x_2} - \Delta f_3 \quad \text{in } \Omega, \\ \psi|_{\Gamma} &= g_1, \quad \frac{\partial \psi}{\partial n} \Big|_{\Gamma} = g_2. \end{aligned}$$

Here  $f_i \in L^2(\Omega)$ ,  $\forall i = 0, 1, 2, 3$ , and the derivatives occurring in (5.57) have to be understood in the sense of distributions. Assuming that  $g_1, g_2$  are sufficiently smooth, problem (5.57) has a unique solution in  $V_g$  (see § 5.3 for the definition of  $V_g$  and  $V_0$ ); this solution  $\psi$  is also the unique solution of the following variational problem:

$$(5.58) \quad \begin{aligned} &\text{Find } \psi \in V_g \text{ such that } \forall \phi \in V_0 \\ &\int_{\Omega} \Delta \psi \Delta \phi \, dx = \int_{\Omega} \left( f_0 \phi + f_1 \frac{\partial \phi}{\partial x_1} + f_2 \frac{\partial \phi}{\partial x_2} - f_3 \Delta \phi \right) dx. \end{aligned}$$

An equivalent *mixed variational formulation* of (5.58) is given by:

$$(5.59) \quad \begin{aligned} &\text{Find } \{\omega, \psi\} \in W_g \text{ such that } \forall \{\theta, \phi\} \in W_0 \\ &\int_{\Omega} \omega \theta \, dx = \int_{\Omega} \left( f_0 \phi + f_1 \frac{\partial \phi}{\partial x_1} + f_2 \frac{\partial \phi}{\partial x_2} + f_3 \theta \right) dx, \end{aligned}$$

where  $W_0$  and  $W_g$  are defined by (5.20) and (5.21), respectively.

Starting from the mixed formulation (5.59) we shall discuss in the following sections the finite element approximation of (5.59) and solution methods for the approximate problems.

**5.5.3.2. Finite element approximation of (5.59).** Following Ciarlet–Raviart [26] and Glowinski–Pironneau [25] we approximate (5.59) by

$$(5.60) \quad \begin{aligned} &\text{Find } \{\omega_h, \psi_h\} \in W_{gh} \text{ such that } \forall \{\theta_h, \phi_h\} \in W_{0h} \\ &\int_{\Omega} \omega_h \theta_h \, dx = \int_{\Omega} \left( f_{0h} \phi_h + f_{1h} \frac{\partial \phi_h}{\partial x_1} + f_{2h} \frac{\partial \phi_h}{\partial x_2} + f_{3h} \theta_h \right) dx, \end{aligned}$$

where  $W_{0h}$  and  $W_{gh}$  are still defined by (5.54) and (5.55), respectively, and where  $f_{ih}$  is, for  $i = 0, 1, 2, 3$ , a convenient approximation to  $f_i$ .

It is quite easy to prove that (5.60) has a unique solution; concerning the convergence of  $\{\omega_h, \psi_h\}$  to  $\{-\Delta\psi, \psi\}$  as  $h \rightarrow 0$ , it follows from Ciarlet-Raviart [26], Scholz [27] that

$$(5.61) \quad \lim_{h \rightarrow 0} \|-\Delta\psi - \omega_h\|_{L^2(\Omega)} = 0, \quad \lim_{h \rightarrow 0} \|\psi - \psi_h\|_{H^1(\Omega)} = 0$$

for all  $k \geq 1$  (in the definition of  $H_h^1$ ; cf. (5.52)). Actually the convergence result (5.61) supposes that some mild assumptions on the angles are satisfied as  $h \rightarrow 0$  (see the two above references for more details).

**5.5.3.3. Decomposition properties of the approximate problem (5.60).** We here follow and extend on some points in Glowinski-Pironneau [25].

A direct solution of (5.60) is a nontrivial task; however taking into account the very special structure of (5.60) we shall be able, *via a decomposition principle*, to reduce its solution to the solution of a family of *discrete Poisson problems which are much easier to solve*.

The starting point of our discussion is the fact that the pair  $\{\omega_h, \psi_h\}$ , solving (5.60), is characterized by the existence of  $p_h$  such that

$$(5.62a) \quad p_h \in H_h^1, \\ \int_{\Omega} \nabla p_h \cdot \nabla \phi_h \, dx = \int_{\Omega} \left( f_{0h} \phi_h + f_{1h} \frac{\partial \phi_h}{\partial x_1} + f_{2h} \frac{\partial \phi_h}{\partial x_2} \right) dx \quad \forall \phi_h \in H_{0h}^1,$$

$$(5.62b) \quad \omega_h \in H_h^1, \\ \int_{\Omega} \omega_h \theta_h \, dx = \int_{\Omega} (f_{3h} + p_h) \theta_h \, dx \quad \forall \theta_h \in H_h^1,$$

$$(5.62c) \quad \psi_h \in H_h^1, \psi_h = g_{1h} \quad \text{on } \Gamma, \\ \int_{\Omega} \nabla \psi_h \cdot \nabla q_h \, dx = \int_{\Omega} \omega_h q_h \, dx + \int_{\Gamma} g_{2h} q_h \, d\Gamma \quad \forall q_h \in H_h^1.$$

To prove the characterization (5.62) we observe that (5.60) is equivalent to the minimization problem:

$$(5.63) \quad \text{Find } \{\omega_h, \psi_h\} \in W_{gh} \text{ such that} \\ j_h(\omega_h, \psi_h) \leq j_h(\theta_h, \phi_h) \quad \forall \{\theta_h, \phi_h\} \in W_{gh},$$

where

$$(5.64) \quad j_h(\theta_h, \phi_h) = \frac{1}{2} \int_{\Omega} \theta_h^2 \, dx - \int_{\Omega} \left( f_{0h} \phi_h + f_{1h} \frac{\partial \phi_h}{\partial x_1} + f_{2h} \frac{\partial \phi_h}{\partial x_2} + f_{3h} \theta_h \right) dx.$$

Hence  $p_h$  appears as a *Lagrange multiplier* for the *linear equality constraints* satisfied by  $\{\omega_h, \psi_h\}$  in (5.62c) (and in the definition of  $W_{gh}$ ; see (5.55)).

To go further into the decomposition properties we introduce a space  $\mathcal{M}_h$  with the following properties:

$$(5.65) \quad \mathcal{M}_h \text{ is a complementary space (not precisely defined for the moment) of } H_{0h}^1 \text{ in } H_h^1, \text{ i.e. } \mathcal{M}_h \subset H_h^1 \text{ and } H_{0h}^1 \oplus \mathcal{M}_h = H_h^1.$$

It follows from (5.65) that the bilinear form  $\mathcal{M}_h \times \mathcal{M}_h \rightarrow \mathbb{R}$  defined by

$$\{\lambda_h, \mu_h\} \rightarrow \int_{\Gamma} \lambda_h \mu_h \, d\Gamma$$

is a *scalar product* over  $\mathcal{M}_h$ . The key step is in fact to introduce a bilinear form  $a_h: \mathcal{M}_h \times \mathcal{M}_h \rightarrow \mathbb{R}$ , defined as follows:

Let  $\lambda_h \in \mathcal{M}_h$  and let  $p_h$ , respectively  $\psi_h$ , be the solutions of the following approximate problems:

$$(5.66a) \quad \int_{\Omega} \nabla p_h \cdot \nabla \phi_h \, dx = 0 \quad \forall \phi_h \in H_{0h}^1, \quad p_h \in H_h^1, \quad p_h - \lambda_h \in H_{0h}^1,$$

$$(5.66b) \quad \int_{\Omega} \nabla \psi_h \cdot \nabla \phi_h \, dx = \int_{\Omega} p_h \phi_h \, dx \quad \forall \phi_h \in H_{0h}^1, \quad \psi_h \in H_{0h}^1.$$

Then we define the bilinear form  $a_h(\cdot, \cdot)$  by

$$(5.66c) \quad a_h(\lambda_h, \mu_h) = \int_{\Omega} p_h \mu_h \, dx - \int_{\Omega} \nabla \psi_h \cdot \nabla \mu_h \, dx \quad \forall \mu_h \in \mathcal{M}_h.$$

It then follows from [25, § 3.5] that  $a_h(\cdot, \cdot)$  is *symmetric* and *positive definite*.

*Application to the decomposition of the approximate problem* (5.60). Let  $\{\omega_h, \psi_h\}$  be the solution of (5.60) and let  $\lambda_h$  be the component in  $\mathcal{M}_h$  of the function  $p_h$  occurring in the characterization (5.62). Let  $\bar{p}_h, \bar{\psi}_h$  be the solutions of

$$(5.67) \quad \int_{\Omega} \nabla \bar{p}_h \cdot \nabla \phi_h \, dx = 0 \quad \forall \phi_h \in H_{0h}^1, \quad \bar{p}_h - \lambda_h \in H_{0h}^1,$$

$$(5.68) \quad \int_{\Omega} \nabla \bar{\psi}_h \cdot \nabla \phi_h \, dx = \int_{\Omega} \bar{p}_h \phi_h \, dx \quad \forall \phi_h \in H_{0h}^1, \quad \bar{\psi}_h \in H_{0h}^1.$$

Let  $p_{0h}$  and  $\psi_{0h}$  be the solutions of

$$(5.69) \quad \int_{\Omega} \nabla p_{0h} \cdot \nabla \phi_h \, dx = \int_{\Omega} \left( f_{0h} \phi_h + f_{1h} \frac{\partial \phi_h}{\partial x_1} + f_{2h} \frac{\partial \phi_h}{\partial x_2} \right) dx \quad \forall \phi_h \in H_{0h}^1, \quad p_{0h} \in H_{0h}^1,$$

$$(5.70) \quad \int_{\Omega} \nabla \psi_{0h} \cdot \nabla \phi_h \, dx = \int_{\Omega} (p_{0h} + f_{3h}) \phi_h \, dx \quad \forall \phi_h \in H_{0h}^1, \quad \psi_h \in H_h^1, \quad \psi_{0h} = g_{1h} \quad \text{on } \Gamma.$$

We clearly have  $p_h = \bar{p}_h + p_{0h}$  and  $\psi_h = \bar{\psi}_h + \psi_{0h}$ .

We shall now show that  $\lambda_h$  is the solution of a variational problem in  $\mathcal{M}_h$ .

**THEOREM 5.1.** *Let  $\{\omega_h, \psi_h\}$  be the solution of (5.60) and let  $\lambda_h$  be the component in  $\mathcal{M}_h$  of  $p_h$  defined from  $\{\omega_h, \psi_h\}$  by (5.62). Then  $\lambda_h$  is the unique solution of the linear variational problem*

$$(5.71) \quad a_h(\lambda_h, \mu_h) = \int_{\Omega} \nabla \psi_{0h} \cdot \nabla \mu_h \, dx - \int_{\Omega} (p_{0h} - f_{3h}) \mu_h \, dx - \int_{\Gamma} g_{2h} \mu_h \, d\Gamma \quad \forall \mu_h \in \mathcal{M}_h, \quad \lambda_h \in \mathcal{M}_h$$

which is equivalent to a linear system with a positive definite matrix.

*Proof.* We have from (5.66)–(5.68) that

$$\begin{aligned}
 a_h(\lambda_h, \mu_h) &= \int_{\Omega} \bar{p}_h \mu_h \, dx - \int_{\Omega} \nabla \bar{\psi}_h \cdot \nabla \mu_h \, dx \\
 &= \int_{\Omega} (p_h - p_{0h}) \mu_h \, dx - \int_{\Omega} \nabla (\psi_h - \psi_{0h}) \cdot \nabla \mu_h \, dx \\
 &= \int_{\Omega} \nabla \psi_{0h} \cdot \nabla \mu_h \, dx - \int_{\Omega} (p_{0h} + f_{3h}) \mu_h \, dx \\
 &\quad - \left( \int_{\Omega} \nabla \psi_h \cdot \nabla \mu_h \, dx - \int_{\Omega} (p_h + f_{3h}) \mu_h \, dx \right) \quad \forall \mu_h \in \mathcal{M}_h.
 \end{aligned}
 \tag{5.72}$$

But from (5.62b, c) we have

$$\begin{aligned}
 \int_{\Omega} \nabla \psi_h \cdot \nabla \mu_h \, dx - \int_{\Omega} (p_h + f_{3h}) \mu_h \, dx &= \int_{\Omega} \nabla \psi_h \cdot \nabla \mu_h \, dx - \int_{\Omega} \omega_h \mu_h \, dx \\
 &= \int_{\Omega} g_{2h} \mu_h \, d\Gamma \quad \forall \mu_h \in \mathcal{M}_h,
 \end{aligned}$$

which, together with (5.72), proves (5.71). The uniqueness is obvious since  $a_h(\cdot, \cdot)$  is positive definite. The equivalence with a positive definite linear system is a classical result in the approximation of linear variational problems.  $\square$

*Remark 5.2.* To compute the right-hand side of (5.71) it is necessary to solve the two approximate Dirichlet problems (5.69) and (5.70). Similarly if  $\lambda_h$  is known, to compute  $p_h$ ,  $\omega_h$  and  $\psi_h$  it is necessary to solve

$$\begin{aligned}
 p_h &\in H_h^1, p_h - \lambda_h \in H_{0h}^1, \\
 \int_{\Omega} \nabla p_h \cdot \nabla \phi_h \, dx &= \int_{\Omega} \left( f_{0h} \phi_h + f_{1h} \frac{\partial \phi_h}{\partial x_1} + f_{2h} \frac{\partial \phi_h}{\partial x_2} \right) dx \quad \forall \phi_h \in H_{0h}^1,
 \end{aligned}
 \tag{5.73}$$

$$\begin{aligned}
 \omega_h &\in H_h^1, \\
 \int_{\Omega} \omega_h \theta_h \, dx &= \int_{\Omega} (f_{3h} + p_h) \theta_h \, dx \quad \forall \theta_h \in H_h^1,
 \end{aligned}
 \tag{5.74}$$

$$\begin{aligned}
 \psi_h &\in H_h^1, \psi_h = g_{1h} \text{ on } \Gamma, \\
 \int_{\Omega} \nabla \psi_h \cdot \nabla \phi_h \, dx &= \int_{\Omega} (p_h + f_{3h}) \phi_h \, dx \quad \forall \phi_h \in H_{0h}^1,
 \end{aligned}
 \tag{5.75}$$

i.e. two discrete Dirichlet problems, (5.73) and (5.75), and (5.74) which is a much simpler linear problem ( $\omega_h$  is in fact the  $L^2$ -projection on  $H_h^1$  of the function  $p_h + f_{3h}$ ).

*Recapitulation.* It has been shown that solving the discrete biharmonic problem (5.60) is equivalent to solving (5.69), (5.70), (5.71), (5.73), (5.74), (5.75) *sequentially*. Problems (5.69), (5.70), (5.73), (5.75) are discrete Dirichlet problems, for the operator  $-\Delta$ , for which very efficient direct or iterative solvers exist. The variational problem (5.74) is even simpler to solve, since the matrix of the equivalent linear system is very sparse, has a condition number  $O(1)$  and is in fact an approximation to the identity operator. Finally the only nonstandard step is the solution of the variational problem (5.71) which is discussed in the following § 5.5.3.4.

**5.5.3.4. Solution of problem (5.71).** Several methods for the solution of (5.71) have been discussed in [25, §§ 4 and 5]. Let us mention among them a *conjugate gradient method* which yields a solution algorithm for the discrete biharmonic problem (5.60);



the cost per iteration is essentially the solution of *two* discrete Dirichlet problems for the operator  $-\Delta$ ; numerical experiments show a convergence in  $O(N_h^{1/2})$  iterations, where  $N_h = \dim \mathcal{M}_h$ . We find also in [25, § 4] a detailed analysis of a direct method for solving (5.71) requiring the construction of the symmetric, positive definite (and full) matrix  $A_h$  of the linear system equivalent to (5.71). In fact one does not construct  $A_h$ , but (using the *Cholesky factorization* method) a lower triangular-regular matrix  $L_h$  such that  $A_h = L_h L_h^T$ ; since the construction of  $L_h$  requires (cf. [25, § 4]) the solution of  $2N_h$  discrete Dirichlet problems it seems preferable to use the conjugate gradient algorithm. However in practice we prefer direct solvers for the following reasons:

(i) Since the  $2N_h$  discrete Dirichlet problems mentioned above have all the same matrix which is symmetric and positive definite, a Cholesky factorization done once and for all will result in an important saving of computational time.

(ii) If a large number of linear discrete biharmonic problems have to be solved—as in time dependent problems or during an iterative process like those discussed in this paper—the solution method of (5.60), founded on the construction of  $L_h$  offers (from our numerical experiments) a more economical strategy than the conjugate gradient algorithms discussed in [25, § 5].

The above comments justify the choice of the direct solution of (5.71) for the numerical experiments described in § 5.6.

We have given in [43] the description of a *new conjugate gradient algorithm with scaling* (i.e. preconditioning). If the speed of convergence is measured in number of iterations, the new algorithm is faster than those discussed in [25, § 5]. However the new algorithm requires the solution of *three* discrete Dirichlet problems instead of *two*, for each iteration, as in the algorithm (5.26)–(5.33) [25, pp. 197–198].

*Remark 5.3 (On the choice of  $\mathcal{M}_h$ ).* Suppose that  $H_h^1$  is composed of ordinary Lagrangian finite elements of order  $k$  ( $k = 1, 2$  in most applications). It follows then from [25] (for which we refer for more details) that the best choice for  $\mathcal{M}_h$  is given by

$$(5.76) \quad \mathcal{M}_h = \{ \mu_h \mid \mu_h \in H_h^1, \mu_h|_T = 0 \quad \forall T \in \mathcal{T}_h \text{ such that } \partial T \cap \Gamma = \emptyset \}.$$

With such a choice the elements of  $\mathcal{M}_h$  are completely determined by the values attained at those nodes of  $\mathcal{T}_h$  belonging to  $\Gamma$ . Thus we should take as basis functions for  $\mathcal{M}_h$  those basis functions of  $H_h^1$  associated with the boundary nodes (again, see [25] for more details).

**5.6. Numerical experiments.**

**5.6.1. Formulation of the test problem.** With  $\Omega = ]0, 1[ \times ]0, 1[$  we consider the following Navier–Stokes test problem:

$$(5.77) \quad \begin{aligned} -\nu \Delta \mathbf{u} + (\mathbf{u} \cdot \nabla) \mathbf{u} + \nabla p &= 0 \quad \text{in } \Omega, \\ \nabla \cdot \mathbf{u} &= 0 \quad \text{in } \Omega, \\ \mathbf{u}(x_1, x_2)|_\Gamma &= \begin{cases} \{1, 0\} & \text{if } x_2 = 1, \\ \{0, 0\} & \text{if } 0 \leq x_2 < 1. \end{cases} \end{aligned}$$

Hence problem (5.77) is the classical *driven cavity* problem. The corresponding  $\{\omega, \psi\}$  formulation is

$$(5.78) \quad \begin{aligned} \nu \Delta \omega + \left( \frac{\partial \psi}{\partial x_2} \frac{\partial \omega}{\partial x_1} - \frac{\partial \psi}{\partial x_1} \frac{\partial \omega}{\partial x_2} \right) &= 0 \quad \text{in } \Omega, \\ -\Delta \psi &= \omega \quad \text{in } \Omega, \\ \psi &= 0 \text{ on } \Gamma; \frac{\partial \psi}{\partial n}(x_1, x_2)|_\Gamma = \begin{cases} 1 & \text{if } x_2 = 1, \\ 0 & \text{if } 0 \leq x_2 < 1. \end{cases} \end{aligned}$$

**5.6.2. Triangulation of  $\Omega$ .** The triangulation  $\mathcal{T}_h$  used to approximate (5.77), (5.78) by the methods of § 5.5, is shown on Fig. 5.1. It contains 800 triangles and since *piecewise quadratic* elements are used (i.e.  $k=2$  in (5.52)), it corresponds to 160 boundary nodes and 1581 interior nodes (vertices and midpoints); we have therefore a nonlinear system of about 3300 unknowns to solve after discretization.

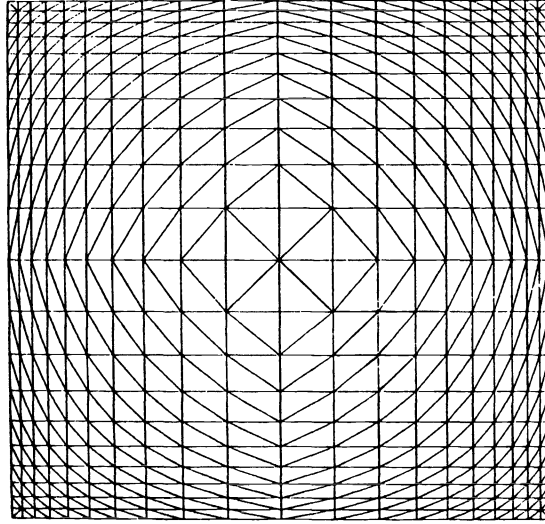


FIG. 5.1

**5.6.3. Numerical results—further comments.** The numerical procedure described in § 5.4.3 has been applied to the solution of the approximate problem (5.56) associated with (5.78) (using  $\lambda = 1/\nu = \text{Re}$ ). The computations have been done on a CRAY-1 computer, with special vectorized subroutines (in particular every subroutine concerning profile matrices (product, Cholesky factorization, resolution of triangular linear systems) has been vectorized).

We have used  $\Delta s = 100$  for  $0 \leq \lambda \leq 1400$ ,  $\Delta s = 200$  for  $1400 \leq \lambda \leq 2600$ ,  $\Delta s = 400$  for  $\lambda = 3000$ .

The conjugate gradient iterations were stopped as soon as the least squares cost functional was less than  $10^{-5}$ . The computations have been done with double precision variables.

Figures 5.2, 5.3, 5.4, 5.5, 5.6 show the variations of the least squares cost functional as a function of the number of conjugate gradient iterations, for  $\lambda = 100, 400, 1600, 2000, 3000$ , respectively; as expected the number of iterations necessary to obtain the convergence is an increasing function of  $\lambda$  ( $= \text{Re}$ ).

For  $\text{Re} = 3000$ , the average CPU for one iteration of conjugate gradient is about 0.9 second on the CRAY-1 computer.

The stream lines for  $\text{Re} = 100, 400, 1600, 2000, 3000$  are shown in Figs. 5.7–5.11 respectively. The values of the stream function along the lines are:

$$\begin{aligned} \psi = & -0.12, -0.1, -0.08, -0.06, -0.04, -0.02, 0.0 \\ & = 0.0025, 0.001, 0.0005, 0.0001, 0.00005. \end{aligned}$$

Even for small values of the Reynolds number, there appear two secondary vortices in the lower upstream and downstream corners. These vortices grow larger as the Reynolds number increases. For values of  $\text{Re}$  beyond 1500, a third secondary vortex

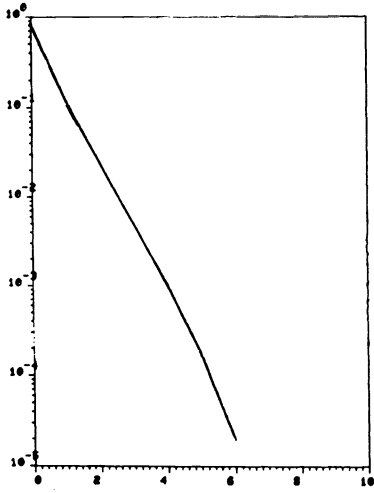


FIG. 5.2.  $Re = 100$ .

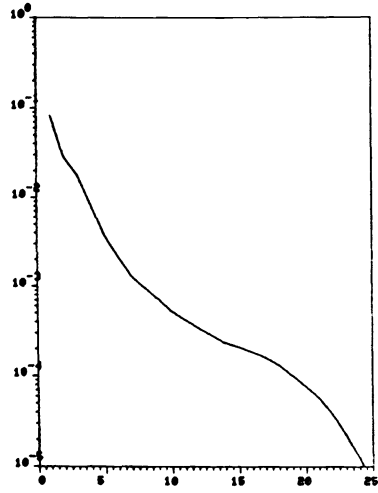


FIG. 5.3.  $Re = 400$ .

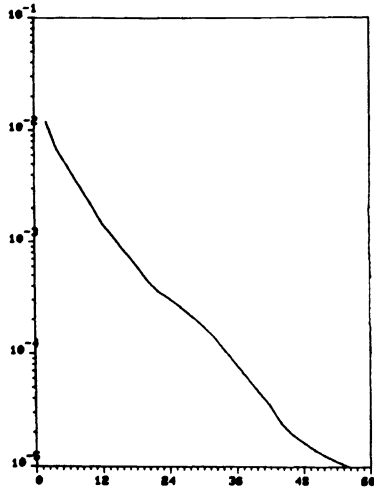


FIG. 5.4.  $Re = 1600$ .

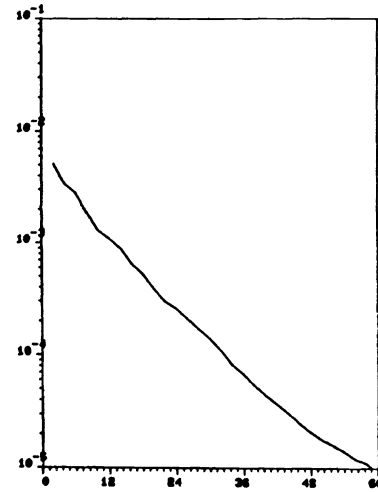


FIG. 5.5.  $Re = 2000$ .

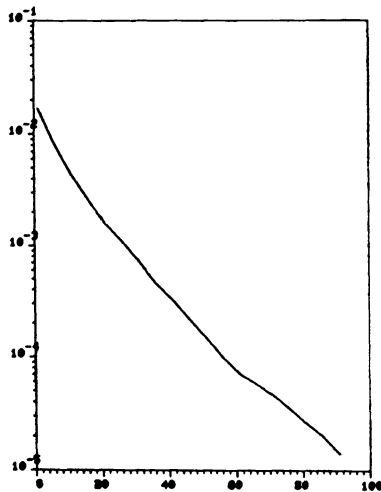


FIG. 5.6.  $Re = 3000$ .

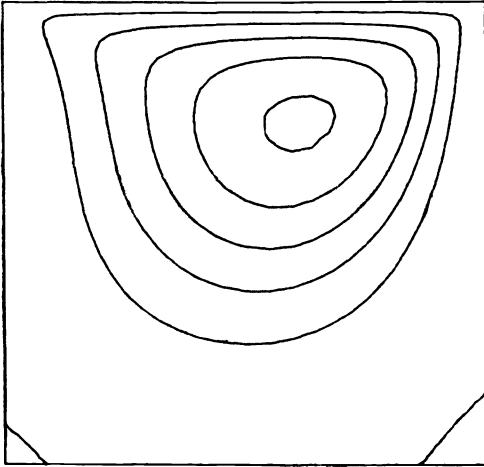


FIG. 5.7.  $Re = 100$ .

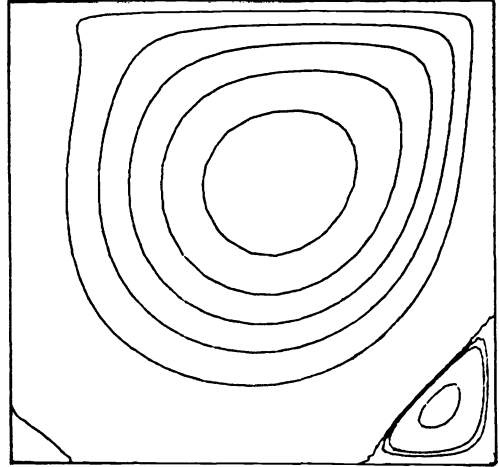


FIG. 5.8.  $Re = 400$ .

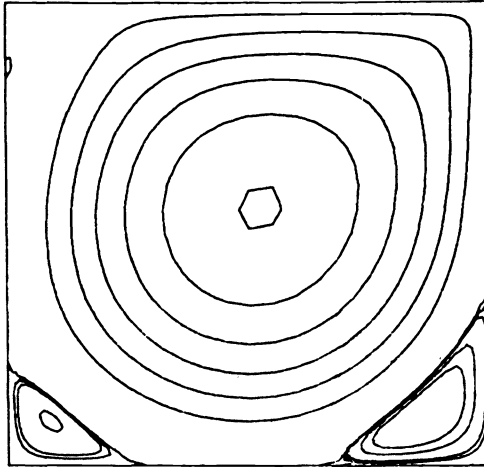


FIG. 5.9.  $Re = 1600$ .

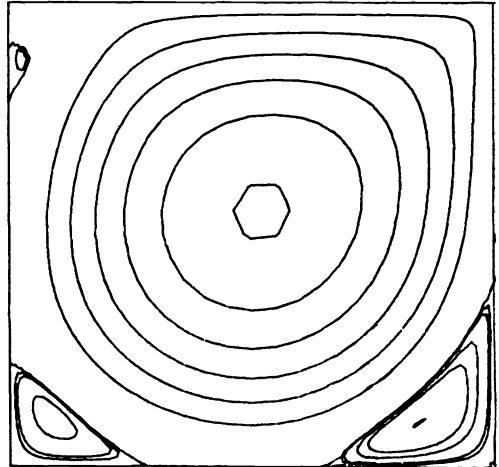


FIG. 5.10.  $Re = 2000$ .

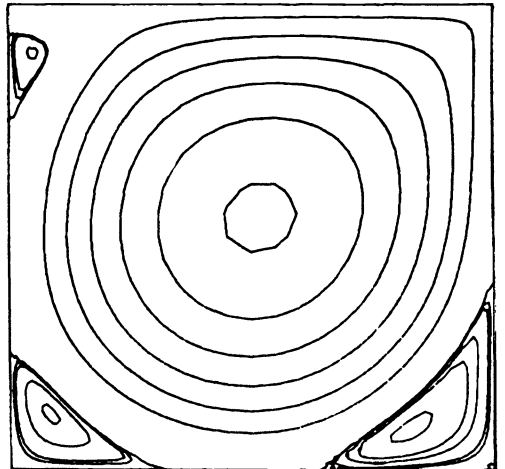


FIG. 5.11.  $Re = 3000$ .

appeared in the upper, upstream corner. These qualitative results agree with the numerical tests done by Olson–Tuann [29] using other finite element methods, by Schreiber–Keller [35] using continuation and finite difference methods and by Winters–Cliff [44] using finite elements and refinements in the corners.

A most interesting question is the possible occurrence of multiple solutions as the Reynolds number increases beyond some critical value. So far, we did not observe such behavior in the range of  $Re$  that we considered in our computations, i.e.  $0 \leq Re \leq 3000$ . Actually and to our knowledge the computed solutions obtained in the range  $0 \leq Re \leq 5000$  by various authors using different methods agree quite well; this observation suggests that multiple solutions can only appear for greater values of  $Re$ . Nevertheless it would be interesting to refine the numerical techniques in order to detect such a behavior.

**6. Conclusion.** We have discussed in this paper the solution of nonlinear boundary value problems containing a parameter by a combination of arc length continuation methods, least squares—conjugate gradient algorithms and finite element approximations. The resulting methodology is quite general and has been applied to the solution of second order and fourth order nonlinear boundary value problems whose branches of solutions may exhibit limit points and bifurcation.

**Acknowledgments.** The authors would like to thank Professor R. B. Simpson and the referees for most helpful comments and suggestions.

#### REFERENCES

- [1] E. POLAK, *Computational Methods in Optimization*, Academic Press, New York, 1971.
- [2] M. J. D. POWELL, *Restart procedure for the conjugate gradient method*, Math. Programming, 12 (1977), pp. 148–162.
- [3] L. REINHART, *Sur la résolution numérique de problèmes aux limites non linéaires par des méthodes de continuation*, Thèse de 3ème Cycle, Université Pierre et Marie Curie, Paris VI, June 1980.
- [4] M. S. BERGER, *Nonlinearity and Functional Analysis*, Academic Press, New York, 1977.
- [5] 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.
- [6] ———, *Global homotopies and Newton methods*, in Recent Advances in Numerical Analysis, C. de Boor and G. H. Golub, eds., Academic Press, New York, 1978, pp. 73–94.
- [7] J. L. LIONS, *Quelques méthodes de résolution des problèmes aux limites non linéaires*, Dunod, Gauthier-Villars, Paris, 1969.
- [8] M. G. CRANDALL AND P. H. RABINOWITZ, *Bifurcation, perturbation of single eigenvalues and linearized stability*, Arch. Rat. Mech. Anal., 52 (1973), pp. 161–180.
- [9] ———, *Some continuation and variational methods for positive solutions of nonlinear elliptic eigenvalue problems*, Arch. Rat. Mech. Anal., 58 (1975), pp. 207–218.
- [10] H. AMANN, *Fixed point equations and nonlinear eigenvalue problems in ordered Banach spaces*, SIAM Rev., 18 (1976), pp. 620–709.
- [11] F. MIGNOT AND J. P. PUEL, *Sur une classe de problèmes non linéaires avec non linéarité positive, croissante, convexe*, Comptes Rendus du Congrès d'Analyse Non Linéaire, Rome, May 1978, Pitagora Editrice, Bologna, 1979, pp. 45–72.
- [12] F. MIGNOT, F. MURAT AND J. P. PUEL, *Variation d'un point de retournement en fonction du domaine*, Comm. Partial Differential Equations, 4 (1979), pp. 1263–1297.
- [13] F. KIKUCHI, *Finite element approximations to bifurcation problems of turning point type*, in Computing Methods in Applied Sciences and Engineering, 1977, Part I, R. Glowinski and J. L. Lions, eds., Lecture Notes in Mathematics 704, Springer-Verlag, Berlin, 1979, pp. 252–266.
- [14] R. B. SIMPSON, *A method for the numerical determination of bifurcation states of nonlinear systems of equations*, SIAM J. Numer. Anal., 12 (1975), pp. 439–451.
- [15] G. MOORE AND A. SPENCE, *The calculation of turning points of nonlinear equations*, SIAM J. Numer. Anal., 17 (1980), pp. 567–576.

- [16] T. F. CHAN AND H. B. KELLER, *Arclength continuation and multigrid techniques for nonlinear eigenvalue problems*, this Journal, 3 (1982), pp. 173-194.
- [17] L. REINHART, *On the numerical analysis of the Von Karman equations: mixed finite element approximation and continuation techniques*, Numer. Math., 39 (1982), pp. 371-404.
- [18] M. G. CRANDALL AND P. H. RABINOWITZ, *Bifurcation from simple eigenvalues*, J. Funct. Anal., 8 (1971), pp. 321-340.
- [19] F. BREZZI, J. RAPPAZ AND P. A. RAVIART, *Finite dimensional approximation of nonlinear problems. Part III: Simple bifurcation points*, Numer. Math., 38 (1981), pp. 1-30.
- [20] O. A. LADYZHENSKAYA, *The Mathematical Theory of Viscous Incompressible Flows*, Gordon and Breach, New York, 1969.
- [21] R. TEMAM, *Navier-Stokes Equations*, North-Holland, Amsterdam, 1977.
- [22] V. GIRAULT AND P. A. RAVIART, *Finite element approximation of the Navier-Stokes equations*, Lecture Notes in Mathematics 749, Springer-Verlag, Berlin, 1979.
- [23] R. RAUTMANN, ed., *Approximation Methods for Navier-Stokes Equations*, Lecture Notes in Mathematics 771, Springer-Verlag, Berlin, 1980.
- [24] F. THOMASSET, *Implementation of Finite Element Methods for Navier-Stokes Equations*, Springer-Verlag, New York, 1981.
- [25] R. GLOWINSKI AND O. PIRONNEAU, *Numerical methods for the first biharmonic equation and for the two-dimensional Stokes problem*, SIAM Rev., 17 (1979), pp. 167-212.
- [26] P. G. CIARLET AND P. A. RAVIART, *A mixed finite element method for the biharmonic equation*, in Mathematical Aspects of Finite Element Methods in Partial Differential Equations, C. de Boor, ed., Academic Press, New York, 1974, pp. 125-145.
- [27] R. SCHOLZ, *A mixed method for 4th order problems using linear finite elements*, Rev. Française Autom. Inf. Rech. Op., Anal. Num., 11 (1977), pp. 197-208.
- [28] D. F. SHANNO, *Conjugate gradient methods with inexact searches*, Math. Oper. Res., 13 (1978), pp. 244-255.
- [29] M. D. OLSON AND S. Y. TUANN, *Further finite element results for the square cavity*, Proc. Third International Conference on Finite Elements in Flow Problems, Banff, Alberta, Canada, 10-13 June, 1980, Volume 1, D. H. Norrie, ed., University of Calgary, pp. 143-152.
- [30] H. B. KELLER AND D. S. COHEN, *Some positive problems suggested by nonlinear heat generation*, J. Math. Mech., 16 (1967), pp. 1361-1376.
- [31] J. P. KEENER AND H. B. KELLER, *Positive solutions of convex nonlinear eigenvalue problems*, J. Diff. Eqs., 16 (1974), pp. 103-125.
- [32] ———, *Perturbed bifurcation theory*, Arch. Rat. Mech. Anal., 50 (1973), pp. 159-175.
- [33] B. MATKOWSKY AND E. L. REISS, *Singular perturbations of bifurcations*, SIAM J. Appl. Math., 33 (1977), pp. 230-255.
- [34] D. PEROZZI, *Thesis, Part II: Analysis of optimal step size selection in homotopy and continuation methods*, California Institute of Technology, Applied Math., 1980, pp. 82-156.
- [35] R. SCHREIBER AND H. B. KELLER, *Driven cavity flows by efficient numerical techniques*, J. Comp. Phys., 40 (1983), pp. 310-333.
- [36] H. B. KELLER AND W. F. LANGFORD, *Iterations, perturbations and multiplicities for nonlinear bifurcation problems*, Arch. Rat. Mech. Anal., 48 (1972), pp. 83-108.
- [37] W. RHEINBOLDT, *Solution field of nonlinear equations and continuation methods*, SIAM J. Numer. Anal., 17 (1980), pp. 221-237.
- [38] P. DEUFLHARD, *A stepsize control for continuation methods and its special application to multiple shooting techniques*, Numer. Math., 33 (1979), pp. 115-146.
- [39] H. B. KELLER, *Practical procedures in path following near limit points*, in Computing Methods in Applied Science and Engineering V, R. Glowinski and J. L. Lions, eds., North-Holland, Amsterdam, 1982, pp. 177-183.
- [40] D. W. DECKER AND H. B. KELLER, *Multiple limit point bifurcation*, J. Math. Anal. Appl., 75 (1980), pp. 417-430.
- [41] W. C. RHEINBOLDT, *Solution fields of nonlinear equations and continuation methods*, SIAM J. Numer. Anal., 17 (1980), pp. 221-237.
- [42] R. GLOWINSKI, *Numerical Methods for Nonlinear Variational Problems*, Springer, New York, 1984.
- [43] R. GLOWINSKI, H. B. KELLER AND L. REINHART, *Continuation-conjugate gradient methods for the least square solution of nonlinear boundary value problems*, Rapport de Recherche INRIA No. 141, June 1982.
- [44] K. H. WINTERS AND K. A. CLIFFE, *A finite element study of driven laminar flow in a square cavity*, AERE Report R. 9444, 1979.

## NUMERICAL CONFORMAL MAPPING OF A TOWEL-SHAPED REGION ONTO A RECTANGLE\*

ALBERT SEIDL† AND HELMUT KLOSE‡

**Abstract.** If technical applications are involved, partial differential equations often have to be solved on a nearly rectangular domain ("towel"), one or more boundaries of which deviate from a straight line. A numerical method is presented for the mapping of such a region onto a rectangle. The calculation consists mainly in solving two coupled Laplace equations.

**Key words.** conformal mapping, Laplace equation, finite difference method, field calculations

**1. Introduction.** The finite difference method is a quite simple and very efficient method for the solution of field, diffusion and hydrodynamic problems. However, the programming effort and the calculation times increase dramatically as soon as complex geometries are involved, where the boundaries no longer fit with the rectangular grid lines. For certain classes of problems, these difficulties can be bypassed by using coordinate transformation techniques. The use of such transformations, which conserve the orthogonality of the grid, is preferred because of easier discretization. Moreover, a nonorthogonal transformation may cut down the convergence of the solver. Many physical problems have to be solved on a nearly rectangular region which is bounded by curved lines.

In this paper, we present a method for establishing an orthogonal finite difference grid in a domain bounded by four curved lines. This domain has to be simply connected and should have four rectangular corners. One way of doing this is to transform a rectangular domain  $D$  (see Fig. 1) to the quasi-rectangle  $W$  by using conformal mapping. The mapping should be done in such a way that the corners of  $D$  are mapped onto the corners of  $W$ . This is only possible for a certain length to width ratio (conformal modulus) of  $D$ , which is unknown at the beginning.

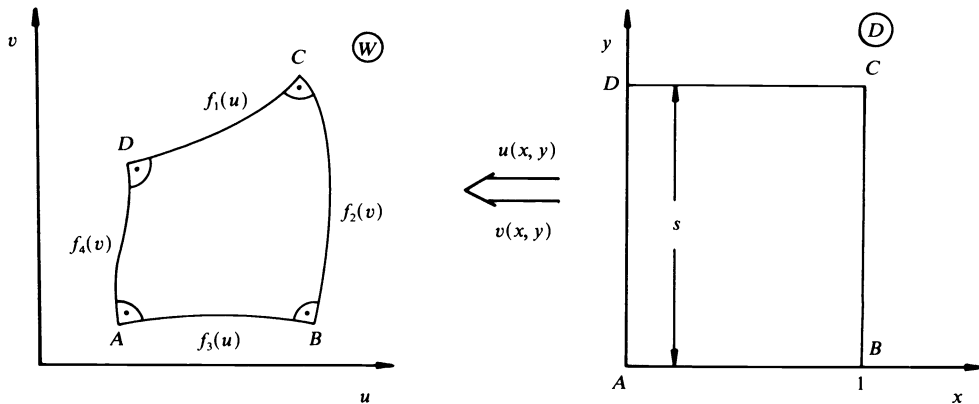


FIG. 1. Definition of physical domain  $W$  and model domain for numerical calculations  $D$ .

Many more or less efficient methods for numerical conformal mapping have been published, but most of them treat a mapping of an arbitrary region onto the unit circle. Such a mapping has traditionally been performed by solving integral equations [4],

\* Received by the editors April 26, 1983, and in revised form January 3, 1984.

† Institut für Festkörpertechnologie, Paul-Gerhardt-Allee 42, 8000 München 60, Germany.

‡ Siemens AG, ZT ZFE ME33, Otto-Hahn-Ring 6, 8000 München 83, Germany.

[13]. The authors of [1] used a finite difference solution of the Cauchy-Riemann equations, developed in [8], whereas the author of [3] calculated coefficients of a Laurent expansion.

Using a disk as an intermediate domain for our towel problem would introduce many difficulties. For some problems, a proposal by Fornberg [3] may work (see Fig. 2a). However, it maps two semi-infinite regions with only one curved boundary.

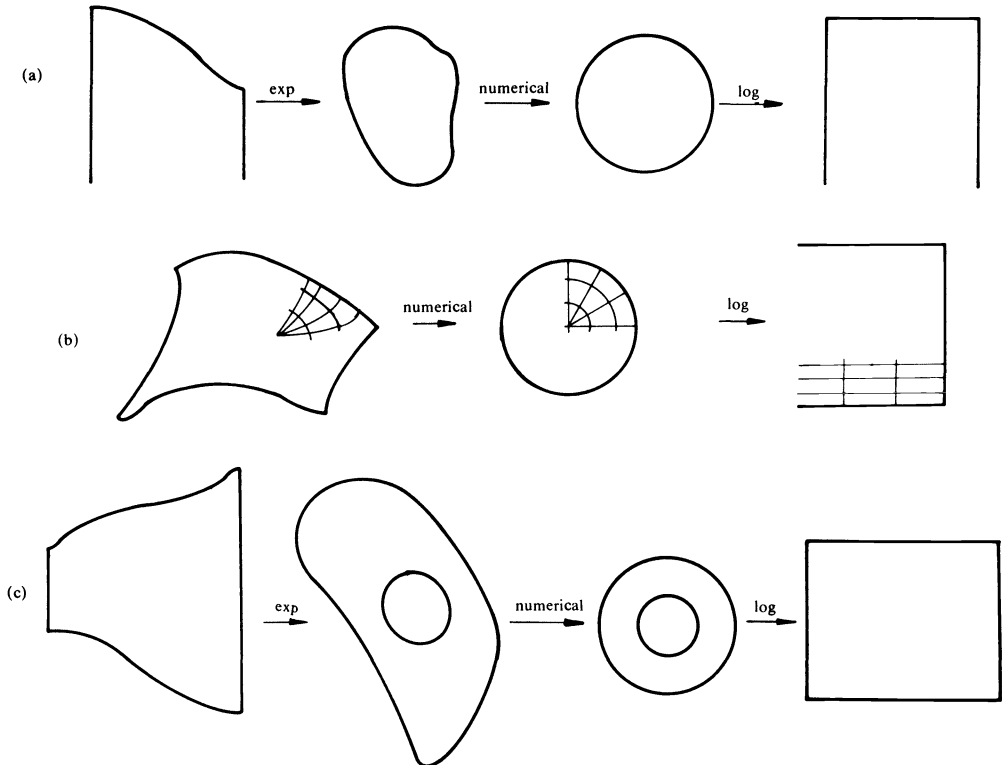


FIG. 2. *Alternative mapping schemes: (a) Fornberg's method; (b) mapping a towel onto a disk directly; (c) method using a doubly connected region.*

The proposal in Fig. 2b introduces singularities at the corners. Thus the unit-circle maps will perform poorly, because a great number of nodes or Laurent coefficients are needed. Note that we want to transform a whole finite difference grid, and that it would be necessary to evaluate the entire Laurent series or boundary integral for every grid point. The grid is transformed into a polar coordinate grid, which is usually not wanted, because grid lines are concentrated in the middle of the region.

In contrast to the above-mentioned methods, the elliptic grid generators (a survey is given in [12]) work on rectangular domains, but they do not necessarily solve the Cauchy-Riemann equations. The grid could become either nonorthogonal or anisotropic. An exception is the work of Challis and Burley [2], who performed a conformal mapping based on a FFT-solution of the Laplace equation. However, their algorithm allows for only one side of the physical domain to be curved.

In this work, the strategy will be as follows:

- The Cauchy-Riemann equations are converted into a set of two Laplace equations, which are coupled by the boundary conditions.



- Two examples are given of how the standard elliptic solvers can be employed for an iterative solution of this equation system.

- Numerical experiments are performed to test the efficiency of the algorithms and to examine for which class of problems they work.

**2. Conversion for numerical solution.** One way to obtain the functions  $u(x, y)$  and  $v(x, y)$  (see Fig. 1) so that the lines  $u = \text{const.}$  and  $v = \text{const.}$  are orthogonal is to satisfy the Cauchy–Riemann equations (CR's) [7]:

$$(1) \quad u_x = v_y,$$

$$(2) \quad u_y = -v_x.$$

As a direct consequence of (1) and (2), both  $u$  and  $v$  have to satisfy the Laplace equation:

$$(3) \quad u_{yy} + u_{xx} = 0,$$

$$(4) \quad v_{yy} + v_{xx} = 0.$$

Note that we want to map the four corners of  $D$  onto the corners of  $W$ . Thus the precondition for existence and uniqueness of  $u$  and  $v$  is that  $s = s_0$ , i.e., the length-to-width ratio of  $D$  is a certain value (conformal modulus). For further details of the conformal module problem, we refer the reader to [2], [5]. If the mapping exists,  $u$  and  $v$  will satisfy the following boundary conditions, where the Dirichlet conditions describe the shape of the physical domain  $W$ . The second CR, (2), is applied to the remaining boundaries in order to provide uniqueness for the solution of (3), (4).

	boundary	$u$	$v$	
(5a, b)	1	$u_y = -v_x$	$v = f_1(u)$	on $y = s$
(5c, d)	2	$u = f_2(v)$	$v_x = -u_y$	on $x = 1$
(5e, f)	3	$u_y = -v_x$	$v = f_3(u)$	on $y = 0$
(5g, h)	4	$u = f_4(v)$	$v_x = -u_y$	on $x = 0$

What follows is the proof that the Laplace equation, together with the boundary conditions (5), indeed solves the Cauchy–Riemann equations.

By solving the Laplace equation, we do not necessarily obtain the functions  $u$  and  $v$ , but rather an infinite number of solutions  $f$  and  $g$ , where

$$f_{xx} + f_{yy} = 0,$$

$$g_{xx} + g_{yy} = 0.$$

They differ from the solutions sought by

$$p = f - u, \quad q = g - v,$$

where  $p$  and  $q$  themselves satisfy the Laplace equation. Substitution of  $f$  and  $g$  into (1) and (2) yields

$$f_x - g_y = p_x - q_y = r_1(x, y),$$

$$f_y + g_x = p_y + q_x = r_2(x, y).$$

It is trivial to show that  $r_1$  and  $r_2$  satisfy the Laplace equation. If we apply boundary conditions that force  $r_1 = r_2 = 0$ , then we obtain the desired solutions

$$f = u, \quad g = v.$$

By applying the second CR as a boundary condition (5a, d, e, h), we obtain  $r_2=0$  at the boundaries.  $r_2=0$  follows for the whole domain because  $r_2$  satisfies the Laplace equation. With the Dirichlet boundary condition (5b, c, f, g), we obtain

$$\begin{aligned} q &= 0 \quad \text{on } y = s, \quad y = 0, \\ p &= 0 \quad \text{on } x = 1, \quad x = 0, \end{aligned}$$

together with  $r_2=0$ , and the condition for  $p$  and  $q$  to satisfy the Laplace equation follows:

$$p = q = r_1 = r_2 = 0.$$

Now let us suppose  $s \neq s_0$ , i.e., we do not yet know the value of the conformal module, and we have solved the system (3), (4), (5) with only a first estimate for  $s$ . We now have one more free variable in the geometry. Solutions  $u$  and  $v$  exist only if we introduce one more free parameter into the Cauchy-Riemann system:

(6) 
$$u_x - v_y = c_1,$$

(7) 
$$u_y + v_x = 0.$$

Now  $u$  and  $v$  can still be obtained by solving the Laplace equations analogously to the solution of (1), (2), because the additional parameter  $c_1$  has been introduced in such a way that

- $u$  and  $v$  still satisfy the Laplace equation,
- the boundary conditions (5) are not violated.

Assume  $u$  and  $v$  have been obtained with an iterative procedure by solving the Laplace equations (3) and (4) with boundary conditions (5). The constant  $c_1$  in (6) can be evaluated as the residual of the first CR (1). Thus, the parameter  $s$  (see Fig. 1) must be determined in such a way that  $c_1$  becomes zero.

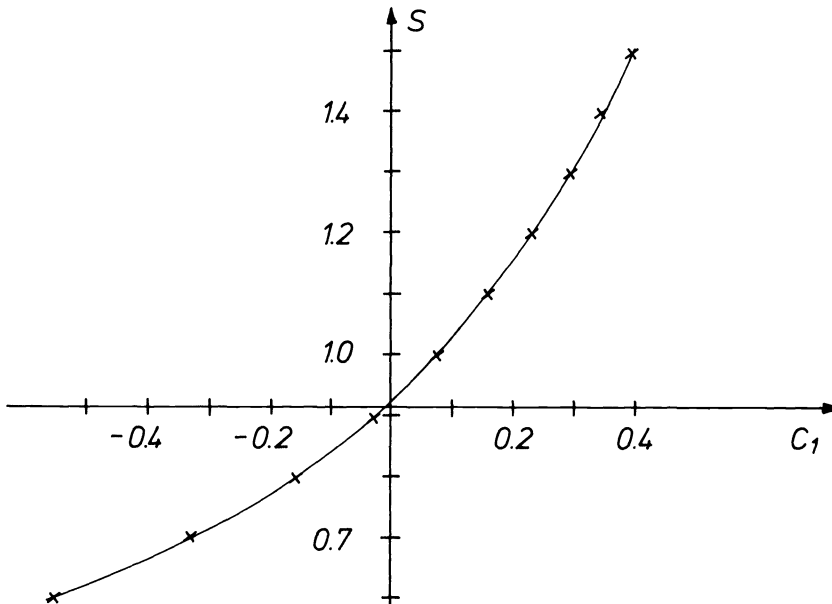


FIG. 3. Typical  $c_1(s)$  curve.

Figure 3 shows the values one obtains for  $c_1$  when solving (3), (4) with boundary conditions (5) for a given  $s$ . The problem now is to find the root of the function  $c_1(s) = 0$ . This can be done in the following manner:

- 1: set a starting value for  $s$
- 2: solve the equation system (3), (4), (5) for a given  $s$  by a Laplace solver
- 3: determine  $c_1(s)$
- 4: calculate a new value for  $s$  (e.g., by Newton's method)
- 5: repeat steps 2-4 until the desired accuracy has been attained.

Step 2 of this procedure will be done by using an iterative method, but as will be seen in the next section, it is better for practical implementations to run both iterations simultaneously.

**3. Simple iterative method.** First, we will give a detailed description of a very simple method based on point successive overrelaxation (SOR); and in the next section, we will present a survey of more efficient methods.

The Laplace equation was discretized using the well-known second order difference scheme. For easier understanding without loss of generality, we assume an equidistant grid:

$$(8) \quad u_{i+1,j} + u_{i-1,j} + \frac{u_{i,j+1} + u_{i,j-1}}{s^2} + \left(2 + \frac{2}{s^2}\right) u_{i,j} = 0.$$

For the Cauchy–Riemann boundaries, we obtain a four point formula by introducing ghost points immediately outside the domain. These ghost points are then eliminated by substituting a discretized equivalent of the boundary condition into (8). Thus, we obtain for the boundary points (e.g., boundary 1;  $y = s$ ):

$$(9) \quad u_{i+1,1} + u_{i-1,1} + \frac{2^* u_{i,2}}{s^2} - \left(2 + \frac{2}{s^2}\right) u_{i,1} = \frac{v_{i+1,1} - v_{i-1,1}}{s}.$$

A relaxation step on the Laplace equations (3), (4) with boundary conditions (5) can be constructed as follows:

- 1: 1 SOR step on  $u$ , including the Cauchy–Riemann boundary conditions (5a, e)
- 2: update the Dirichlet boundaries of  $v$  (5b, f), e.g.,  $v^{k+1} = f_3(u)$
- 3: 1 SOR step on  $v$ , including the Cauchy–Riemann boundary conditions (5d, h)
- 4: update the Dirichlet boundaries of  $u$  (5c, g).

For a nonequidistant grid, successive overrelaxation by lines (SLOR) works better than point SOR.

An outer iteration cycle is necessary to find the root of the function  $c_1(s)$  (see Fig. 3 and (6)). After a certain number of relaxation iterations, as described above, we update the value of  $s$ . This can be done by using a secant iteration:

$$(10) \quad s^{k+1} = s^k - \frac{s^k - s^{k-1}}{c_1^k - c_1^{k-1}} c_1^k.$$

An estimate for  $c_1$  is calculated by using a discretized equivalent of (6):

$$(11) \quad c_1 = \frac{1}{n^2} \sum_{i=1}^n \sum_{j=1}^n \left( \frac{u_{i+1,j} - u_{i-1,j}}{2h} - \frac{v_{i,j+1} - v_{i,j-1}}{2sh} \right).$$

Better stability than in the case of the Newton method was obtained by using the following formula, derived from (11) and setting  $c_1$  to zero.

$$(12) \quad \sum_{i=1}^n \sum_{j=1}^n \left( \frac{u_{i+1,j} - u_{i-1,j}}{2h} - \frac{s_{\text{old}}}{s_{\text{new}}} \frac{v_{i,j+1} - v_{i,j-1}}{2hs_{\text{old}}} \right) = 0.$$

Numerical experiments were performed to determine how many inner relaxation iterations per outer iteration yield the best convergence. For all examples treated in this paper, the optimal number of inner iterations was about 5.

It is reasonable to assume that the number of relaxation iterations necessary to solve only the coupled Laplace equations (3), (4), (5) without an  $s$ -iteration is the minimum number required for the solution with  $s$ -iteration. The optimal number of iterations was typically exceeded by 10% to 20%.

As a stopping criterion, both the correction of  $s$  and the residual of the discretized Laplace equation were checked. A flow chart of the whole algorithm is shown in Fig. 4.

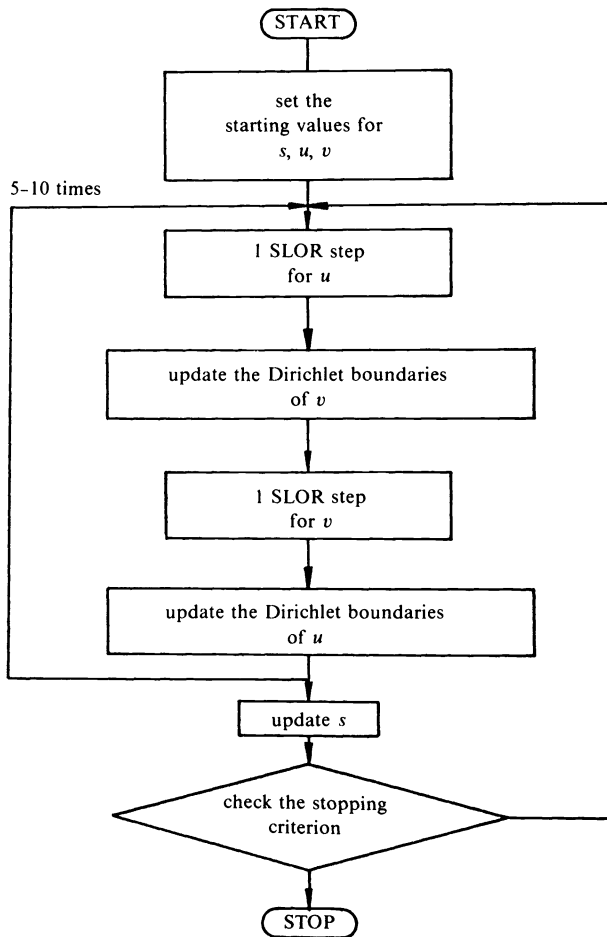


FIG. 4. Flow chart of iterative method implemented in this work.

**4. More efficient methods.** The relaxation method described above is carried out with minimal storage of 2 times the number of grid points. If more storage space is available, fast Laplace solvers can be used as well. Let the discretized equivalent of

(3) and (4), with boundary conditions (5), be represented by the nonlinear equation

$$(13) \quad \mathcal{L}(w) = 0$$

with  $w = \begin{pmatrix} u \\ v \end{pmatrix}$ .

Many efficient iterative solvers, such as multigrid [6] and incomplete  $LU$  decomposition [11], exist that do not work with (13) directly, but rather, with an equation for the residual of (13).

If  $w^k$  denotes the approximation for  $w$  after the  $k$ th iteration and

$$\mathcal{L}(w^k) = r,$$

a correction vector  $e$  is calculated by solving the linearized matrix equation

$$(14) \quad -Le = r.$$

An iteration cycle can then be constructed:

$$w^{k+1} = w^k + e.$$

Inside the domain  $D$ , the problem is linear. Therefore,  $L$  is identical with  $\mathcal{L}$ . For the boundaries, the nonlinear coupled Dirichlet conditions (5b, c, f, g) were substituted into (9). The resulting formula was linearized in a Newton-like manner:

$$(15) \quad \left(1 - \frac{f'_3(u_{i-1,j})}{s}\right) e_{i-1,1} - \left(2 + \frac{2}{s^2}\right) e_{i,1} + \left(1 + \frac{f'_3(u_{i+1,j})}{s}\right) e_{i+1,1} + \frac{2}{s^2} e_{i,2} - h^2 r_{i,1} = 0.$$

In this work, a multigrid method [9] was used. The following procedure describes what we will later refer to as “one multigrid step:”

- 1: restriction of the matrix equation for a coarse grid
- 2: relaxation sweep
- 3: interpolation on the next finer grid
- 4: repeat steps 2 and 3 until the finest grid is reached.

A detailed description of the multigrid method can be found in [6] and [9]. It can be applied to our problem as follows:

- 1: guess starting values for  $u$ ,  $v$  and  $s$  by applying a multigrid step directly to the nonlinear system (13).  $s$  is adjusted at every level
- 2: calculate  $e$  by applying one multigrid step to (14)
- 3: update  $s$
- 4: repeat steps 2 and 3 until a converged solution is obtained.

**5. Hybrid methods.** The results published in [10] indicate that the CPU times for the solution of a Laplace equation are below  $7.1 \cdot 10^{-4}$  sec per point on a Cyber 175. The reason that this optimum is not reached for some cases is due to the nonlinearity of the problem, i.e., the boundary conditions (5) and the  $s$ -iteration. For special problems, it is possible to bypass some of these nonlinearities.

For many technical problems, we have only two curved boundaries on opposite sides of the domain. Theodorsen’s integral equation for a doubly-connected region, as described by Gaier [4], can be applied directly as shown in Fig. 2c. A formula for the calculation of the conformal module is also given in [4]. As soon as the mapping of the boundaries is done, the coupled boundary conditions (5b, c, f, g), e.g.,

$$v = f_1(u),$$

become ordinary Dirichlet conditions:

$$v = f_1(x).$$

Thus, if the boundaries have been mapped by using an integral equation or a Laurent series method [3], a multigrid solution of the Laplace equation is a good alternative to the solution of a line integral or to the evaluation of the entire Laurent series for every grid point.

**6. Numerical results.** Our testing program was implemented in FORTRAN and runs on both a Cyber 175 and a VAX 11/750. It has the ability to work with a nonequidistant grid and with four curved boundaries. The computation times given in the following section were obtained with this program on the Cyber with simple accuracy (60 bit wordlength). It performed just as well on the VAX with 32 bit. The CPU times were 17 times greater on the VAX. It is obvious that the program can be further optimized if one has a special application in mind. Particularly if the discretization is restricted to be equidistant, a reduction in CPU time of at least 50% is possible, due to reduced effort for the calculation of the coefficients of the Laplace equation.

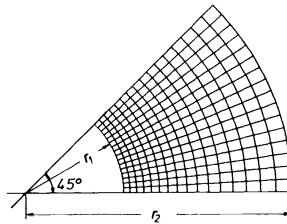


FIG. 5. The rectangle was mapped onto the part of a sector between  $1 < r < e$  using the function  $w = e^z - 1$ . This figure shows the numerical solution obtained on the grid  $17 * 17$ .

To test the accuracy, the numerical solution was compared with an analytical one. A rectangle was mapped onto the sector of a circle by using the exponential function according to Fig. 5. The following measure was used to judge the accuracy:

$$(16) \quad \bar{f} = \frac{1}{n^2} \sum_{i=1}^n \sum_{j=1}^n |\tilde{u}_{i,j} - u(x, y)| + |\tilde{v}_{i,j} - v(x, y)|,$$

where  $\tilde{u}, \tilde{v}$  denote the numerically calculated solutions and  $u, v$  the exact solutions. For the case where no analytical solution exists, a further criterion was checked using the discretized equivalents of the CR's (1) and (2):

$$(17a) \quad \text{def1} = \frac{1}{n^2} \sum_{i=1}^n \sum_{j=1}^n \left| \frac{\tilde{u}_{i+1,j} - \tilde{u}_{i-1,j}}{2h} - \frac{\tilde{v}_{i,j+1} - \tilde{v}_{i,j-1}}{2sh} \right|,$$

$$(17b) \quad \text{def2} = \frac{1}{n^2} \sum_{i=1}^n \sum_{j=1}^n \left| \frac{\tilde{u}_{i+1,j} - \tilde{u}_{i-1,j}}{2sh} + \frac{\tilde{v}_{i,j+1} - \tilde{v}_{i,j-1}}{2h} \right|.$$

TABLE 1

grid	$\bar{f}$ (see (10))	def <sub>1,2</sub> (see (11))	s
analytical	0	0/0	.78540
65*65	6.86E-5	4.7E-5/2.4E-6	.78534
33*33	2.75E-4	1.4E-4/4.6E-6	.78517
17*17	1.11E-3	5.9E-4/6.9E-5	.78448
9*9	4.32E-3	1.2E-3/1.2E-6	.78178

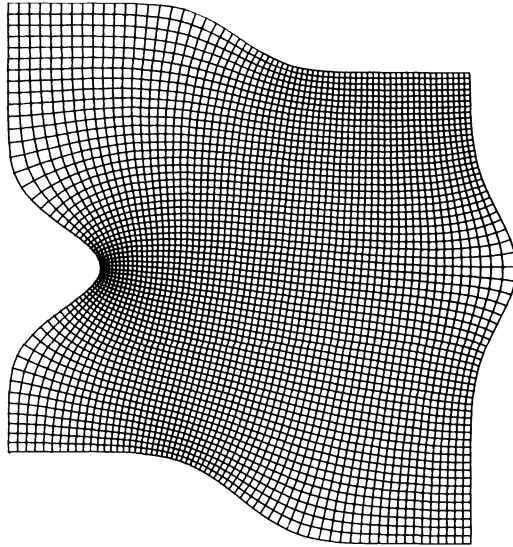


FIG. 6. Typical towel:

grid	33*33	65*65
def <sub>1,2</sub>	3.6E-3/1.8E-6	1.1E-3/9.5E-7
calculated <i>s</i>	1.0364	1.0367
CPU time (SOR)	2.2 sec	15 sec.

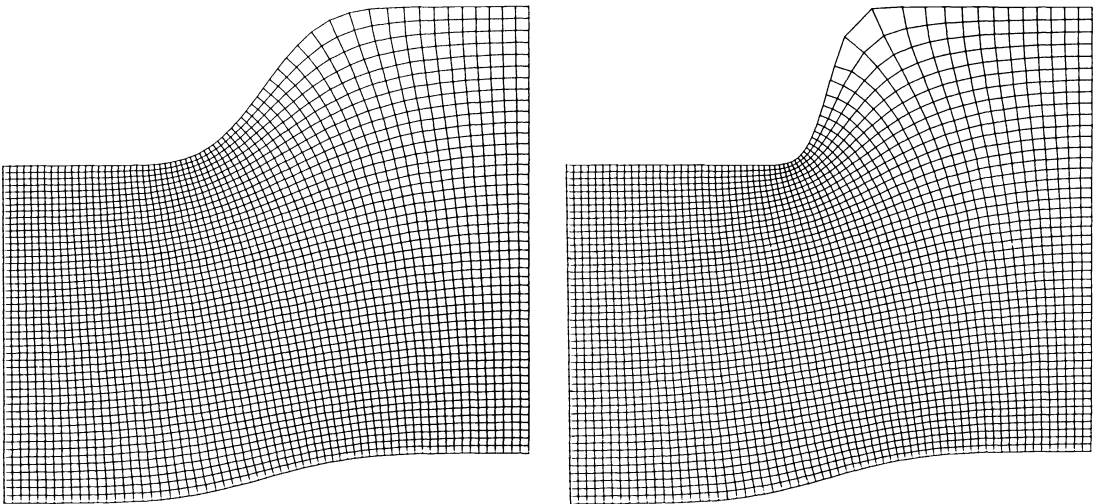


FIG. 7. Comparison of calculation times with SLOR and multigrid depending on the smoothness of the boundary  $y=0$ .

case 1:	33	65	33	65	case 2 (MG):	33	65
MG	5	5	1.382	3.802		5	6
SLOR	12	22	1.642	10.4		1.213	4.32

*n* *T*

*n* := number of outer iterations  
*T* := CPU time/seconds  
 33 = coarse grid: 33\*25 grid points  
 65 = fine grid: 65\*49 grid points

The numerical results exhibit an error dependence proportional to  $1/h^2$ . The calculated value for  $s$  was used as an additional criterion.

In Fig. 6, an example of a domain bounded by four curved lines is presented.

Further calculations were done to investigate the dependence of convergence on the shape of the boundary and on the choice of the Laplace solver. The results (Fig. 7) show how the calculation times become better for smoother boundaries. Case 2 is an example where convergence slows down to an almost impractical value with our SOR procedure. When using the multigrid method, the example shown was the steepest boundary profile for which reasonable results were obtained. Moreover, a comparison is made in Fig. 7 between the CPU times using a SLOR ( $T_{\text{SLOR}}$ ) and a multigrid solver ( $T_{\text{MG}}$ ).

**7. Summary.** In this paper, a new method has been introduced to solve the Cauchy-Riemann equations using a relatively simple and fast computer program. With this algorithm, it is possible to calculate the velocity and potential distribution for hydrodynamic and electric field problems on nonrectangular regions. Moreover, this method can be used to map a nonrectangular domain onto a rectangular one, in order to solve the Poisson equation by standard elliptic solvers for curved boundaries on a very simple region. Thus, this numerical conformal map, together with the finite difference method, could be used as a substitute for various finite element applications, which require complicated equation systems, complex codes and long computation times.

**Acknowledgments.** The authors wish to thank R. Fössmeier and L. Zink (Institut für Informatik, TU München) for valuable advice and helpful discussions.

#### REFERENCES

- [1] S. CHAKRAVARTHY AND D. ANDERSON, *Numerical conformal mapping*, Math. Comp., 33 (1979), pp. 953-969.
- [2] N. V. CHALLIS AND D. M. BURLEY, *Numerical method for conformal mapping*, IMA J. Numer. Anal., 2 (1982), pp. 169-181.
- [3] B. FORNBERG, *A numerical method for conformal mappings*, SIAM J. Sci. Stat. Comput., 1 (1980), pp. 386-400.
- [4] D. GAIER, *Konstruktive Methoden der Konformen Abbildung*, Springer Tracts in Natural Philosophy 3, Springer-Verlag, Berlin, 1964.
- [5] ———, *Determination of the conformal module of quadrilaterals by difference methods*, Numer. Math., 19 (1972), pp. 179-194.
- [6] W. HACKBUSCH, *Introduction to multigrid methods*, Lecture Series of Computational Fluid Dynamics, Karman Institute of Fluid Dynamics, Rhode Saint Genese,
- [7] P. HENRICI AND R. JELTSCH, *Komplexe Analysis für Ingenieure*, Birkhäuser Verlag, Basel, 1977.
- [8] H. LOMAX AND E. D. MARTIN, *Fast direct numerical solution of the nonhomogeneous Cauchy-Riemann equations*, J. Comp. Phys., 15 (1974), pp. 55-80.
- [9] A. SEIDL, *A multigrid method for the solution of the diffusion equation in VLSI process modeling*, IEEE Trans. Electron Dev., ED-30 (1983), pp. 999-1004.
- [10] J. STOEHR AND R. BULIRSCH, *Einführung in die Numerische Mathematik II*, Springer-Verlag, Berlin 1979.
- [11] H. L. STONE, *Iterative solution of implicit approximations of multidimensional partial differential equations*, SIAM J. Numer. Anal., 5 (1968), pp. 530-558.
- [12] J. F. THOMPSON, *Numerical grid generation*, April 1982, Nashville, TN, pp. 79-105.
- [13] R. WEGMANN, *An iterative method in conformal mapping*, Numer. Math., 30 (1978), pp. 453-466.



## NUMERICAL RESOLUTION OF MAXWELL'S EQUATIONS IN POLARISABLE MEDIA AT RADIO AND LOWER FREQUENCIES\*

BERNARDO COCKBURN†

**Abstract.** A new model of geophysical prospecting based on a modification of Maxwell's equations governing low frequency electromagnetic wave propagation in a polarisable medium is considered. The main difficulty of the mathematical and numerical study of the problem is the treatment of the convolution term containing the information on the polarisable properties of the medium. A powerful method to treat the convolution term numerically is proposed and given a firm mathematical basis. Numerical simulations demonstrating the efficiency of the method are presented.

**Key words.** Maxwell's equations, polarisation, convolution, Fourier transform

**1. Introduction.** In [2] Dias suggests the use of the concept of a total current complex conductivity  $\sigma$  to model the electromagnetic behaviour of polarisable media at radio and lower frequencies. In this range of frequencies the relationship between the electric and the magnetic fields can be assumed to be linear. His very form is obtained in the frequency domain.

Dias writes down Maxwell's equations in the frequency domain as follows,

$$(1) \quad \begin{aligned} \operatorname{rot} \mathbf{H} &= \sigma \mathbf{E}, \\ \operatorname{rot} \mathbf{E} &= -i\omega \mathbf{B}, \\ \operatorname{div} \mathbf{B} &= 0, \\ \operatorname{div} \mathbf{D} &= 0, \end{aligned}$$

where the total current complex conductivity  $\sigma$  is given by

$$(2) \quad \begin{aligned} \sigma &= \sigma_0(\lambda + (1 - \lambda)\tilde{\sigma}), \\ \tilde{\sigma} &= (1 + (i\omega/\omega_c)^{1/2})^{-1} \end{aligned}$$

(see Fig. (1)), and depends on spatial variables by means of the conductivity  $\sigma_0$ , a real valued function, the polarisation parameter  $\lambda$ ,  $\lambda \geq 1$ , and the critical frequency  $\omega_c$ .

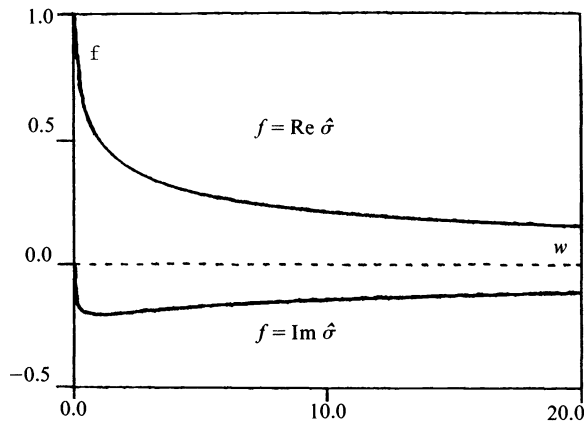


FIG. 1. The real and imaginary parts of  $\hat{\sigma}(w) = (1 + (iw)^{1/2})^{-1}$ .

\* Received by the editors January 16, 1984, and in revised form June 21, 1984.

† INRIA, Domaine de Voluceau, Rocquencourt B.P. 105 78153, Le Chesnay Cedex, France.

It depends also on the frequency  $\omega$  by means of the polarisation law  $\vec{\sigma}$  only when the medium exhibits electrical polarisation, i.e., only when  $\lambda > 1$ .

The purpose of this paper is to study this model in the one-dimensional case, as we are interested in applying it to geophysical prospecting.

More precisely, the problem to be studied is the determination of the magnetic field at the surface,  $\mathbf{H}_0$ , when the electrical field  $\mathbf{E}_0$  at the same point is known. See Fig. 2.

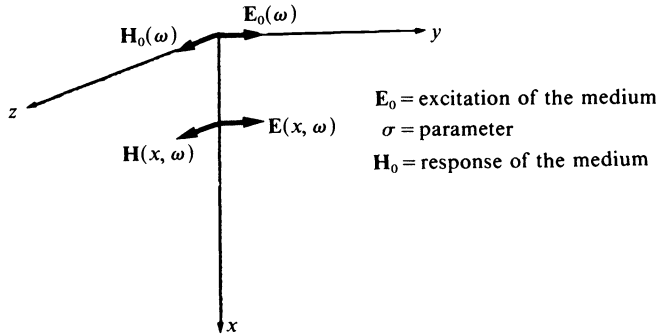


FIG. 2. The precise problem we consider.

Fixing the frequency  $\omega$  equations (1) can be solved. Then to obtain the time-dependent solution the inverse Fourier transform<sup>1</sup> is applied. This method will be referred to as the frequency method. Equations (1) can be solved analytically in the one-dimensional case (and on a stratified medium) but this advantage, from the numerical point of view, no longer exists in the case of 2 or 3 dimensions.

This remark motivates us to solve directly the equations (1) written in the time domain,

$$\begin{aligned}
 \mu_0 \operatorname{rot} \mathbf{h} &= \nu^{-1}(e - K * e), \\
 \operatorname{rot} e &= -\mu_0 \partial_t \mathbf{h}, \\
 \operatorname{div} \mathbf{h} &= 0, \\
 \operatorname{div} e &= 0,
 \end{aligned}
 \tag{3}$$

where

$$\begin{aligned}
 \nu &= (\mu_0 \sigma_0 \lambda)^{-1}, \\
 K &= (1 - \lambda^{-1}) F^{-1}(\vec{\sigma}).
 \end{aligned}
 \tag{4}$$

The method of solving (3) will be called the time method, in contrast to the frequency method ( $F$  denotes the Fourier transform).

Mathematically the polarisation phenomenon can be considered as a perturbation of an autonomous system introducing a memory in it. So, to obtain numerical solutions we have

- (i) to discretize the convolution term with accuracy (see § 3),
- (ii) to stock all the history of the electrical field (see § 4).

To avoid the second difficulty we propose to replace the polarisation law  $\vec{\sigma}$  by another

<sup>1</sup> The Fourier transform used in this paper is the conjugate of the one that is used by Dias in [2].

$\tilde{\sigma}_{a_N}$  of the form

$$(5) \quad \tilde{\sigma}_{a_N} = \sum_{k=1}^N b_k (1 + ia_k \omega / \omega_c)^{-1},$$

where  $a_k, b_k \in \mathbb{R}$  for  $1 \leq k \leq N$ . In this way the convolution term is replaced by a simple sum of  $N$  functions each of them satisfying an ordinary differential equation of first order (see (14)).

The organization of this paper is as follows. In § 2 we formulate the time problem. In § 3 we show that the polarisation phenomenon introduces only a slight perturbation on the solution of the case  $K = 0$ . In § 4 we consider the problem of the approximation of  $\tilde{\sigma}$  by  $\tilde{\sigma}_a$ . This problem is written as a nonlinear minimization one and is solved in the particular case we consider, using an algorithm based on the conjugate-gradient method. Finally in § 5 we show some numerical simulations comparing the time method solution with that of the frequency-method. The latter solution will be considered to be exact.

**2. Formulation of the time-problem.** We suppose that

- (i) all the functions appearing in (3) depend only on  $t$  and on the depth  $x$ ,
- (ii) the system is at rest for  $t \leq 0$ ,
- (iii) it is excited by imposing the electrical field at the surface, and
- (iv) on values of the fields  $\mathbf{e}$  and  $\mathbf{h}$  only past values have influence.

We shall restrict ourselves to the domain  $Q = ]0, L[ \times ]0, T[$ .

With these assumptions we must write the electric and magnetic field as follows,  $\mathbf{e} = (0, e_1, e_2)$  and  $\mathbf{h} = (\text{const}, -h_2, h_1)$  where  $(e_i, h_i)$   $i = 1, 2$  is the solution of the following equations,

$$(6) \quad \begin{aligned} \mu_0 \partial_x h + \nu^{-1}(e - g(e)) &= 0 && \text{in } Q, \\ \mu_0 \partial_t h + \partial_x e &= 0 && \text{in } Q, \\ e(t=0) = h(t=0) &= 0 && \text{on } [0, L], \\ e(x=0) = \phi, e(x=L) &= 0 && \text{on } [0, T], \end{aligned}$$

where

$$(7) \quad g(e)(t) = (K * e)(t) = \int_0^t K(t-s)e(s) ds \quad \forall t \geq 0.$$

We point out that  $e(t) = 0$  for  $t \leq 0$  and that

$$(7') \quad K(t) = 0 \quad \forall t < 0$$

by (iv).

We shall call “the time problem” the problem of finding solutions of (6), and we shall denote it by (P). In [3] a mathematical analysis of problem (P) is done. See also [5].

For regular parameters existence and uniqueness of the solution are obtained by writing it as a perturbation of the well-known solution of the case  $K = 0$  and then applying a fixed point technique, see Brezis [1]. In order to extend this result to a wide class of parameters appearing in applications the method of a priori estimates is used, see Lions [7].

These estimates give immediately the continuous dependence of the solutions on the parameters.

In the case considered here the bilinear form

$$a(e, v) = (e, v) - (g(e), v), \quad (e, v) \in V \times V,$$

where  $(\cdot, \cdot)$  stands for the inner product of  $V = L^2(0, T; L^2(\mathbb{R}^+))$ , is  $V$ -elliptic, i.e.,

$$(8) \quad \exists \beta > 0: \quad a(e, e) \geq \beta \|e\|_V^2$$

the  $V$ -ellipticity of  $a$  is verified if the following condition is fulfilled,

$$(8') \quad \exists \beta > 0: \quad \operatorname{Re} (F(K(\cdot, x))(\omega)) \leq 1 - \beta \quad \forall x \text{ a.e. in } \mathbb{R}^+, \quad \forall \omega \in \mathbb{R}.$$

Roughly speaking, it is this property that makes our solution behave in the same way as that of the case  $K = 0$ . Because of this it is possible to get a nice estimate of the energy, prove the existence of a periodic solution of the problem (P) with  $g$  replaced by

$$g'(e)(t) = \int_{-\infty}^t K(t-s) e(s) ds \quad \forall t \in \mathbb{R}$$

and show that as  $t \rightarrow \infty$  the solution of (P) converges to the periodic solution, when a  $T_0$ -periodic boundary condition  $\phi$  satisfying the mean time zero condition,

$$\int_0^{T_0} \phi(s) ds = 0$$

is imposed.

**3. The polarisation phenomenon: a slight perturbation.** In practical cases the polarisation introduces only a slight perturbation on the solution of the case  $K = 0$  (or equivalently  $\lambda = 1$ , see (2)).

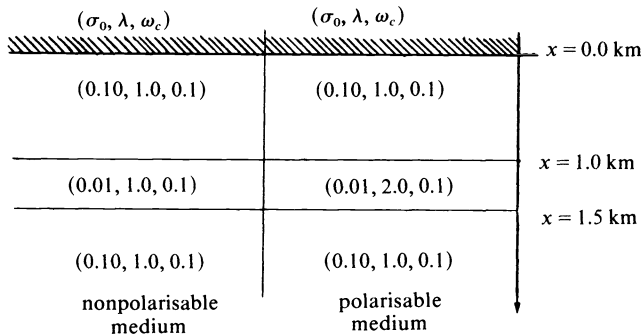


FIG. 3. Values of the space-dependent parameters.

To see this we shall compare the response  $h_0$  (i.e. the values of the magnetic field at the surface, calculated by the frequency method), for each of the media shown in Fig. 3.

The response obtained on the polarisable (res. nonpolarisable) media will be denoted by  $h_0(p)$  (resp.  $h_0(np)$ ). The relative importance of the polarisation phenomenon will be measured by using what will be called the polarisation pattern,

$$(9) \quad p = (h_0(p) - h_0(np)) / h_0(np).$$

This pattern is shown in Fig. 5b in the case of a boundary condition  $\phi$  equal to zero except in the interval  $]0, 1[$  in which it is equal to 1. It can be seen that the polarisation introduces a perturbation of less than 1% of the response of the nonpolarisable medium. This shows the necessity of using a very accurate scheme to discretise problem (P).

**4. The approximate polarisation law.**

**4.1. Introducing the approximate polarisation law.** Even if we discretise problem (P) with such a scheme we still have the following difficulty: to evaluate  $g(e)$ , (see (7)), we have to stock all the past values of  $e$  and this is not very convenient to do in 2 or 3 dimensions. The best way to overcome this difficulty would be to use the same memory place as in the case of a vanishing convolution kernel. But, this can only be achieved when  $K$  is an exponential function.

In this way we are led to replace our exact kernel  $K$  by a sum of exponential functions and this is equivalent to replacing  $\tilde{\sigma}$  (see (2)) by a function  $\tilde{\sigma}_{a_N}$  given by (5).  $g(e)$  is then replaced by a sum of  $N$  functions, each of them satisfying a simple first order differential equation (see (14)).

We want  $\tilde{\sigma}_{a_N}$  to have the two most important properties of the function  $\tilde{\sigma}$ : (7') and (8'). More precisely, if we put

$$K_{a_N} = (1 - \lambda^{-1})F^{-1}(\tilde{\sigma}_{a_N}) \quad (\text{see (4)}),$$

then we shall only consider those  $\sigma_{a_N}$  for which  $K_{a_N}$  verifies (7') and (8'). In fact, (7') is verified if and only if

$$(10) \quad a_k > 0, \quad 1 \leq k \leq N,$$

and (8') is verified if,

$$(11) \quad \sum_{k=1}^N b_k(1 + a_k \eta^2)^{-1} \leq 1 \quad \forall \eta \in \mathbb{R}.$$

In other words, we shall replace  $\tilde{\sigma}$  by a function  $\tilde{\sigma}_{a_N}$  given by (5), verifying the properties (10) and (11).

**4.2. Fitting  $\tilde{\sigma}_{a_N}$  to  $\tilde{\sigma}$ .** Let us introduce  $\hat{\sigma}(w) = \tilde{\sigma}(w\omega_c(x), x)$  and  $\hat{\sigma}_{a_N}(w) = \tilde{\sigma}_{a_N}(w\omega_c(x), x)$ . It can be proved (see [3] [4]) that a choice  $\hat{\sigma}_{a_N}$  fitting  $\hat{\sigma}$  in an interval of the form  $]0, w_1]$  is sufficient to ensure that the solutions of our time-problem with  $\tilde{\sigma}$  and  $\tilde{\sigma}_{a_N}$  remain close to each other. (Here  $\hat{\sigma}_{a_N}$  fitting  $\hat{\sigma}$  means that  $\hat{\sigma}_{a_N}$  is a good approximation of  $\hat{\sigma}$  in the region of (low) frequencies lying between 0 and  $w_1\omega_c(x)$ , for each  $x \in ]0, L[$ ).

By (5) and the definition of  $\hat{\sigma}_{a_N}$ , we have

$$(12) \quad \hat{\sigma}_{a_N}(w) = \sum_{k=1}^N b_k(1 + ia_k w)^{-1}$$

so that we can "identify" the function  $\hat{\sigma}_{a_N}$  with the point  $x_N = (a_1, b_1, \dots, b_N) \in \mathbb{R}^{2N}$ .

We shall take  $\hat{\sigma}_{a_N}$  to be the solution of the following problem.

Find  $\hat{\sigma}_{a_N}$  in  $\mathcal{A}_N$  such that,

$$J(\hat{\sigma}_{a_N}) \leq J(\hat{\sigma}_{a_N}) \quad \forall \hat{\sigma}_{a_N} \text{ in } \mathcal{A}_N,$$

where

$$J(\hat{\sigma}_{a_N}) = \|\hat{\sigma}_{a_N} - \hat{\sigma}\|_{H^1(0, w_1)}^2,$$

$$\mathcal{A}_N = \{\hat{\sigma}_{a_N} \text{ of the form (12) verifying (10) and (11)}\}.$$

This is a nonlinear minimisation problem that we solve using an algorithm based on the conjugate-gradient method. We remark that  $\mathcal{A}_N$  is of dimension  $2N$ . For details see [3]. Theoretical convergence results can be found in [5].

**4.3. The numerical results.** In this case the value  $w_1 = 20$  has been taken. The numerical results given in Figs. 4 and 5 show that this value was large enough. On the one hand, in Fig. 4 a comparison of the real and imaginary parts of  $\hat{\sigma}$  and  $\sigma_{a_N}$  is made. A good fitting is obtained with  $N = 4$ .

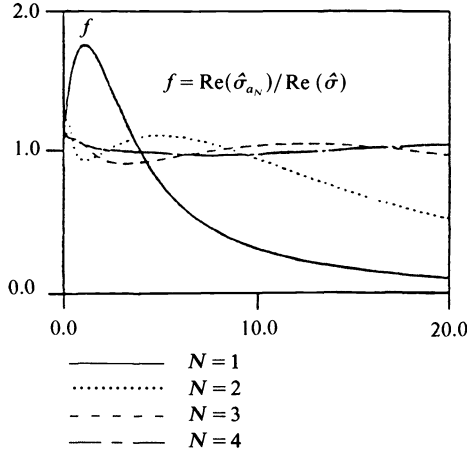


FIG. 4a. Comparison of the real parts of  $\hat{\sigma}$  and  $\hat{\sigma}_{a_N}$

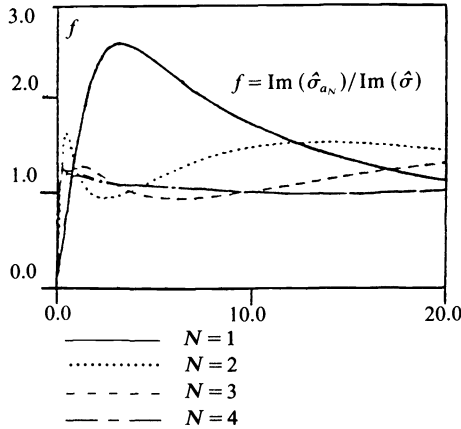


FIG. 4b. Comparison of the imaginary parts of  $\hat{\sigma}$  and  $\hat{\sigma}_{a_N}$

On the other hand, in Fig. 5 a comparison of the polarisation pattern  $p$  (see (9)) and the errors introduced by replacing  $\hat{\sigma}$  by  $\tilde{\sigma}_{a_N}$  is made. These errors, which will be called the approximation errors,  $e_{app}$ , are given by

$$(13) \quad e_{app}(N) = p - p_N,$$

where  $p_N$  denotes the polarisation pattern obtained by the frequency method using  $\tilde{\sigma}_{a_N}$  instead of  $\hat{\sigma}$ . It can be seen that they decrease monotonically with  $N$  and that for  $N = 4$ ,  $e_{app}(N)$  does not distort significantly the polarisation pattern.

So, it is possible to replace the exact polarisation law  $\hat{\sigma}$  by an approximate law  $\tilde{\sigma}_{a_N}$  and still have the same polarisation properties of the system.

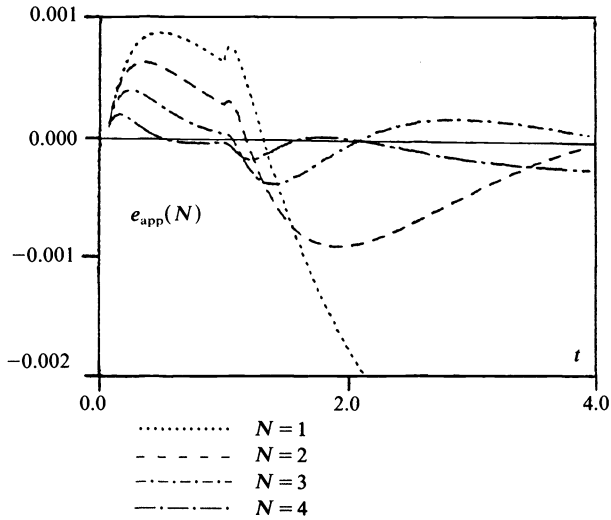


FIG. 5a. Comparison of the errors in the polarisation pattern due to the approximation of  $\hat{\sigma}$  by  $\hat{\sigma}_{N_a}$  (the polarisation patterns have been obtained by the frequency method).  $e_{app}(N) = p - p_N$  (see (9), (13)).

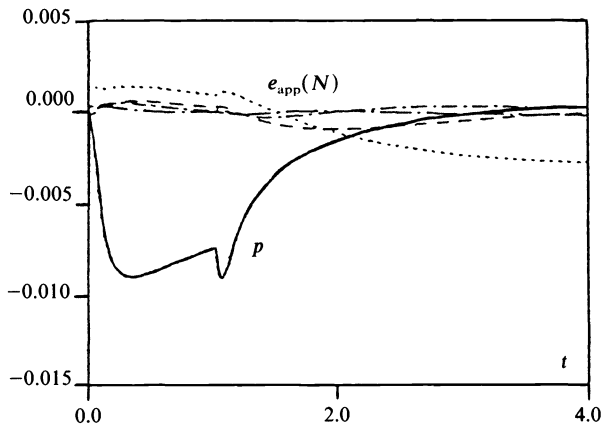


FIG. 5b. Comparison of the approximation error  $e_{app}(N)$  and the exact polarisation pattern. — the exact polarisation pattern  $p$  (see (9)),  $\cdots$  the approximation error for  $N = 1$ ,  $---$  the approximation error for  $N = 2$ ,  $-\cdot-\cdot-$  the approximation error for  $N = 3$ ,  $-\cdot-\cdot-$  the approximation error for  $N = 4$ .

**5. Numerical analysis.**

**5.1. Discretisation of the time-problem.** If in (6)  $\tilde{\sigma}$  is replaced by  $\tilde{\sigma}_{a_N}$ , the equations can be rewritten as follows:

$$\begin{aligned}
 \mu_0 \partial_x h + \nu^{-1} e &= \nu^{-1} \sum_{k=1}^N f_k && \text{in } Q, \\
 \partial_t f_k &= -\frac{\omega_c}{a_k} f_k + (1 - \lambda^{-1}) b_k \frac{\omega_c}{a_k} e && \text{in } Q, \quad 1 \leq k \leq N, \\
 \mu_0 \partial_t h + \partial_x e &= 0 && \text{in } Q, \\
 e(t=0) = h(t=0) &= 0 && \text{on } [0, L], \\
 f_k(t=0) &= 0 && \text{on } [0, L], \quad 1 \leq k \leq N, \\
 e(x=0) = \phi, \quad e(x=L) &= 0 && \text{on } [0, T].
 \end{aligned}
 \tag{14}$$

These equations are discretised using finite differences. The scheme obtained in this way is implicit and unconditionally stable. It is of second order in time and of second order in space locally (when two consecutive step sizes are equal). When  $K$  vanishes the scheme reduces to the classical Crank–Nicolson scheme, see Richtmyer and Morton [9].

To study the properties of this scheme the same techniques applied in the continuous case are used (see [3]). The discrete equivalents of the  $V$ -ellipticity properties (8) and (8') hold in this case and it is possible to get estimates of the energy as in the continuous case when the time and space step sizes are small. Stability and convergence properties of this scheme are then exactly the same as in the case when  $K$  vanishes.

We shall discretise the domain  $Q$  as in Fig. 6.

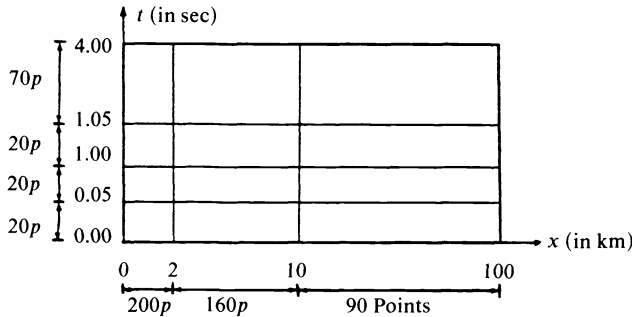


FIG. 6. Discretisation of the domain  $Q = ]0, 100[ \times ]0, 4[$ .

**5.2. The error introduced by the time-method.** We want to know if the time method proposed to solve numerically problem (P) is able to detect the polarisation phenomenon. In order to do that the polarisation patterns obtained by this method,  $p_{tN}$  ( $N$  indicates that  $\hat{\sigma}_{a_N}$  has been used), will be compared with  $p$  (see (9)) the polarisation pattern obtained with the frequency method using  $\hat{\sigma}$  (see (2)).  $p$  will be considered to be exact.

Let us define the total error  $e_{tot}(N)$ .

$$(15) \quad e_{tot}(N) = p - p_{tN}$$

In Table 1, the relative percentage  $L^1$ -errors, that is,

$$E(e) = 100(\|e\|_{L^1(0,T)}\|p\|_{L^1(0,T)}^{-1})$$

are shown.

TABLE 1  
Relative percentage  $L^1$ -errors of polarisation pattern.

$N$	$E(e_{app})$	$E(e_{tot})$	$r$
1	.56	.59	.95
2	.16	.22	.73
3	.08	.12	.67
4	.03	.12	.25

The ratio  $r$  of the approximation error to the total error decreases monotonically with increasing  $N$ . For  $N = 4$  this ratio is 0.25 showing that it is the discretisation



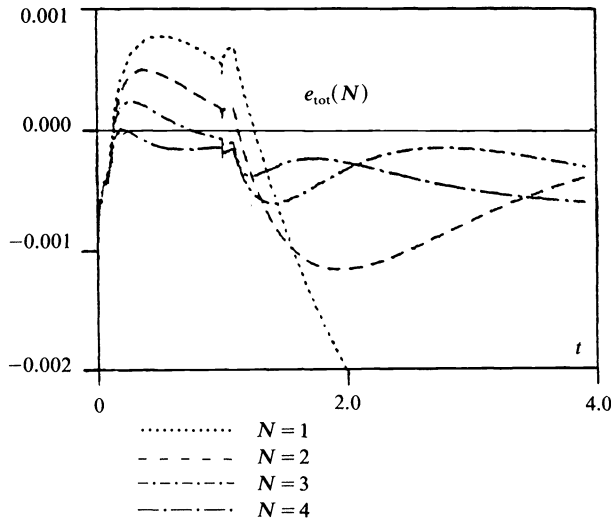


FIG 7a. Comparison of the errors in the polarisation pattern introduced by the time method.  $e_{tot}(N) = p - p_{tN}$  (see (9), (15)).

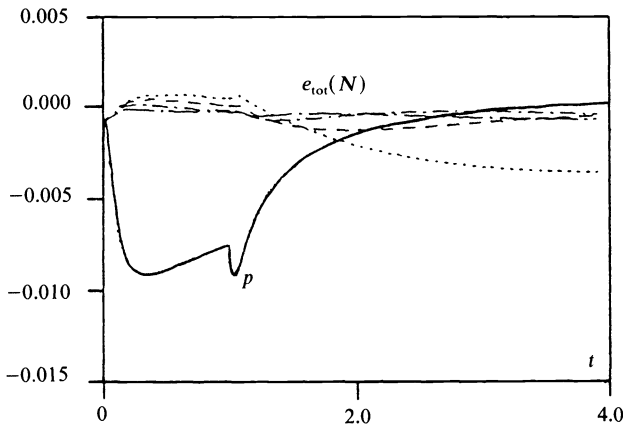


FIG 7b. Comparison of the total errors  $e_{tot}(N)$  and the exact polarisation pattern. — the exact polarisation pattern (see (9)),  $\cdots$  the total error for  $N=1$ , --- the total error for  $N=2$ , - · - · the total error for  $N=3$ , — · — the total error for  $N=4$ .

process and not the replacement of the polarisation law that has the most important influence on the total error. Moreover this error also decreases with  $N$  and is very small where the polarisation is “important”, i.e., on the interval  $[0, 2]$ . (see Fig. 7).

We point out that the cpu-time needed to compute the response of the polarisable medium (see Fig. 3) with  $N=4$  was only 5.6% greater than that needed for the same computation on the nonpolarisable medium. In other words, the introduction of the polarisation phenomenon, or equivalently the introduction of a (weak) memory, in the model does not increase significantly the cpu-time.

**6. Conclusion.** To model the electromagnetic behaviour of polarisable media at radio and lower frequencies we consider an analytical model proposed by Dias [2]. Numerical solutions can be obtained by solving the equations (1) in the frequency domain and then applying the inverse Fourier transform. Equations (1) can be solved

analytically in the one-dimensional case (and on a stratified medium), but this advantage no longer exists in the case of 2 or 3 dimensions. Instead, equations (1) are written directly in the time domain and then solved numerically. It can be seen that the polarisation phenomenon is introduced as a convolution perturbation term that makes a memory to appear in the system. To overcome the difficulty of evaluating numerically this term (in 2 or 3 dimensions) we propose a special numerical treatment which consists in replacing the polarisation law by another one of a more adequate form (see (5)).

It has been shown that the polarisation phenomenon causes only a slight perturbation in the solution of the case in which it does not appear. Despite this fact it has been proved that the replacement of the polarisation law can be done in such a way the polarisation phenomenon is not distorted significantly and that the proposed numerical method is able to detect the polarisation phenomenon well enough.

**Acknowledgments.** The author would like to thank A. Bamberger for fruitful discussions, and Y. Goldman for his help in carrying out certain computations.

#### REFERENCES

- [1] H. BREZIS, *Equations d'évolution non-linéaires en mécanique et en physique*, DEA's course notes, Paris VI, 1981.
- [2] C. A. DIAS, *Analytical model for a polarisable medium at radio and lower frequencies*, J. Geophys. Res., 77 (1972), pp. 4945, 4956.
- [3] B. COCKBURN, *Etude mathématique et numérique des équations de Maxwell dans des milieux polarisables*, 3-cycle doctoral thesis, Paris IV, Juin 1983.
- [4] ———, *Equations de Maxwell et perturbations singulières*, INRIA report, to appear.
- [5] B. COCKBURN AND P. JOLY, *Justification théorique d'une méthode de résolution des équations de Maxwell en milieu polarisable*, INRIA report to appear.
- [6] S. HARAUX, *Semi-groupes linéaires et équations d'évolution linéaires périodiques*, Lab. d'Analyse Numérique, Paris, VI, no 78011.
- [7] J. L. LIONS AND E. MAGENES, *Problèmes aux limites non homogènes et applications*, vol. 1, Dunod, Paris, 1968.
- [8] R. F. HARRINGTON, *Time-Harmonic Electromagnetic Fields*, McGraw-Hill, New York, 1961.
- [9] R. D. RICHTMYER AND K. W. MORTON, *Difference Methods for Initial Value Problems*, John Wiley, New York, 1967.

## A JACOBI-LIKE ALGORITHM FOR COMPUTING THE SCHUR DECOMPOSITION OF A NONHERMITIAN MATRIX\*

G. W. STEWART†

*Dedicated to Peter Henrici on his sixtieth birthday*

**Abstract.** This paper describes an iterative method for reducing a general matrix to upper triangular form by unitary similarity transformations. The method is similar to Jacobi's method for the symmetric eigenvalue problem in that it uses plane rotations to annihilate off-diagonal elements, and when the matrix is Hermitian it reduces to a variant of Jacobi's method. Although the method cannot compete with the QR algorithm in serial implementation, it admits of a parallel implementation in which a double sweep of the matrix can be done in time proportional to the order of the matrix.

**Key words.** eigenvalue, Schur decomposition, Jacobi algorithm, parallel algorithms

**1. Introduction.** Let  $A$  be a Hermitian matrix of order  $n$ , and let

$$(1.1) \quad \Lambda = U^H A U$$

( $\Lambda$  diagonal,  $U$  unitary) be the spectral decomposition of  $A$ . The Jacobi algorithm for computing the decomposition (1.1) is based on the following observation. Let  $a_{kl}$  ( $k < l$ ) be an off-diagonal element of  $A$ , and let

$$(1.2) \quad \begin{bmatrix} \bar{c} & \bar{s} \\ -s & c \end{bmatrix} \begin{bmatrix} a_{kk} & a_{kl} \\ a_{lk} & a_{ll} \end{bmatrix} \begin{bmatrix} c & -\bar{s} \\ s & \bar{c} \end{bmatrix} = \begin{bmatrix} \hat{a}_{kk} & 0 \\ 0 & \hat{a}_{ll} \end{bmatrix}$$

be the spectral decomposition of the  $2 \times 2$  submatrix subtended by the rows and columns  $k$  and  $l$  of  $A$  (hereafter, called the *pivot matrix*). If

$$(1.3) \quad R_{kl} = \begin{bmatrix} I_{k-1} & 0 & 0 & 0 & 0 \\ 0 & c & 0 & -\bar{s} & 0 \\ 0 & 0 & I_{l-k-1} & 0 & 0 \\ 0 & s & 0 & \bar{c} & 0 \\ 0 & 0 & 0 & 0 & I_{n-l} \end{bmatrix}$$

is the rotation in the  $(i, j)$ -plane corresponding to the transformation (1.2), then  $\hat{A} = R_{kl}^H A R_{kl}$  satisfies

$$(1.4) \quad \begin{aligned} 1. \quad & \hat{a}_{kl} = \hat{a}_{lk} = 0, \\ 2. \quad & \sum \hat{a}_{ii}^2 = \sum a_{ii}^2 + 2|a_{kl}|^2. \end{aligned}$$

Thus by annihilating the off-diagonals  $a_{kl}$  and  $a_{lk}$  the transformation  $R_{kl}$  moves  $A$  toward diagonality, in the sense that the sum of squares of the diagonal elements is increased by the squares of magnitudes of the annihilated elements.

The Jacobi algorithm consists of applying Jacobi rotations of the form (1.2) iteratively. If the sequence of pivot matrices is properly chosen, the off-diagonal elements of  $A$  converge to zero—ultimately quadratically [3], [9, Chap. 5]. Since each Jacobi rotation affects only two rows and columns of  $A$ , an entire sweep of the off-diagonal elements can be accomplished in  $O(n^3)$  arithmetic operations. In practice, the algorithm is observed to converge in a small number of sweeps, which makes it a

\* Received by the editors June 21, 1984. This research was supported by the Air Force Office of Sponsored Research under grant AFOSR-82-0078.

† Department of Computer Science, University of Maryland, College Park, Maryland 20742.

workable algorithm for the symmetric eigenvalue problem, although it is usually slower than the QR algorithm. However, Brent and Luk [1] have recently shown that the method can be implemented on a grid-connected system of  $n^2$  processors in such a way that a sweep requires  $O(n)$  time. In this environment the Jacobi method is effectively an  $O(n)$  algorithm for solving the symmetric eigenvalue problem.

The purpose of this paper is to describe a generalization of Jacobi's method to non-Hermitian matrices that also lends itself to parallel implementation. There are two paths that such a generalization might take. One is to drop the requirement that the transformations  $R_{ij}$  be unitary, while continuing to demand that  $\hat{a}_{kl} = \hat{a}_{lk} = 0$ . The prototypical algorithm of this kind is due to Eberlein [2]. When it works, it reduces  $A$  to diagonal form, and the convergence is ultimately quadratic [7]. However, the intermediate transformations can fail to exist or, what is more likely, may be very ill conditioned.

The second path, which we shall follow in this paper, is to continue to work with unitary transformations but drop the requirement that  $\hat{a}_{kl} = 0$ . The goal of the resulting iteration is the Schur decomposition

$$(1.5) \quad U^H A U = T,$$

where  $T$  is an upper triangular matrix. Although we cannot guarantee the global convergence of the method and the ultimate convergence is only linear, the fact that a sweep of the algorithm requires only  $O(n)$  time in a suitable parallel implementation makes it a serious candidate for the parallel solution of non-Hermitian eigenvalue problems.

The key to the algorithm is to work with rotations that, unlike the usual Jacobi rotations, are as far as possible from the identity matrix. Nonetheless, the algorithm reduces to a variant of the Jacobi algorithm in the symmetric case. In the next section we introduce the rotations used in the algorithm. In § 3 we describe the algorithm itself and establish its relation to the Jacobi algorithm. In § 4 a parallel version of the algorithm is described, and in § 5 its local convergence properties are examined. The paper concludes with some numerical examples and a discussion.

**2. Schur rotations.** In this section we shall derive the basic transformations used in our algorithm. For now let  $A$  be the  $2 \times 2$  matrix

$$(2.1) \quad A = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix},$$

and let  $(c \ s)^T$  be an eigenvector of  $A$  normalized to have Euclidean norm one; i.e.,

$$(2.2) \quad A \begin{bmatrix} c \\ s \end{bmatrix} = \hat{a}_{11} \begin{bmatrix} c \\ s \end{bmatrix}$$

and

$$(2.3) \quad |c|^2 + |s|^2 = 1.$$

Then the matrix

$$(2.4) \quad R = \begin{bmatrix} c & -\bar{s} \\ s & \bar{c} \end{bmatrix}$$

is unitary and

$$(2.5) \quad \hat{A} \equiv R^H A R = \begin{bmatrix} \hat{a}_{11} & \hat{a}_{12} \\ 0 & \hat{a}_{22} \end{bmatrix}$$

is upper triangular. Thus the *Schur transformation*  $R$  reduces  $A$  to Schur form.<sup>1</sup>

In general  $A$  will have two independent eigenvectors and hence two associated Schur rotations. We shall call the one for which  $|c|$  is the largest the *inner Schur rotation* and the other the *outer Schur rotation* (the inner rotation is the one nearest the identity when  $c$  is taken to be real and positive).<sup>2</sup> There are also inner and outer Jacobi rotations, and most implementations of the Jacobi method work with the former (e.g., see the elegant code of Rutishauser [8]). An essential feature of the algorithm proposed in this paper is that it uses outer Schur rotations.

Inner and outer Schur rotations are related in a way that helps explain the relation of our algorithm to the Jacobi method. Let

$$(2.6) \quad P = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}.$$

If  $(c \ s)^T$  is an eigenvector of  $A$ , then  $(s \ c)^T$  is an eigenvector of  $PAP$ . In view of (2.5), if  $R$  is an inner rotation of  $A$  then  $PR$  is an outer rotation of  $PAP$ , and vice versa. Thus the application of an outer rotation to  $A$  may be regarded as a symmetric permutation followed by the application of an inner rotation.

**3. The algorithm.** In order to motivate the algorithm, it is useful to consider a matrix that is almost in Schur form; that is, a matrix that has the form illustrated below for  $n = 4$ :

$$(3.1) \quad A = \begin{bmatrix} \times & \times & \times & \times \\ \varepsilon & \times & \times & \times \\ \varepsilon & \varepsilon & \times & \times \\ \varepsilon & \varepsilon & \varepsilon & \times \end{bmatrix}.$$

Here  $\times$  is generic for an arbitrary matrix element and  $\varepsilon$  for a small one. The idea of the algorithm is to use Schur rotations  $R_{kb}$  formed in analogy with (1.3), to eliminate  $\varepsilon$  in the  $(l, k)$  position. A *sine qua non* for such an algorithm is that it not destroy  $\varepsilon$ 's in one part of the matrix while it introduces zeros in another. This requirement restricts the rotations we are allowed to perform. For example, if the first and fourth diagonal elements of  $A$  are about  $\varepsilon$  apart, then an inner rotation  $R_{14}$  used to annihilate the  $(4, 1)$ -element will have a value of  $s$  that is of order one, and the resulting matrix  $\hat{A}$  will have the form

$$(3.2) \quad \hat{A} = \begin{bmatrix} \times & \times & \times & \times \\ \times & \times & \times & \times \\ \times & \varepsilon & \times & \times \\ 0 & \times & \times & \times \end{bmatrix}.$$

<sup>1</sup> In so calling these transformations we follow a tradition, already incipient in the literature, of naming plane rotations according to their applications. Thus a Givens rotation introduces a zero à la Givens, while a Jacobi rotation solves a symmetric  $2 \times 2$  eigenvalue problem. Note that in practice any of these rotations would be scaled so that  $c$  or  $s$  is real.

<sup>2</sup> In applications where rounding error is involved, ties are unlikely to occur. It will be assumed that when they do, they are broken in some systematic manner.

Our algorithm is based first of all on the observation that if the pivot block is taken from two contiguous diagonal elements, then no  $\epsilon$  can be destroyed. More precisely, let

$$(3.3) \quad A = E + B,$$

where  $E$  is strictly lower triangular and  $B$  is upper triangular. Let

$$(3.4) \quad \sigma = \|E\|_F \quad \text{and} \quad \tau = \|B\|_F.$$

Then the effect of using a Schur rotation in the  $(k, k + 1)$ -plane to annihilate  $a_{k+1,k}$  is to produce a matrix  $\hat{A}$  satisfying

$$(3.5) \quad \hat{\sigma}^2 = \sigma^2 - |a_{k+1,k}|^2$$

and

$$(3.6) \quad \hat{\tau}^2 = \tau^2 + |a_{k+1,k}|^2.$$

Thus a rotation on a contiguous pivot matrix shares with the Jacobi method the property that it drives the matrix toward the form to be computed [cf. (1.4)].

The foregoing suggests that we use contiguous rotations cyclicly to annihilate elements on the first subdiagonal, say in the order  $(2, 1), (3, 2), (4, 3), \dots$ . Unfortunately, if inner rotations are used the process can quickly stagnate. To see how this happens, suppose that the diagonal elements of  $A$  are well separated, so that any inner Schur rotation will have an  $s$ -value of order  $\epsilon$ . After the rotation  $R_{12}$  in the  $(1, 2)$ -plane is applied, the resulting matrix has the form

$$(3.7) \quad \begin{bmatrix} \times & \times & \times & \times \\ 0 & \times & \times & \times \\ \epsilon & \epsilon & \times & \times \\ \epsilon & \epsilon & \epsilon & \times \end{bmatrix}.$$

The next rotation has the form

$$(3.8) \quad R_{23} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \times & \epsilon & 0 \\ 0 & \epsilon & \times & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix},$$

from which it is easily seen that  $R_{23}^H A R_{23}$  has the form

$$(3.9) \quad \begin{bmatrix} \times & \times & \times & \times \\ \epsilon^2 & \times & \times & \times \\ \epsilon & 0 & \times & \times \\ \epsilon & \epsilon & \epsilon & \times \end{bmatrix}.$$

Arguing similarly, we see that after a sweep through the first subdiagonal,  $A$  has the form

$$(3.10) \quad \begin{bmatrix} \times & \times & \times & \times \\ \epsilon^2 & \times & \times & \times \\ \epsilon & \epsilon^2 & \times & \times \\ \epsilon & \epsilon & 0 & \times \end{bmatrix}.$$

After  $m - 1$  sweeps,  $A$  will be bounded by a matrix of the form

$$(3.11) \quad \begin{bmatrix} \times & \times & \times & \times \\ \varepsilon^m & \times & \times & \times \\ \varepsilon & \varepsilon^m & \times & \times \\ \varepsilon & \varepsilon & 0 & \times \end{bmatrix}.$$

In effect, the elements of the first subdiagonal, which are converging to zero, act as a barrier that keeps the rest of  $E$  from getting up to where it can be incorporated into  $B$ .

A way out of this dilemma is suggested by the fact that the nearness of the rotation (3.8) to the identity is responsible for the appearance of  $\varepsilon^2$  in (3.9); if the rotation were more balanced, it would move a larger fraction of  $\varepsilon$  in the (3, 1)-element into the (2, 1)-element. We therefore propose that outer rotations be used to annihilate elements of the first subdiagonal. If the  $\varepsilon$ 's are small, so that the diagonal elements of  $A$  approximate eigenvalues, then after one sweep through the first subdiagonal, the diagonals will appear in the order 2, 3,  $\dots$ ,  $n - 1$ , 1. This suggests that the process be repeated to move the second diagonal into position  $n - 1$ , repeated again to move the third into position  $n - 2$  and so on. This leads to the sequence of pivot matrices

$$(3.12) \quad \begin{array}{l} (1, 2), (2, 3), \dots, (n-2, n-1), (n-1, n) \\ (1, 2), (2, 3), \dots, (n-2, n-1) \\ \dots \\ (1, 2), (2, 3) \\ (1, 2) \end{array}$$

which we shall call a *forward sweep* of the matrix  $A$ .

The relation between inner and outer rotations allows us to show that the forward sweep is equivalent to a full Jacobi sweep when the matrix  $A$  is Hermitian. For  $n = 4$  let

$$(3.13) \quad S_{12}, S_{23}, S_{34}, \bar{S}_{12}, \bar{S}_{23}, \hat{S}_{12}$$

be the sequence of outer rotations in a forward sweep. By the observation at the end of § 2,  $S_{kl} = P_{kl}R_{kl}$ , where  $R_{kl}$  is an inner rotation in the  $(k, l)$ -plane. From this it can be verified directly that the result of a forward sweep is equivalent to operating on the matrix

$$(3.14) \quad \hat{P}_{12}\bar{P}_{23}\bar{P}_{12}P_{34}P_{23}P_{12}AP_{12}P_{23}P_{34}\bar{P}_{12}\bar{P}_{23}\hat{P}_{12}$$

with the sequence of inner rotations

$$(3.15) \quad \begin{array}{l} \hat{P}_{12}\bar{P}_{23}\bar{P}_{12}P_{34}P_{23}R_{12}P_{23}P_{34}\bar{P}_{12}\bar{P}_{23}\hat{P}_{12} \\ \hat{P}_{12}\bar{P}_{23}\bar{P}_{12}P_{34}R_{23}P_{34}\bar{P}_{12}\bar{P}_{23}\hat{P}_{12} \\ \hat{P}_{12}\bar{P}_{23}\bar{P}_{12}R_{34}\bar{P}_{12}\bar{P}_{23}\hat{P}_{12} \\ \hat{P}_{12}\bar{P}_{23}\bar{R}_{12}\bar{P}_{23}\hat{P}_{12} \\ \hat{P}_{12}\bar{R}_{23}\hat{P}_{12} \\ \hat{R}_{12}. \end{array}$$

The permuted matrix (3.14) has the form

$$(3.16) \quad \begin{bmatrix} \times & \varepsilon & \varepsilon & \varepsilon \\ \times^6 & \times & \varepsilon & \varepsilon \\ \times^5 & \times^4 & \times & \varepsilon \\ \times^3 & \times^2 & \times^1 & \times \end{bmatrix},$$

and the inner rotations (3.15) annihilate the elements  $\times$  in the order indicated by the superscripts in (3.16). From this it is seen that although the algorithm appears to be eliminating only elements on the first subdiagonal, it is actually sweeping the entire permuted matrix. In particular, if  $A$  is Hermitian, then one forward sweep of our algorithm is equivalent to one Jacobi sweep on the permuted matrix. This leads us to conjecture that our algorithm will perform very well on nearly Hermitian matrices, something that will be borne out by the analysis in § 5 and the examples in § 6.

Corresponding to the forward sweep, there is a *backward sweep* that restores the approximate eigenvalues to their original order. Here the rotations are performed in the order

$$(3.17) \quad \begin{aligned} &(n-1, n), (n-2, n-1), \dots, (2, 3), (1, 2) \\ &(n-1, n), (n-2, n-1), \dots, (2, 3) \\ &\dots \\ &(n-1, n), (n-2, n-1) \\ &(n-1, n). \end{aligned}$$

We shall call a forward sweep followed by a backward sweep a *double sweep*, and it is this algorithm that we shall consider in the rest of the paper. There are two reasons. First, one double sweep seems to reduce  $\sigma$  more than two forward sweeps or two backward sweeps. Second, on a grid-connected system of processors, the backward and forward sweeps mesh nicely to increase the parallelism of the algorithm, as we shall see in the next section.

**4. Parallel implementation.** A useful strategy for devising a parallel implementation of an algorithm is to fix on a suitably chosen subtask, parallelize it, and hope that the rest of the algorithm will follow along. In the double-sweep algorithm the place to look to is the diagonal of the matrix, whose elements are in effect being moved around by pairwise interchanges. Figure 4.1 exhibits a parallel implementation of these interchanges for the case  $n = 6$ . The numbers to the side are time steps, and the six numbers following them mark the current position of the original diagonal elements. A dash between two elements indicates an interchange to be performed in passing to the next time step. Forward and backward sweeps are separated by a vertical bar.

From the figure it is seen that the forward and backward sweeps can be implemented simultaneously. During the first six steps, the forward sweep propagates the first element to the last position on the diagonal and the second element to the next-to-last position, after which the backward sweep begins. The timing is such that as the first element moves backward it is met by the proper element just ending its own forward sweep. At step twelve the first element has returned to its original position, and at step thirteen it is joined by the second element, whereupon another forward sweep commences.

It is also seen from Fig. 4.1 that the rotations can be generated alternately: first for all the pivot matrices that begin with odd-numbered diagonal elements and then for those beginning with even-numbered diagonal elements. The rotations propagate



1. 1 - 2 3 4 5 6
2. 2 1 - 3 4 5 6
3. 2 - 3 1 - 4 5 6
4. 3 2 - 4 1 - 5 6
5. 3 - 4 2 - 5 1 - 6
6. 4 3 - 5 2 - 6 | 1
7. 4 - 5 3 - 6 | 2 - 1
8. 5 4 - 6 | 3 - 1 2
9. 5 - 6 | 4 - 1 3 - 2
10. 6 | 5 - 1 4 - 2 3
11. 6 - 1 5 - 2 4 - 3
12. 1 | 6 - 2 5 - 3 4
13. 1 - 2 | 6 - 3 5 - 4
14. 2 1 - 3 | 6 - 4 5
15. 2 - 3 1 - 4 | 5 - 6

FIG. 4.1. Parallel implementation of diagonal interchanges.

through the matrix as shown in Fig. 4.2. The first matrix in the sequence shows the rotations being generated in the odd pivot matrices (the blocks labeled with a 1). Each block passes its rotations north, south, east and west. Where (in blocks also labeled 1) two rotations meet in the second matrix, the rotations are applied and then passed on to the blocks next outward in the third matrix. At this point the elements of the even pivot matrices will not be further altered by the rotations labeled 1. Hence their rotations (labeled 2) can be generated. These rotations fan out following the first set until in the fifth matrix they have moved far enough to allow the rotations for the odd pivot matrices to be generated (labeled 3). The process now continues in an obvious manner.

- |   |   |
|---|---|
| <ol style="list-style-type: none"> <li>1. 1 1 × × × × × ×<br/>             1 1 × × × × × ×<br/>             × × 1 1 × × × ×<br/>             × × 1 1 × × × ×<br/>             × × × × 1 1 × ×<br/>             × × × × 1 1 × ×<br/>             × × × × × × 1 1<br/>             × × × × × × 1 1</li> <li>3. × × × × 1 1 × ×<br/>             × 2 2 × 1 1 × ×<br/>             × 2 2 × × × 1 1<br/>             × × × 2 2 × 1 1<br/>             1 1 × 2 2 × × ×<br/>             1 1 × × × 2 2 ×<br/>             × × 1 1 × 2 2 ×<br/>             × × 1 1 × × × ×</li> <li>5. 3 3 × 2 2 × × ×<br/>             3 3 × × × 2 2 ×<br/>             × × 3 3 × 2 2 ×<br/>             2 × 3 3 × × × 2<br/>             2 × × × 3 3 × 2<br/>             × 2 2 × 3 3 × ×<br/>             × 2 2 × × × 3 3<br/>             × × × 2 2 × 3 3</li> </ol> | <ol style="list-style-type: none"> <li>2. × × 1 1 × × × ×<br/>             × × 1 1 × × × ×<br/>             1 1 × × 1 1 × ×<br/>             1 1 × × 1 1 × ×<br/>             × × 1 1 × × 1 1<br/>             × × 1 1 × × 1 1<br/>             × × × × 1 1 × ×<br/>             × × × × 1 1 × ×</li> <li>4. × 2 2 × × × 1 1<br/>             2 × × 2 2 × 1 1<br/>             2 × × 2 2 × × ×<br/>             × 2 2 × × 2 2 ×<br/>             × 2 2 × × 2 2 ×<br/>             × × × 2 2 × × 2<br/>             1 1 × 2 2 × × 2<br/>             1 1 × × × 2 2 ×</li> <li>6. × × 3 3 × 2 2 ×<br/>             × × 3 3 × × × 2<br/>             3 3 × × 3 3 × 2<br/>             3 3 × × 3 3 × ×<br/>             × × 3 3 × × 3 3<br/>             2 × 3 3 × × 3 3<br/>             2 × × × 3 3 × ×<br/>             × 2 2 × 3 3 × ×</li> </ol> |
|---|---|

FIG. 4.2. Propagation of the rotations.

In a parallel implementation of the algorithm, it is natural to associate a processor with each  $2 \times 2$  block of contiguous elements. In Fig. 4.3 the processors associated with the block beginning with  $a_{IJ}$  are labeled  $(I, J)$ . These are connected horizontally and vertically in staggered grids to allow the rotations to pass through the matrix. After a processor has finished generating or applying a rotation, it passes the elements of its block (denoted by  $\times$  in the figure) diagonally to its four neighboring processors, so that they can apply their rotations. Synchronization is by data flow; a processor may compute any time it has all its elements and rotations. For more details, see [6].

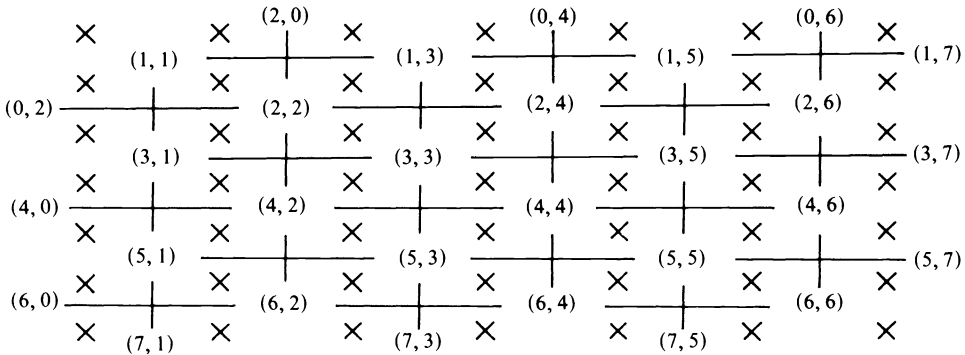


FIG. 4.3. Computational network for the Jacobi-Schur algorithm.

Each cluster of rotations requires  $O(n)$  time to pass completely out of the matrix, and since the clusters follow each other at intervals of two time steps, the realization of a double sweep will also require only  $O(n)$  time. Since the algorithm reduces to a variant of Jacobi's method when  $A$  is Hermitian, it can be regarded as a  $O(n)$  implementation of that method—however, one that is different from the implementation of Brent and Luk [1].

**5. Asymptotic properties.** As was indicated in the introduction it is not possible to establish a global convergence theorem for the algorithm proposed in this paper; indeed, as we shall see later in this section, there are matrices that are not in Schur form for which the algorithm is stationary. However, we are able to analyze the behavior of the algorithm under the assumption that the matrix  $E$  of (3.3) is small, and this analysis sheds some light on its properties.

We shall assume that the eigenvalues of  $A$  are distinct. This implies that the upper triangular matrix  $T$  in the Schur decomposition is uniquely determined up to diagonal unitary similarities by the order of the eigenvalues on the diagonal of  $T$ . Moreover, if  $\bar{S}_{12}, \bar{S}_{23}, \dots$  are the outer rotations resulting from one double sweep of  $T$ , then they are differentiable functions of the elements of  $T$ . It follows that if  $\sigma$  in (3.4) is sufficiently small, then the sequence  $S_{12}, S_{23}, \dots$  of rotations for  $A$  satisfies

$$(5.1) \quad S_{kl} = \bar{S}_{kl} + O(\sigma).$$

Now let

$$(5.2) \quad A_1 \equiv S_{12}^H A S_{12} = E_1 + B_1,$$

where  $E_1$  is strictly lower triangular. If  $P_{kl}$  is the operator that sets the  $(l, k)$ -element of a matrix to zero, then it is easy to verify that

$$(5.3) \quad E_1 = S_{12}^H P_{12}(E) S_{12}.$$

In view of (5.2),

$$(5.4) \quad E_1 = \bar{S}_{12}^H \mathbf{P}_{12}(E) \bar{S}_{12} + O(\sigma^2).$$

Similarly, if

$$(5.5) \quad A_2 \equiv S_{23}^H A S_{23} = E_2 + B_2,$$

then

$$(5.6) \quad E_2 = \bar{S}_{23}^H \mathbf{P}_{23}(E) \bar{S}_{23} + O(\sigma^2).$$

Proceeding in this manner, we find that the error matrix  $E$ , resulting from a full double sweep can be written in the form

$$(5.7) \quad E_* = \mathbf{S}(E) + O(\sigma^2),$$

where  $\mathbf{S}$  is a linear operator on the space of strictly lower triangular matrices (normed by the Frobenius norm), which is composition of  $m$  pairs of unitary similarity transformations and operators of the form  $\mathbf{P}_{k,k+1}$ . If  $0 < \|\mathbf{S}\| < 1$ , where  $\|\cdot\|$  is the operator norm, then (5.7) implies that the iteration converges at a rate at least as fast as the convergence of  $\|\mathbf{S}\|^k$ . If  $\|\mathbf{S}\| = 0$ , then the convergence is quadratic.

It is easy to show that  $\|\mathbf{S}\|$  is not greater than one. For the unitary similarity transformations that are part of  $\mathbf{S}$  are also unitary transformations in the space of  $n \times n$  matrices, since they do not change the Frobenius norm of a matrix. The operators  $\mathbf{P}_{kl}$  are orthogonal projections, since they simply set elements to zero. Thus  $\mathbf{S}$  is the product of unitary transformations and orthogonal projections, and its norm is therefore bounded by one.

The exact value of  $\|\mathbf{S}\|$  will depend on the matrix  $T$ . It is instructive to examine two boundary cases: one in which  $\|\mathbf{S}\| = 1$ ; and one in which  $\|\mathbf{S}\| = 0$ .

The first case occurs when  $T$  has the form illustrated below for  $n = 5$ :

$$(5.8) \quad T = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}.$$

It is easily verified that the iteration is stationary for any matrix of the form

$$(5.9) \quad T = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 \\ \times & 0 & 1 & 1 & 0 \\ \times & \times & 0 & 1 & 1 \\ \times & \times & \times & 0 & 1 \end{bmatrix}.$$

In other words  $\mathbf{S}$  leaves invariant any strictly lower triangular matrix  $E$ , which implies that  $\|\mathbf{S}\| = 1$ .

On the other hand, it can be verified that whenever the transformations  $S_{k,k+1}$  are permutations, then  $\|\mathbf{S}\| = 0$ . This will happen when

$$(5.10) \quad T = \text{diag}(\lambda_1, \lambda_2, \dots, \lambda_n),$$

where the  $\lambda_i$  are distinct. This happens precisely when  $A$  is normal with distinct eigenvalues, and it follows that if our method converges at all for such a matrix, the

ultimate convergence must be quadratic. Since multiple eigenvalues do not seem to prevent the quadratic convergence of the Jacobi algorithm, there is a strong presumption that a closer analysis will establish the local quadratic convergence of our algorithm for all normal matrices.

The matrix  $T$  of (5.8), being a Jordan block of order  $n$ , is about as defective as a matrix can be. On the other hand, the diagonal matrix  $T$  of (5.10) lies at the extreme of nondefectiveness. This suggests that our algorithms will tend to perform well in proportion as the matrix in question is nondefective. In the next section, we shall see that the matter is not that simple, but some insight into how defectiveness enters may be seen by examining the outer Schur rotations  $\tilde{S}_{k,t}$ .

Consider, for example, the leading  $2 \times 2$  submatrix of  $T$ . Its outer rotation will be determined by the eigenvector  $(1 \ t)^T$  satisfying

$$(5.11) \quad \begin{bmatrix} a_{11} & a_{12} \\ 0 & a_{22} \end{bmatrix} \begin{bmatrix} 1 \\ t \end{bmatrix} = a_{22} \begin{bmatrix} 1 \\ t \end{bmatrix}.$$

From (5.11) it follows that

$$(5.12) \quad t = \frac{a_{22} - a_{11}}{a_{12}}.$$

The number  $t^{-1}$  may be taken as a measure of the departure of the matrix

$$(5.13) \quad \begin{bmatrix} a_{11} & a_{12} \\ 0 & a_{22} \end{bmatrix}$$

from normality<sup>3</sup>: if it is small, then (5.13) is near, in a relative sense, to a normal matrix; if it is large, to a defective matrix. Thus when (5.13) is nearly normal it tends to produce permutations, which we have seen corresponds to quadratic convergence. On the other hand, when it is nearly defective, it tends to produce identity matrices, which are associated with stationarity.

**6. Examples and discussion.** In this section we shall present the results of some numerical experiments performed with the double sweep algorithm. The reader should keep in mind that the examples are intended primarily to illustrate the considerations of § 5 and not as a broad justification of the algorithm itself. The latter can only be accomplished by testing the algorithm in parallel implementation on a wide variety of real-life examples.

The examples in this section were generated in 64-bit hexadecimal arithmetic using the MATLAB system of C. B. Moler [5]. The first sequence shows the effects of departure from normality on the convergence of the method. The matrices in these examples have the form

$$(6.1) \quad A = U(D + \alpha F)U^H,$$

where

$$(6.2) \quad D = \text{diag}(1, 2, 3, 4, 5),$$

$$(6.3) \quad U = \begin{bmatrix} -.3627 & .0216 & -.1466 & -.9195 & .0316 \\ .7020 & -.1795 & .5282 & -.3735 & -.2378 \\ .4062 & .7709 & -.4325 & -.0806 & -.2169 \\ .3440 & .0179 & -.0949 & -.0882 & .9298 \\ .3039 & -.6105 & -.7095 & -.0271 & -.1756 \end{bmatrix}$$

<sup>3</sup> It is closely related to the measure introduced by Henrici in [4].

and

$$(6.4) \quad F = \begin{bmatrix} 0.0000 & .3269 & .0614 & -.1891 & .4250 \\ 0.0000 & 0.0000 & -1.7158 & .1023 & -1.2859 \\ 0.0000 & 0.0000 & 0.0000 & -.4575 & -.6453 \\ 0.0000 & 0.0000 & 0.0000 & 0.0000 & .9182 \\ 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 \end{bmatrix}.$$

Table 6.1 shows the convergence of the algorithm for various values of  $\alpha$ . The columns labeled  $\sigma$  contain the Frobenius norm of  $E$  after the double sweep numbered in the first column. The columns labeled  $\rho$  contain the ratios of the current sigma to the previous one—an estimate of the rate of linear convergence.

TABLE 6.1  
Convergence and normality.

SWEEP	$\alpha$							
	0.01		0.1		1.0		10.0	
	$\sigma$	$\rho$	$\sigma$	$\rho$	$\sigma$	$\rho$	$\sigma$	$\rho$
0	1.7e+00	—	1.7e+00	—	2.1e+00	—	1.5e+01	—
1	3.0e-02	1.8e-02	5.9e-02	3.5e-02	3.3e-01	1.6e-01	7.3e-01	4.8e-02
2	4.2e-10	1.4e-08	1.3e-05	2.2e-04	1.4e-02	4.2e-02	1.9e-01	2.6e-01
3	—	—	3.3e-09	2.5e-04	1.9e-03	1.4e-01	1.0e-01	5.5e-01
4	—	—	8.2e-13	2.5e-04	2.9e-04	1.5e-01	7.1e-02	6.8e-01
5	—	—	—	—	4.4e-05	1.5e-01	5.3e-02	7.5e-01
6	—	—	—	—	6.6e-06	1.5e-01	4.3e-02	8.0e-01
7	—	—	—	—	9.9e-07	1.5e-01	3.6e-02	8.4e-01
8	—	—	—	—	1.5e-07	1.5e-01	3.1e-02	8.7e-01
9	—	—	—	—	2.2e-08	1.5e-01	2.8e-02	8.9e-01

The first example, with  $\alpha = 0.01$ , almost exhibits the quadratic convergence typical of the Jacobi algorithm for symmetric matrices. The third iterate (not shown) is well below the level of rounding error. The convergence of the second example ( $\alpha = 0.1$ ) is quite satisfactory, even though the matrix deviates quite a bit from symmetry. The numbers  $\rho$  settle down to  $2.5 \cdot 10^{-4}$ , demonstrating the linear convergence established in § 5. The convergence is also satisfactory for the decidedly nonsymmetric matrix in the third example ( $\alpha = 1.0$ ), or at least satisfactory for a parallel implementation. It is linear with a ration of about 0.15. Only when  $\alpha = 10.0$  does the convergence become intolerably slow. Note that this is not a nice matrix: a perturbation of order 0.015 can make it defective.

The last example is intended to demonstrate that defectiveness alone is not necessarily disastrous. The matrix  $A$  is generated according to (6.1) but with

$$(6.5) \quad D = \text{diag}(1, 2, 3, 3, 4)$$

and  $\alpha = 0.1$ . Although the matrix has a nonlinear elementary divisor corresponding to the eigenvalue three, the convergence shown in Table 6.2 is quite satisfactory, being only seven times slower than the corresponding nondefective matrix in Table 6.1.

Since the algorithm proposed in this paper is a variant of the Jacobi algorithm, it cannot compete with the QR algorithm in a serial implementation. However, the fact that a full double sweep can be implemented in  $O(n)$  time on a grid-connected

TABLE 6.2  
A defective matrix.

SWEEP	$\alpha$	
	0.1	
	$\sigma$	$\rho$
0	1.3e+00	—
1	9.8e-02	7.4e-02
2	9.5e-05	9.7e-04
3	1.8e-07	1.9e-03
4	3.4e-10	1.9e-03
5	6.2e-13	1.8e-03

system of processors makes it a serious candidate for the parallel solution of eigenvalue problems. It has the advantage over algorithms that seek a diagonal form that it is very stable; and it shares with them the drawback that it must be implemented in complex arithmetic, even when the original matrix is real. The major objection to the algorithm is its slow convergence for strongly nonnormal matrices. However, the QR algorithm itself converges slowly for defective matrices, and, as we noted in the introduction, the Eberlein-like algorithms tend to produce ill-conditioned transformations. The choices among the alternatives will become clear only with further experimentation.

## REFERENCES

- [1] R. P. BRENT AND F. T. LUK, *A systolic architecture for almost linear-time solution of the symmetric eigenvalue problem*, Tech. Rep. TR-CS-82-525, Dept. Computer Science, Cornell Univ., Ithaca, NY 1982.
- [2] P. J. EBERLEIN, *A Jacobi-like method for the automatic computation of eigenvalues and eigenvectors of an arbitrary matrix*, SIAM J. Appl. Math., 10 (1962), pp. 74-88.
- [3] P. HENRICI, *On the speed of convergence of cyclic and quasicyclic Jacobi methods for computing eigenvalues of Hermitian matrices*, J. Soc. Indust. Appl. Math., 6 (1958), pp. 144-162.
- [4] ———, *Bounds for iterates, inverses, spectral variation and fields of values of nonnormal matrices*, Numer. Math., 4 (1962), pp. 24-40.
- [5] C. B. MOLER, *MATLAB Users' Guide*, Dept. Computer Science, Univ. New Mexico, Albuquerque, 1981.
- [6] D. P. O'LEARY AND G. W. STEWART, *Data-flow algorithms for parallel matrix computations*, Computer Science Tech. Rep. 1366, Univ. Maryland, College Park, 1984.
- [7] A. RUHE, *On the quadratic convergence of a generalization of the Jacobi method to arbitrary matrices*, BIT, 8 (1968), pp. 210-231.
- [8] H. RUTISHAUSER, *The Jacobi method for real symmetric matrices*, Numer. Math., 9 (1966), pp. 1-10.
- [9] J. H. WILKINSON, *The Algebraic Eigenvalue Problem*, Oxford Univ. Press, Oxford, 1965.

## PRACTICAL USE OF POLYNOMIAL PRECONDITIONINGS FOR THE CONJUGATE GRADIENT METHOD\*

YOUCEF SAAD†

**Abstract.** This paper presents some practical ways of using polynomial preconditions for solving large sparse linear systems of equations issued from discretizations of partial differential equations. For a symmetric positive definite matrix  $A$  these techniques are based on least squares polynomials on the interval  $[0, b]$ , where  $b$  is the Gershgorin estimate of the largest eigenvalue. Therefore, as opposed to previous work in the field, there is no need for computing eigenvalues of  $A$ . We formulate a version of the conjugate gradient algorithm that is more suitable for parallel architectures and discuss the advantages of polynomial preconditioning in the context of these architectures.

**Key words.** conjugate gradient method, polynomial preconditionings, parallel algorithms, vectorization

**1. Introduction.** When combined with a suitable preconditioning, the conjugate gradient method constitutes one of the most powerful techniques for solving large sparse symmetric positive definite linear systems of equations. However, most of the preconditionings have originally been designed for scalar computers and much work must be done to reformulate them or develop new ones that are more suitable for the new generation of computers. One attractive possibility considered by several authors [1], [3], [5], [8], [10], [19] is the use of polynomial preconditionings. Given a symmetric system  $Ax = f$ , the principle of polynomial preconditioning, which goes back to Rutishauser [16], consists in solving the (preconditioned) linear system  $s(A)Ax = s(A)f$ , where  $s$  is some polynomial, usually of low degree. The polynomial  $s$  is chosen so that the matrix  $s(A)A$  has an eigenvalue distribution that is favorable to the conjugate gradient method, i.e. so that the conjugate gradient method applied to the preconditioned system converges rapidly. In the classical context of scalar computers, there is little reason for using polynomial preconditionings because the conjugate gradient method is an optimal process and the total number of matrix by vector multiplications required by the conjugate gradient method applied to the preconditioned system  $s(A)Ax = s(A)f$  will be higher than that of the nonpreconditioned system  $Ax = f$ . The loss incurred by a larger number of matrix-vector multiplications may in some cases be offset by the smaller number of inner products required when polynomial preconditioning is used but the overall performance is not likely to be much different in those cases. Moreover, incomplete factorization based preconditionings are then quite effective, and are usually preferred to polynomial preconditionings.

Jordan [10] reports some experiments on the CRAY-1 showing that polynomial preconditionings are competitive on vector computers. He concludes that more work is needed to simplify the derivation of "good" polynomials. One of our main goals is to propose a class of effective polynomials that are easy to derive and that do not require eigenvalue estimates.

In addition to their importance for vector computers polynomial preconditionings are attractive for parallel architectures. As will be seen, when  $A$  is block tridiagonal, a great deal of parallelism can be achieved when computing  $s(A)Av$ . A few ideas describing how to take advantage of polynomial preconditionings in this particular context will be seen in § 3.

---

\* Received by the editors December 28, 1983, and in final revised form August 1, 1984. 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-81-06181.

† Computer Science Department, Yale University, New Haven, Connecticut 06520.

In the first part of this paper we focus on the problem of choosing a good polynomial  $s$ . The classical choice is to take  $s(\lambda)$  so that the residual polynomial  $R(\lambda) \equiv 1 - \lambda s(\lambda)$  minimizes  $\|R(\lambda)\|_\infty$  over all polynomials  $R$  of degree not exceeding  $k$  so that  $R(0) = 1$ , where  $\|\cdot\|_\infty$  is the infinity norm on some interval  $[a, b]$  containing the spectrum of  $A$ , with  $0 < a < b$  [16]. This leads to the well-known Chebyshev iteration. In this paper we suggest using a polynomial  $s$  that minimizes the  $L_2$ -norm  $\|R(\lambda)\|_w$  with respect to some weight function  $w$  defined on an interval  $[a, b]$  that contains the spectrum of  $A$ . The idea of using the  $L_2$ -norm instead of the infinity norm goes back to Stiefel [20] and was recently suggested for polynomial preconditioning by Johnson, Micchelli and Paul [8]. An important observation made by Stiefel is that  $a$  and  $b$  need not be accurate estimates of the smallest and the largest eigenvalues of  $A$  as long as  $[a, b]$  contains the spectrum of  $A$ . Indeed, while it is necessary for Chebyshev iteration that  $0 < a < b$ , for the  $L_2$ -norm there is no such restriction and one can simply use the interval which is provided by Gershgorin's theorem. For example, we may use an interval of the form  $[0, b]$  when it is known that  $A$  is positive definite and that  $b$  is an upper bound for the largest eigenvalue provided e.g. by Gershgorin's theorem. The Gershgorin bounds can be obtained as the matrix  $A$  is built and therefore there is no need for an adaptive scheme which may considerably slow down the flow of computations in vector or parallel machines.

One might ask whether we lose efficiency when using least squares polynomials instead of Chebyshev polynomials since we require less information, i.e. since we require Gershgorin values instead of eigenvalues. In fact, an interesting revelation from the numerical experiments is that the usual optimum parameters  $a = \lambda_1 \equiv$  the smallest eigenvalue of  $A$  and  $b = \lambda_N \equiv$  the largest eigenvalue of  $A$ , used in the Chebyshev iteration, *do not in general minimize the total number of conjugate gradient iterations required for convergence*. If these values were used, then our experiments show that the least squares polynomial approach performs better, not worse as might have been expected, than the more complicated Chebyshev polynomial approach. The above optimal parameters are known to minimize the condition number of the iteration matrix  $As(A)$ , as was proven by Johnson, Micchelli and Paul [8]. However, the condition number does not matter as much as the overall distribution of eigenvalues. For this reason it is clear that it will be difficult to compute the best parameters, i.e. the parameters  $a$  and  $b$  that maximize the rate of convergence. This constitutes the main drawback of Chebyshev polynomial preconditioning. The numerical experiments indicate that the least squares polynomials perform quite well in spite of the fact that they do not require eigenvalue estimates.

The second part of this paper will discuss the practical implementation of polynomial preconditionings in parallel computation. Although there are many possible implementations, our idea rests on the simple principle, which is not completely new, that when  $A$  is block tridiagonal then one can perform the products  $Av, A^2v, \dots, A^kv$  concurrently.

## 2. Polynomial iteration and polynomial preconditioning.

### 2.1. Basic theory. Consider the linear system

$$(1) \quad Ax = f,$$

where  $A$  is symmetric positive definite. An efficient technique for solving (1) is the conjugate gradient method applied to the preconditioned system

$$(2) \quad Q^{-1}Ax = Q^{-1}f,$$



where  $Q^{-1}$  is some approximate inverse of  $A$ , for which systems of the form  $Qy = z$  are easy to solve.

The matrix  $Q$  is referred to as the preconditioning matrix and a classical example is the incomplete Choleski factorization of  $A$  [12]. Although the preconditioned system (2) is no longer symmetric, symmetry can be recovered by using the inner product  $(x, y)_Q = (Qx, y)$  instead of the Euclidean inner product in the conjugate gradient method [12]. It is assumed that  $Q^{-1}A$  is self-adjoint and positive definite for this inner product.

Incomplete factorization preconditionings are very powerful techniques in scalar machines but do not vectorize well and may not be the best choice for supercomputers. Some newly developed preconditionings do, however, vectorize fairly well. Among them let us mention the vectorizable version of the incomplete Choleski factorization proposed by Van der Vorst [21]. In [10] a few ways of adapting classical preconditioners to vector and parallel computers are compared.

Several authors have suggested using polynomial preconditionings, that is taking  $Q^{-1} \equiv s(A)$ , where  $s$  is some polynomial; see [5], [8], [10]. The simplest such polynomial suggested by Dubois et al. [5], is given by the Neuman series

$$s(A) = I + N + N^2 + \dots + N^k$$

where  $N = I - A$  is assumed to be such that  $\|N\| \leq 1$ , which is often verified.

More efficient polynomials can be obtained if one knows good estimates  $a$  and  $b$  of the smallest and the largest eigenvalues of  $A$ . Indeed, let  $\sigma(A) = \{\lambda_i\}_{i=1, \dots, N}$  be the spectrum of  $A$  with  $\lambda_1 \leq \lambda_2 \leq \dots \leq \lambda_N$ . Let  $s(\lambda)$  be any polynomial of degree not exceeding  $k - 1$  and consider the matrix  $Q^{-1}A = s(A)A$  of the preconditioned system (2). The purpose of the preconditioning is to transform the eigenvalue distribution into one that is more favorable to the conjugate gradient method. For example, we could choose the polynomial  $s$  so that  $\|I - s(A)A\|$  is minimized, where  $\|\cdot\|$  represents the 2-norm. Noticing that

$$(3) \quad \|I - s(A)A\| = \max_{\lambda_i \in \sigma(A)} |1 - \lambda_i s(\lambda_i)|,$$

it is clear that a good polynomial  $s(\lambda)$  would be one for which

$$(4) \quad \max_{\lambda \in [a, b]} |1 - \lambda s(\lambda)|,$$

is minimal over all polynomials of degree  $\leq k - 1$ , where  $[a, b]$  is an interval that contains  $\sigma(A)$  with  $0 < a < b$ . It is then well known that the best such polynomial is such that  $1 - \lambda s(\lambda)$  is an appropriately scaled and shifted Chebyshev polynomial of degree  $k$  of the first kind, see [2]. This constitutes the foundation of Chebyshev iteration [6] and was also considered for polynomial preconditioning [8], [10]. In fact it can be shown [8] that when  $a = \lambda_1$  and  $b = \lambda_N$ , the resulting preconditioned matrix minimizes the condition number of the preconditioned matrices of the form  $As(A)$  over all polynomials  $s$  of degree  $\leq k - 1$ . However, an interesting numerical observation is that when used in conjunction with the conjugate gradient method the best polynomial, i.e. the one which minimizes the total number of conjugate gradient iterations *is far from being the one that minimizes the condition number*. The behaviour of the polynomial preconditioned conjugate gradient is not simple to analyse. Thus, if instead of taking  $a = \lambda_1$  and  $b = \lambda_N$  we took  $[a, b]$  to be slightly inside the interval  $[\lambda_1, \lambda_N]$ , we would achieve faster convergence in general. Unfortunately, the true optimal parameters, i.e. those that minimize the number of iterations of the polynomial preconditioned conjugate gradient method, are not known and we do not know of any way to obtain them.

Several authors have also considered instead of (4) a  $L_2$ -norm over the interval  $[a, b]$  with  $a = \lambda_1, b = \lambda_N$  with respect to some weight function  $w$  [8], [17], [18], [20]. Johnson, Micchelli and Paul [8] have experimentally shown that the resulting preconditionings may lead to faster convergence than the Chebyshev based ones.

A further disadvantage of the approaches described above is that the parameters  $a$  and  $b$  which approximate the smallest and largest eigenvalues of  $A$  are usually not available beforehand and must be obtained in some dynamic way. This slows down the process and makes it unattractive for supercomputers where one should avoid halting the flow of computation.

In order to overcome the difficulty of computing the parameters  $a$  and  $b$ , Stiefel suggested using for  $a$  and  $b$  the values provided by an application of Gershgorin's theorem. Thus, the parameter  $a$  which estimates the smallest eigenvalue of  $A$  may be nonpositive even when  $A$  is a positive definite matrix. However, when  $a \leq 0$  the problem of minimizing (4) is not well defined, i.e. it does not have a unique solution. This is due to the nonstrict-convexity of the uniform norm. Stiefel then suggests using the  $L_2$ -norm on  $[a, b]$  with respect to some weight function  $w(\lambda)$ .

Consider the inner product on the space  $P_k$  of polynomials of degree not exceeding  $k$ :

$$(5) \quad \langle p, q \rangle = \int_a^k p(\lambda)q(\lambda)w(\lambda) d\lambda$$

where  $w(\lambda)$  is some nonnegative weight function on  $(a, b)$ . We will denote by  $\|p\|_w$  and call  $w$ -norm the 2-norm induced by this inner product.

From the above discussion, we seek the polynomial  $s_{k-1}(\lambda)$  which minimizes

$$(6) \quad \|1 - s(\lambda)\|_w$$

over all polynomials  $s$  of degree  $\leq k - 1$ . We will call  $s_{k-1}$  the least squares iteration polynomial, or simply the least squares polynomial and will refer to  $R_k(\lambda) \equiv 1 - \lambda s_{k-1}(\lambda)$  as the least squares residual polynomial. A crucial observation made by Stiefel is that, as opposed to the classical approach using the infinity norm, the least squares polynomial is now well defined for arbitrary values of  $a$  and  $b$ . As will be shown,  $s_{k-1}(A)$  will constitute a good approximation to  $A^{-1}$  as the degree  $k - 1$  increases. Computing the polynomial  $s_{k-1}(\lambda)$  is not a difficult task when the weight function  $w$  is suitably chosen. This will constitute the object of the next section.

**2.2. Computation of the least squares polynomials.** There are at least three ways of computing the least squares polynomial defined in the previous section:

1. By using the kernel polynomials formula [20]:

$$(7) \quad R_k(\lambda) = \left[ \sum_{i=0}^k q_i(0)q_i(\lambda) \right] / \left[ \sum_{i=0}^k q_i(0)^2 \right]$$

in which the  $q_i$ 's represent a sequence of polynomials orthogonal with respect to the weight function  $w(\lambda)$ .

2. By generating a three term recurrence satisfied by the residual polynomials  $R_k(\lambda)$  [20]. Indeed, it is known that the residual polynomials are orthogonal with respect to the new weight function  $\lambda w(\lambda)$ . For more details see Stiefel [20].

3. By solving the corresponding normal equations [17]:

$$(8) \quad \langle 1 - \lambda s_{k-1}(\lambda), \lambda Q_j(\lambda) \rangle = 0, \quad j = 0, 1, 2, \dots, k - 1$$

where  $Q_j, j = 1, \dots, k - 1$  is any basis of the space  $P_{k-1}$  of polynomials of degree  $\leq k - 1$ . See [17] for the treatment of a similar problem in a slightly more general context.

Each of these three approaches is useful in a different context. Approach 1 is general and is useful for computing explicitly least squares polynomials of low degree. For high degree polynomials the last two approaches are to be preferred for numerical stability. Approach number 2 is restricted to the case where  $a \geq 0$ , while approach number 3 is more general. We should point out that the degrees of polynomial preconditioners are often low, e.g. not exceeding 5 or 10 so we will describe the first formulation in more detail.

Let  $q_i(\lambda)$ ,  $i = 0, 1, \dots, n, \dots$  be the *orthogonal* polynomials with respect to  $w(\lambda)$ . It is known that the least squares residual polynomial  $R_k(\lambda)$  of degree  $k$  is determined by the kernel polynomials formula (7). To get  $s_{k-1}(\lambda)$  simply notice that

$$s_{k-1}(\lambda) = (1 - R_k(\lambda))/\lambda = \left[ \sum_{i=0}^k q_i(0)t_i(\lambda) \right] / \left[ \sum_{i=0}^k q_i(0)^2 \right]$$

with

$$t_i(\lambda) = (q_i(0) - q_i(\lambda))/\lambda$$

which allows one to compute  $s_{k-1}$  as a linear combination of the polynomials  $t_i(\lambda)$ . Thus from the orthogonal polynomials  $q_i$  one can compute the desired least squares polynomials. The polynomials  $q_i$  satisfy a three term recurrence of the form

$$\beta_{i+1}q_{i+1}(\lambda) = (\lambda - \alpha_i)q_i(\lambda) - \beta_iq_{i-1}(\lambda), \quad i = 1, 2, \dots,$$

from which we derive the following recurrence for the  $t_i$ 's

$$\beta_{i+1}t_{i+1}(\lambda) = (\lambda - \alpha_i)t_i(\lambda) - \beta_it_{i-1}(\lambda) + q_i(0), \quad i = 1, 2, \dots$$

The weight function  $w$  will be chosen so that the three term recurrence of the orthogonal polynomials  $q_i$  is explicitly known and/or is easy to generate. One interesting class of weight functions that satisfy this requirement is considered next.

**2.3. Choice of the weight functions.** In this section we assume that  $a = 0$  and  $b = 1$ . Consider the Jacobi weights

$$(9) \quad w(\lambda) = \lambda^{\alpha-1}(1-\lambda)^\beta \quad \text{where } \alpha > 0 \quad \text{and} \quad \beta \geq -\frac{1}{2}.$$

For these weight functions, the recurrence relations are explicitly known for the polynomials that are orthogonal with respect to  $w(\lambda)$ ,  $\lambda w(\lambda)$  or  $\lambda^2 w(\lambda)$ , thus allowing the use of any of the three methods described in the previous section for computing  $s_{k-1}(\lambda)$ . Moreover, from a result of [8], the matrix  $As_k(A)$  is known to be positive definite when  $A$  is positive definite and  $\alpha - 1 \geq \beta \geq -\frac{1}{2}$ .

The following explicit formula for  $R_k(\lambda)$  can easily be derived from the explicit expression of the Jacobi polynomials [4] and the fact that  $\{R_k\}$  is orthogonal with respect to the weight  $\lambda w(\lambda)$ :

$$(10) \quad R_k(\lambda) = \sum_{j=0}^k \kappa_j^{(k)} (1-\lambda)^{k-j} (-\lambda)^j \quad \text{where} \quad \kappa_j^{(k)} = \binom{k}{j} \prod_{i=0}^{j-1} \frac{k-i+\beta}{i+1+\alpha}.$$

From (1) it is easy to derive the polynomial  $s_{k-1}(\lambda) = (1 - R_k(\lambda))/\lambda$  "by hand" for small degrees.

As an example, when  $\alpha = \frac{1}{2}$  and  $\beta = -\frac{1}{2}$  we get the following first four polynomials:

$$\begin{aligned} s_0(\lambda) &= \frac{4}{3}, \\ s_1(\lambda) &= 4 - (16/5)\lambda, \\ s_2(\lambda) &= \frac{2}{3}[28 - 56\lambda + 32\lambda^2], \\ s_3(\lambda) &= \frac{2}{9}[60 - 216\lambda + 288\lambda^2 - 128\lambda^3]. \end{aligned}$$

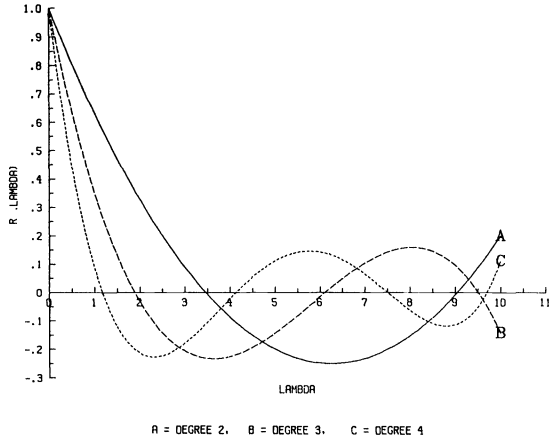


FIG. 2.1. Residual polynomials  $R_k$  for  $k = 2, 3$  and  $4$ .

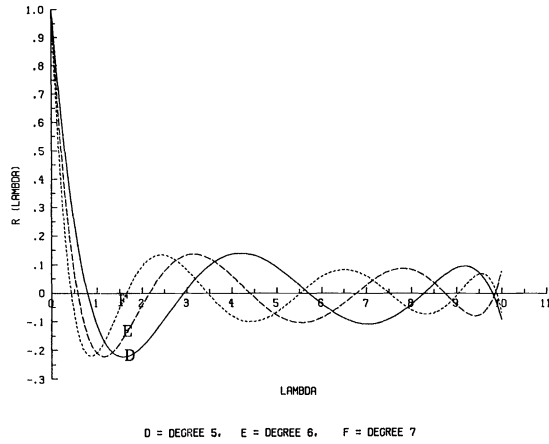


FIG. 2.2. Residual polynomials  $R_k$  for  $k = 5, 6$  and  $7$ .

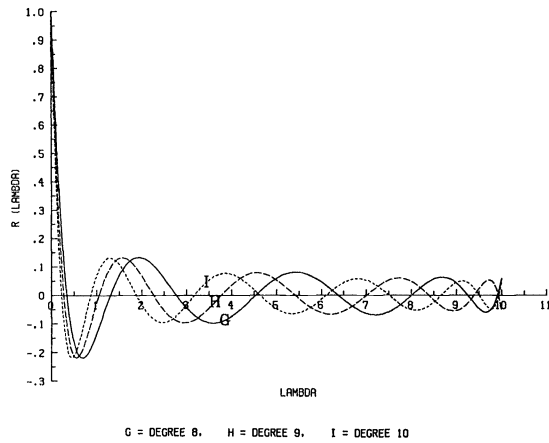


FIG. 2.3. Residual polynomials  $R_k$  for  $k = 8, 9$  and  $10$ .

Recall that the above polynomials correspond to the interval  $[0, 1]$ . For a more general interval of the form  $[0, b]$ , a change of variable must be applied to map the variable in  $[0, b]$  into  $[0, 1]$ , i.e. the  $k$ th best polynomial for the interval  $[0, b]$  is  $(1/b)s_{k-1}(\lambda/b)$ . The plots of the residual polynomials  $R_k(\lambda)$ ,  $k = 2, 3, \dots, 10$ , for the above weight functions are presented in Figs. 2.1, 2.2, 2.3. The interval considered is  $[a, b] = [0, 10]$ . The polynomials  $s_k(\lambda)$   $k = 1, 2, \dots, 10$  have been computed by the method of the previous section and are listed in the appendix.

Note that we have taken  $\alpha = \frac{1}{2}$  and  $\beta = -\frac{1}{2}$  as an example only because this choice leads to a very simple recurrence for the polynomials  $q_i$ , which are the Chebyshev polynomials of the first kind. We will also use this selection for the numerical experiments in § 4. Johnson, Micchelli and Paul [8] have taken as an example  $\alpha = 1$ , and  $\beta = 0$  which corresponds to the Legendre weight. An interesting problem from the practical viewpoint is to determine what is a “good” choice for  $\alpha$  and  $\beta$ .

**2.4. Theoretical considerations.** An interesting theoretical question is whether the least squares residual polynomial will become small in some sense as its degree increases. Consider first the case  $0 < a < b$ . Since the residual polynomial  $R_k$  minimizes the norm  $\|R\|_w$  associated with the weight  $w$ , over all polynomials  $R$  of degree  $\leq k$  such that  $R(0) = 1$ , the polynomial  $(1 - \lambda/c)^k$  where  $c = (a + b)/2$  satisfies

$$\|R_k\|_w \leq \|(1 - \lambda/c)^k\|_w \leq \|[ (b - a)/(b + a) ]^k\|_w = \kappa [ (b - a)/(b + a) ]^k,$$

where  $\kappa$  is the  $w$ -norm of the function unity on the interval  $[a, b]$ . Hence, the (known) result that the norm of  $R_k$  will tend to zero geometrically as  $k$  tends to infinity.

Consider now the case  $a = 0, b = 1$  and the Jacobi weight (9). Then for this choice of the weight function, the least squares residual polynomial is known to be  $p_k(\lambda)/p_k(0)$  where  $p_k$  is the  $k$ th degree Jacobi polynomial associated with the weight function  $w'(\lambda) = \lambda^\alpha(1 - \lambda)^\beta$ , [20]. Consider the 2-norm of such a residual polynomial with respect to this weight. From [4] and after a change of variable that maps the interval  $[-1, 1]$  into  $[0, 1]$  we obtain

$$p_k(0) = \frac{\Gamma(k + \alpha + 1)}{\Gamma(k + 1)\Gamma(\alpha + 1)}$$

where  $\Gamma$  represents the  $\Gamma$  function, and

$$\|p_k\|_{w'}^2 = \frac{1}{2k + \alpha + \beta + 1} \frac{\Gamma(k + \alpha + 1)\Gamma(k + \beta + 1)}{\Gamma(k + 1)\Gamma(k + \beta + \alpha + 1)}.$$

Hence,

$$\|p_k/p_k(0)\|_{w'}^2 = \frac{\Gamma^2(\alpha + 1)\Gamma(k + \beta + 1)}{(2k + \alpha + \beta + 1)\Gamma(k + \alpha + \beta + 1)} \frac{\Gamma(k + 1)}{\Gamma(k + \alpha + 1)}.$$

For the case  $\alpha = \frac{1}{2}$  and  $\beta = -\frac{1}{2}$  this becomes

$$\frac{[\Gamma(3/2)]^2}{(2k + 1)(k + 1/2)} = \frac{\pi}{2(2k + 1)^2}.$$

Therefore, the  $w'$ -norm of the least squares residual polynomial will converge to zero like  $1/k$  as the degree  $k$  increases. This is not as fast as when  $\alpha > 0$  but we must remember that the condition  $p(0) = 1$ , implies that the polynomial is large in some interval around the origin.

The case  $a < 0 < b$  is more difficult to study. Clearly, the problem is to analyse the numbers

$$\min_{p \in P_k, p(0)=1} \int_a^b |p(\lambda)|^2 w_0(\lambda) d\lambda$$

as  $k$  increases to infinity, where  $w_0$  is some weight function. By the change of variable  $\mu = (\lambda - a)(b - a)$ , these numbers are transformed into

$$\kappa_k(\gamma) = \min_{p \in P_k, p(\gamma)=1} \int_0^1 |p(\mu)|^2 w(\mu) d\mu,$$

where  $w(\mu) = w_0(\lambda)$  and  $\gamma = -a/(b - a)$  is inside  $[0, 1]$ . A detailed analysis by Nevai [14] has shown that this function of  $\gamma$ , called the Christoffel function, decreases to zero like  $1/k$ , when  $w$  is a Jacobi weight.

**3. Polynomial preconditionings and parallel processing.**

**3.1. Parallel computation of  $p(A)v$ .** Consider a linear system  $Ax = f$  issued from the discretization of a partial differential equation in two or three dimensions. With a suitable ordering of the nodes, the matrix  $A$  is block-tridiagonal and often of very large size. We would like to show how to exploit the polynomial preconditioning discussed in the preceding sections to solve these linear systems. A critical part in the realization of the conjugate gradient method lies in the computation of  $s(A)Av$  for any given vector  $v$ , where  $p(A) \equiv As(A) = s(A)A$  is the polynomial preconditioned matrix. Because of the particular structure of  $A$ , a great deal of parallelism can be achieved in the computation of  $p(A)v$ . In the following description we will assume that the degree of  $p$  is 3. We will call  $\nu$  the block dimension of  $A$  and  $m$  the dimension of each block, i.e. we have  $N = \nu m$ .

Let  $v$  be any vector and let  $w = Av$ ,  $z = A^2v = Aw$ ,  $t = A^3v = Az$ . We partition all the vectors according to the block structure of  $A$  and denote by  $v_i, w_i, z_i, t_i, i = 1, \dots, \nu$ , the block entries of  $v, w, z$ , and  $t$  respectively. Notice that the computation of any block entry  $w_i$  of  $w = Av$ , requires only the knowledge of the block entries  $v_{i-1}, v_i$  and  $v_{i+1}$ . Then a key observation is the following: while we compute  $w_i$  from  $v_{i-1}, v_i$  and  $v_{i+1}$  we can at the same time compute  $z_{i-2}$  from  $w_{i-3}, w_{i-2}, w_{i-1}$  and  $t_{i-4}$  from  $z_{i-5}, z_{i-4}, z_{i-3}$ . This is illustrated in Fig. 3.1 where we assume that we perform the computations

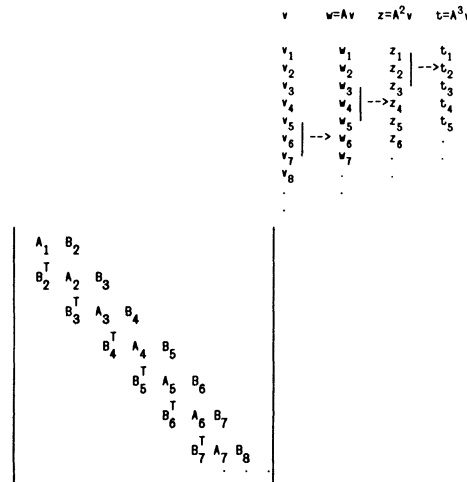


FIG. 3.1. Simultaneous computation of  $Av, A^2v$  and  $A^3v$ .

from top to bottom. These three computations can obviously be performed independently from each other by three different processors. This simple idea is not completely new as a similar principle was already used in a different context in 1963 by Pfeifer [15] who suggested performing simultaneously several steps of the three line cyclically reduced SOR method.

Concerning the entries of the matrix  $A$ , note that in order to compute  $w_i$  we only need the blocks  $A_i$ ,  $B_i$  and  $B_{i+1}$ . Likewise, we need  $B_{i-2}$ ,  $A_{i-2}$  and  $B_{i-1}$  to compute  $z_{i-2}$  and we need  $B_{i-4}$ ,  $A_{i-4}$  and  $B_{i-3}$ , to compute  $t_{i-4}$ . Hence, the computation can be pipelined by holding 4 blocks in each processor and moving everything to the left by two blocks ( $A_i$  and  $B_{i+1}$ ) at each time step as is shown in Fig. 3.2. Although only three of the four blocks in each processor are used, this organization simplifies the data flow of the matrix entries. Meanwhile, the vectors move only by one block at a time. Figure 3.2 illustrates two successive steps of the process. Note that the computations in each processor are identical but use three different sets of data. Processor  $P_1$  has the task of computing  $w = Av$ , processor  $P_2$  computes  $z = Aw$ , and processor  $P_3$  computes  $t = Az$ . As soon as a component is computed, it is sent to the processor on its left. For example, as the figure shows, when the computation of  $w_6$  is completed,  $w_6$  is moved to processor  $P_2$ , where it will be used in the next step together with  $w_4$  and  $w_5$  to produce  $z_5$ . Simultaneously,  $z_4$  is moved to  $P_2$  and  $t_2$ , the final result, is moved to some host processor that will complete the conjugate gradient step as will be discussed shortly.

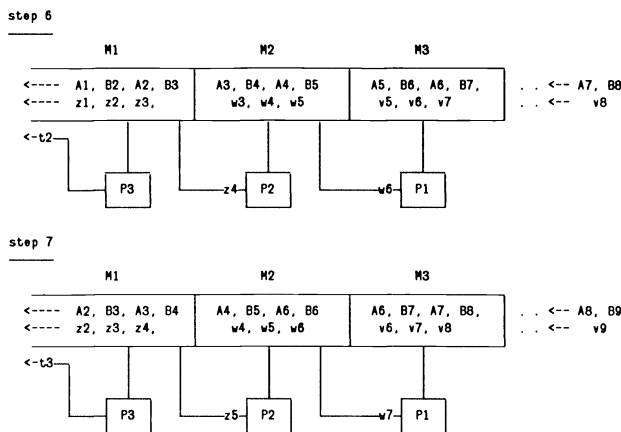


FIG. 3.2. Parallel processing for computing  $p(A)v$ .

In the network described by the figure, the processors  $P_1, P_2, P_3$  may represent vector or array processors with their own memories  $M_1, M_2, M_3$  sufficiently large to hold four blocks of  $A$ . In [11] Jordan describes a technique for computing  $w = Av$  that only requires two blocks of  $A$  at a time, namely  $A_i$  and  $B_{i+1}$  to get one corresponding bloc of  $w$ . This amounts to separating the computation of  $w_i = B_i^T v_{i-1} + A_i v_i + B_{i+1} v_{i+1}$   $i = 1, 2, \dots$ , into

$$(11) \quad w_i = \tilde{w}_i + A_i v_i + B_{i+1} v_{i+1},$$

$$(12) \quad \tilde{w}_{i+1} = B_{i+1}^T v_i,$$

with  $\tilde{w}_1 = 0$ . With such a technique we would only need two blocks of  $A$  in each processor. The vector  $\tilde{w}_{i+1}$  is not moved to the left but will remain in the same processor to compute  $w_{i+1}$  by (11) in the next time step.

The advantages of the approach outlined in this section are the following:

1. A high degree of parallelism is reached. The processors are idle only during a very small portion of the total time spent to compute  $p(A)v$ , namely at the beginning and at the end of the computation.

2. The computation involved in each of the processors is a highly vectorizable process. Thus, each processor can represent a vector or array processor and this can be quite important for very large problems, e.g. those issued from finite difference discretizations of three-dimensional PDE's.

3. The data communication required is regular. Moreover, a large part of the data transfer can easily be overlapped with computation.

We have just described how to compute the vectors  $Av$ ,  $A^2v$ ,  $A^3v$  but in reality we need to compute the vector  $t = p(A)v$  where  $p$  is some polynomial of the form  $p(\lambda) = \alpha_3[\lambda^3 + \alpha_2\lambda^2 + \alpha_1\lambda + \alpha_0]$ . Therefore, we will not compute the vectors  $w = Av$ ,  $z = Aw$ ,  $t = Az$ , but rather the vectors:

$$(13) \quad w := \alpha_2 v + Av,$$

$$(14) \quad z := \alpha_1 v + Aw,$$

$$(15) \quad t := \alpha_0 v + Az,$$

$$(16) \quad t := \alpha_3 t,$$

which result from the application of Horner's scheme for evaluating a polynomial. The resulting modification to the above scheme requires the transfer of the additional vector  $v$  through the three processors, and is straightforward.

**3.2. Application to the conjugate gradient method.** We now turn to the implementation of the polynomial preconditioned conjugate gradient method a classical version of which can be described as follows.

**ALGORITHM 1.**

1. *Start:* Choose  $x^{(0)}$  and compute  $r^{(0)} := s(A)(f - Ax^{(0)})$ . Set  $v^{(0)} := r^{(0)}$ ,  $\rho^{(0)} := (r^{(0)}, r^{(0)})$ .

2. *Iterate:* For  $j = 1, 2, \dots$  until convergence *do*

$$(17) \quad t^{(j)} := s(A)Av^{(j)};$$

$$(18) \quad \alpha^{(j)} := \rho^{(j)} / (t^{(j)}, v^{(j)});$$

$$(19) \quad x^{(j+1)} := x^{(j)} + \alpha^{(j)}v^{(j)}$$

$$(20) \quad r^{(j+1)} := r^{(j)} - \alpha^{(j)}t^{(j)};$$

$$(21) \quad \rho^{(j+1)} := (r^{(j+1)}, r^{(j+1)});$$

$$(22) \quad \beta^{(j)} := \rho^{(j+1)} / \rho^{(j)};$$

$$(23) \quad v^{(j+1)} := r^{(j+1)} + \beta^{(j)}v^{(j)}.$$

The inner products in (18) and (22) constitute two bottlenecks in the above algorithm. Indeed, as the algorithm is presented, we must proceed as follows. As the blocks of the vectors  $v$  and  $t$  emerge from the pipeline of Fig. 3.2 we must compute the partial inner product in (18) corresponding to these blocks. The blocks are then stored back into memory awaiting for the completion of the partial inner products. After the  $v$  partial inner products have been computed and added up, the vectors  $t$



and  $v$  are then recalled one block at a time and the scalar by vector products (19), (20) will be performed.

A similar remark holds for the computation of the inner product (22). However, an interesting observation is that there is an alternative way of computing  $\beta^{(j)}$  which avoids this second bottleneck. Indeed, let  $r^{(j+1)}$  and  $r^{(j)}$  be two successive residual vectors, and  $t^{(j)} = p(A)v^{(j)}$ ,  $\alpha^{(j)} = (r^{(j)}, r^{(j)}) / (t^{(j)}, v^{(j)})$  where  $v^{(j)}$  is the conjugate direction at step  $j$ . Since the two vectors  $r^{(j)}$  and  $r^{(j+1)}$  are known to be orthogonal, we have from (20)

$$(r^{(j+1)}, r^{(j+1)}) + (r^{(j)}, r^{(j)}) = [\alpha^{(j)}]^2 (t^{(j)}, t^{(j)}).$$

Hence, another way of obtaining the inner product  $(r^{(j+1)}, r^{(j+1)})$  is through the formula

$$(24) \quad (r^{(j+1)}, r^{(j+1)}) = [\alpha^{(j)}]^2 (t^{(j)}, t^{(j)}) - (r^{(j)}, r^{(j)}).$$

As a consequence  $(r^{(j+1)}, r^{(j+1)})$  is available from  $(r^{(j)}, r^{(j)})$  and  $(t^{(j)}, t^{(j)})$ . The fundamental difference with the usual way of computing  $(r^{(j+1)}, r^{(j+1)})$  is that  $(t^{(j)}, t^{(j)})$  can be computed simultaneously with  $(t^{(j)}, v^{(j)})$ , i.e. it can be accumulated as the blocks of the vectors  $t^{(j)}$  come out of the network of Fig. 3.1 one by one. A similar observation was made by Johnsson [9] while Van Rosendale [22] studied the inner product data dependencies in the conjugate gradient method in a more general and theoretical way.

The following algorithm implements the above approach.

ALGORITHM 2.

1. *Start*: Choose  $x^{(0)}$  and compute  $r^{(0)} := s(A)(f - Ax^{(0)})$ . Set  $v^{(0)} := r^{(0)}$ ,  $\rho^{(0)} := (r^{(0)}, r^{(0)})$ .

2. *Iterate*: For  $j = 1, 2, \dots$  until convergence *do*

a) Compute in parallel:

$$(25) \quad t^{(j)} := s(A)Av^{(j)}$$

b) Compute in parallel:

$$(26) \quad (t^{(j)}, v^{(j)}),$$

$$(27) \quad (t^{(j)}, t^{(j)}),$$

then compute

$$(28) \quad \alpha^{(j)} := \rho^{(j)} / (t^{(j)}, v^{(j)}),$$

$$\beta^{(j)} := \rho^{(j)} \frac{(t^{(j)}, t^{(j)})}{(t^{(j)}, v^{(j)})^2} - 1,$$

$$\rho^{(j+1)} := \rho^{(j)} \beta^{(j)}.$$

c) Compute in parallel:

$$(29) \quad x^{(j+1)} := x^{(j)} + \alpha^{(j)}v^{(j)}$$

$$(30) \quad r^{(j+1)} := r^{(j)} - \alpha^{(j)}t^{(j)},$$

$$(31) \quad v^{(j+1)} := r^{(j+1)} + \beta^{(j)}v^{(j)}.$$

Note that because of roundoff, the equation (28) could lead to a negative value for  $\beta^{(j)}$ . The effect of roundoff can be reduced by a periodic application of the less effective formulas of Algorithm 1. In any case the sign of  $\beta^{(j)}$  should be checked before

applying the result to the next equations and if this sign is negative then  $\beta^{(j)}$  should be recomputed by the classical formulas of Algorithm 1.

This second form of the conjugate gradient algorithm is also to be preferred on vector computers such as the CRAY-1. Indeed, in that situation the expressions “in parallel” in the above algorithm should be interpreted in the sense that the presence of the operands in the vector registers must be exploited to perform the desired computations at once thus economizing vector “load” and “store” operations. Practically, this can be achieved in FORTRAN by having a single DO loop for (26) and (27) and another single DO loop for (29), (30) and (31).

Fig. 3.3 shows a network of processors for realizing the above conjugate gradient algorithm. The first part represents the pipelined module described in Fig. 3.2 consisting of  $k$  linearly connected processors  $P_1, \dots, P_k$ . The second part is a host processor, denoted by  $H$  in the figure, which realizes steps 2b and 2c of Algorithm 2. Processor  $H$  need not hold the matrix  $A$  but only the vectors  $x, r, v$  and  $t$ . It may itself consist of one or more array processors.

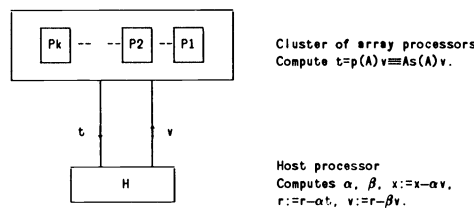


FIG. 3.3. A parallel architecture for the polynomial preconditioned conjugate gradient method.

An interesting question is how does this compare with the conjugate gradient algorithm without preconditioning. The following is a rough estimate of the time spent to execute one step of the polynomial preconditioned conjugate gradient method. Assume that  $H$  and each of the  $P_i$ 's are similar and that all can perform the product of two vectors of length  $m$ , or the product of a vector by a scalar, in an average time of  $\tau_1 m$ , and an inner product in an average time of  $\tau_2 m$ . These times are assumed to include data transfers. Furthermore, assume that  $A$  is a 5-point discretization matrix. Taking into account the combinations (13)–(15), the computation of each block of  $p(A)v$  can be achieved in time  $6m\tau_1$  in each processor. Since the time to make the first block available, i.e. the start up time, is  $k(6m\tau_1)$  and since we have  $\nu$  blocks to treat,  $p(A)v$  can be computed in time  $6\tau_1 mk + 6\tau_1 m\nu = [6\tau_1 + k/\nu]N$ .

After computation of each block of the vector  $t = p(A)v$  is completed, this block is sent to  $H$  to begin computing the inner products  $(t, v)$  and  $(t, t)$  of (26) and (27). Each partial inner product costs  $m\tau_2$ . However, assuming that this time is not larger than  $6m\tau_1$ , we see that the computation of the inner products can be overlapped with the computation of  $t$ . The only time that is really spent is the start up time in  $H$ , which is  $m\tau_2$  for each of the inner products (26) and (27), i.e.  $2m\tau_2$  altogether. The rest of the conjugate gradient calculations (29), (30), (31) consumes a time of  $3\rho_1 N$ . Therefore, an estimate of the time spent to perform one iteration of the polynomial preconditioned conjugate gradient algorithm is

$$\tau = N[(9 + k/\nu)\tau_1 + (2/\nu)\tau_2].$$

With a single processor, a conjugate gradient step would require  $5\tau_1 N$  (for the matrix by vector product) +  $2\tau_2 N$  (for the inner products (18), (22)) +  $3\tau_1 N$  (for (19), (20), (23)) which sums up to  $8\tau_1 N + 2\tau_2 N$ . This is slightly larger than the previous

case when  $k \ll \nu$  and  $\tau_1$  and  $\tau_2$  are of the same order. Hence the speed up is of the same order as the ratio of the total number of preconditioned conjugate gradient steps over that of the nonpreconditioned conjugate gradient steps. Although it is difficult to a priori estimate this ratio, we can say that it is of the form  $\gamma(k+1)$  where  $\gamma$  is a scalar larger than one. Often  $\gamma$  will be between 1 and 2, sometimes close to one. In an example shown in § 4,  $\gamma$  is around 1.2.

The above comparison uses a very simple model. An advantage of the multiprocessor described above and not stressed in the comparison is that the matrix is accessed only once to perform  $k$  matrix by vector products, and then moved in a uniform way. With a single processor we would have to access the matrix  $k$  times to perform  $k$  similar matrix by vector products. This will be reflected by smaller times  $\tau_1$  and  $\tau_2$  in a multiprocessor environment than in a single processor environment, and this was not taken into account in the above simplistic comparison.

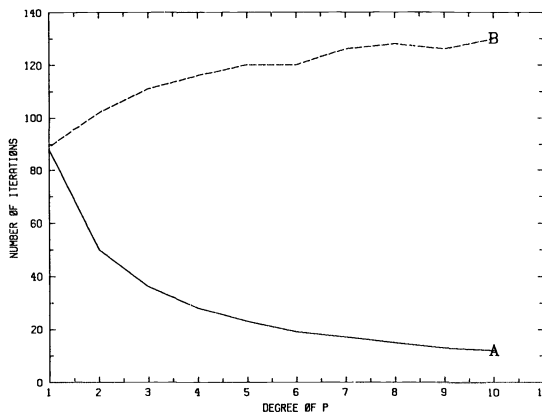
**4. Numerical experiments.** In this section we describe a few numerical experiments in order to point out some additional facts about polynomial preconditionings. All tests have been performed in double precision on a VAX-11-700 computer for which the double precision mantissa is 56. The least squares polynomials referred to in this section are those associated with the Jacobi weight with  $\alpha = \frac{1}{2}$  and  $\beta = -\frac{1}{2}$  given as an example in § 2.

**4.1. Varying the degree  $k$ .** The purpose of the first experiment is to illustrate the behaviour of the polynomial preconditioned conjugate gradient method as the degree of  $s_{k-1}$  varies. Consider the  $1200 \times 1200$  block tridiagonal matrix:

$$A = \text{Block-Tridiag} [B_i^T, A_i, B_{i+1}], \quad i = 1, \dots, \nu$$

with  $A_i = \text{Tridiag} [-1, 4, -1]$  and  $B_i = \text{Diag} (-1)$ ,

resulting from the 5-point discretization of the Laplacian operator on a rectangle. The dimension of each of the blocks is  $m = 40$ , i.e. the block-dimension of  $A$  is  $\nu = 30$ . We tested the polynomial preconditioned conjugate gradient method on the system  $Ax = f$  and have used for  $a$  and  $b$  the Gershgorin values  $a = 0$ ,  $b = 8$ . The right-hand side is chosen to be  $f = Ae$ , where  $e = (1, 1, 1, \dots, 1)^T$ . The initial vector is a random vector. The algorithm is stopped as soon as the approximate solution  $x^{(j)}$  satisfies



A - CG ITERATIONS ON PRECONDITIONED PROBLEM VERSUS DEGREE OF P  
 B - TOTAL NUMBER OF MATRIX-VECTOR MULTIPLIES VERSUS DEGREE OF P

FIG. 4.1. Iterations versus degree of preconditioning.

$\|f - Ax^{(j)}\| / \|f - Ax^{(0)}\| \leq \varepsilon$ , where  $\varepsilon = 10^{-5}$ . These residual norms are actually computed at each (preconditioned) conjugate gradient step. Note that in a realistic implementation we need not compute these actual residual vectors because we can either use the generalized residual vectors  $r^{(j)} = s_k(A)(f - Ax^{(j)})$  or use a classical implementation of the preconditioned conjugate gradient algorithm in which both the actual residual and the generalized residual are available, at the expense of a little more storage, e.g. see [12]. The plot of Fig. 4.1, shows the number of iterations for all values of the degree  $k$  of  $p_k(A) \equiv s_{k-1}(A)$  between 1 and 10. Note that  $k = 1$  means that  $s_{k-1}$  is a constant and corresponds to the nonpreconditioned conjugate gradient algorithm.

On the same figure we have plotted the total number of matrix by vector multiplications needed for convergence, i.e. the numbers  $k(IT+1)$  where  $IT$  is the number of iterations. Note that except for the first 3 degrees, the total number of matrix by vector multiplications changes relatively little which is why the number of iterations drops like  $1/k$  as  $k$  increases. The polynomials  $s_{k-1}$  have been determined by the kernel polynomial formulation described in § 2.2.

**4.2. Varying the parameter  $a$ .** In the next experiment we illustrate the behaviour of the polynomial preconditioned conjugate gradient method as the parameter  $a$  varies. It is known that the performance of the Chebyshev iteration algorithm [6], [7] depends critically on  $a$  being an accurate approximation of the smallest eigenvalue  $\lambda_1$ . We would like to show that the number of steps of the polynomial preconditioned conjugate gradient method depends very little on the accuracy of  $a$  as an approximation to the eigenvalue  $\lambda_1$ . The linear system tested is the same as above, and so are the tolerance  $\varepsilon$  and the initial vector  $x_0$ . The value of the parameter  $b$  is fixed to 8 and  $a$  is varied from 0.0 to 1.3. The larger values of  $a$  are more widely spaced than the first ones. We have tried three different degrees:  $k = 3$ ,  $k = \beta$  and  $k = 10$ . The results are shown in Fig. 4.2 in which we plot the total number of *CG* iterations to achieve convergence. Notice that around the smallest eigenvalue which is  $\lambda_1 \approx 1.613 \times 10^{-2}$ , there is little change in using different values of  $a$ . Amazingly, even for quite large values of  $a$ , the performance is little affected. For  $k = 3$ , for example, there is hardly any difference in performance for  $a = 0.0$  through  $a = 1.3$ . We attribute this phenomenon to the fact that

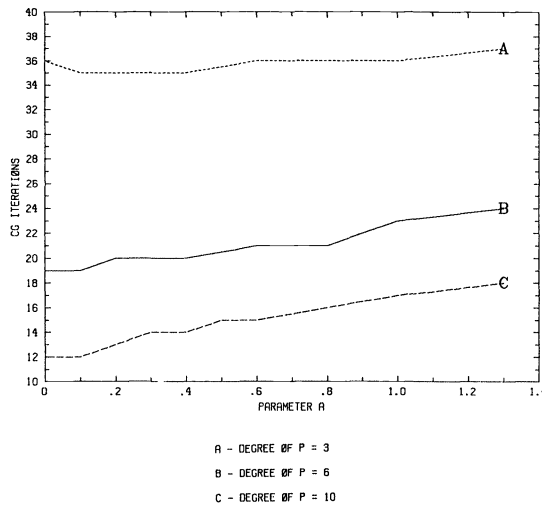


FIG. 4.2. Behavior of the least squares polynomial preconditioning as the parameter  $a$  varies.

the polynomials change very little when the parameter  $a$  moves around the origin to its right. We believe that this phenomenon will also be verified for other choices of the Jacobi weights such as the Legendre weight although this remains to be verified. This would mean that we expect the preconditionings suggested here to perform as well as those of [8] in spite of the fact that they require no eigenvalue computation.

**4.3. Comparison with Chebyshev polynomial preconditionings.** The next experiment compares Chebyshev polynomial preconditionings with the least squares polynomial preconditionings and reveals an interesting phenomenon which does not seem to have been pointed out in the literature. Let us consider the problem tested in § 4.1 with the same dimension  $N = 1,200$ , the same right-hand side  $f$  and initial vector  $x_0$  and the same stopping criterion. If we try the Chebyshev polynomial preconditioning with  $a = 0.016 \approx \lambda_1$  and  $b = 7.984 \approx \lambda_N$ , we find that for a polynomial  $\lambda s(\lambda)$  of degree 5, the method converges in a total of 165 matrix by vector multiplications versus 120 for the least squares polynomial preconditioning using  $a = 0$ ,  $b = 8$  as is reported in § 4.1.

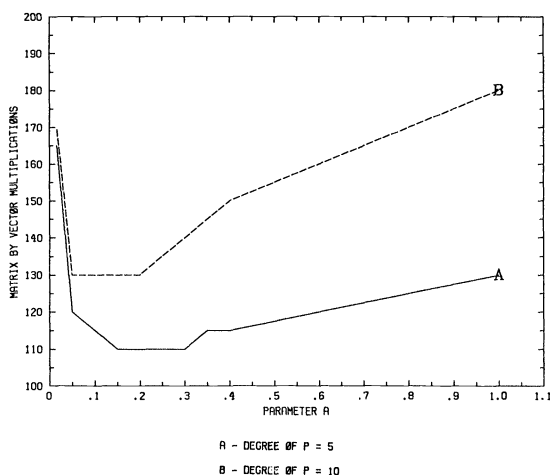


FIG. 4.3. Behavior of Chebyshev polynomial preconditioning as the parameter  $a$  varies.

Redoing the experiment with several different values for the parameter  $a$ ,  $b$  being fixed, we find, to our surprise, that  $a = \lambda_1$  is not the best possible value for  $a$ . Indeed, Fig. 4.3 shows that as  $a$  increases the performance improves drastically in the beginning and then starts deteriorating slowly as  $a$  becomes too large. In the plot the value of the parameter  $b$  is fixed to be 7.984 throughout and  $a$  is varied from 0 to 1. The best value for  $a$  is around  $a \approx 0.2$  which yields convergence in a total of 110 matrix by vector multiplications. This phenomenon can be explained as follows. By taking a slightly larger  $\lambda_1$ , we obtain a matrix  $As(A)$  that has a few “large” eigenvalues around one but the remainder of the eigenvalues are smaller than with  $a = \lambda_1$ . This situation is more favourable for the conjugate gradient method than when there are no “large” eigenvalues but the whole spectrum is wider; this occurs when  $a$  moves towards the origin. The same experiment is repeated for the degree  $k = 10$ . The conclusions to be drawn from these experiments are the following:

1. The usual optimal parameters  $a = \lambda_1$  and  $b = \lambda_N$  used in Chebyshev iteration are no longer optimal in Chebyshev polynomial preconditioned conjugate gradient method. When  $b$  is fixed, the true optimal parameter  $a$  is likely to be larger than  $\lambda_1$ . We do not know of any way of determining the actual optimal parameters  $a$  and  $b$ .

2. The least squares polynomial preconditioning with the simple parameters  $a = 0$ ,  $b = 8$  given by a Gershgorin argument performs nearly as well as the best performing Chebyshev polynomial preconditionings using the parameters  $a = 0.2$  for  $k = 5$  and  $a = 0.1$  for  $k = 10$ .

Note that these observations are not isolated. Therefore, not only is the Chebyshev polynomial approach difficult to optimize but even when optimized, the result is not significantly better than the simpler least squares polynomial approach. Using the eigenvalues  $\lambda_1$  and  $\lambda_N$  as optimal parameters for Chebyshev preconditioning may yield poor convergence.

Finally we would like to close this section by mentioning that G. Meurant [13] recently performed some interesting experiments on the CRAY-1 and CYBER 205 vector machines which show that polynomial preconditionings of the type proposed in this paper compare quite well with other vector machine oriented preconditionings. His final conclusion, however, is that there is no winner as performances depend heavily on the architecture as well as the data. An important advantage of polynomial preconditionings is that whereas classical preconditionings must start by computing an approximate factorization, polynomial preconditionings do not require any sort of preprocessing. It should be pointed out that in most cases the computation of an incomplete factorization is *not* a vectorizable process, and can take a substantial portion of the overall computing time. The timings reported in [13] do not account for preprocessings. Two other advantages of the least squares polynomial preconditionings described in this paper are their simplicity and their generality.

**5. Appendix.** This appendix gives the least squares polynomials up to the degree 10, when we use the Jacobi weights with  $\alpha = \frac{1}{2}$  and  $\beta = -\frac{1}{2}$ . We chose to develop the formulas for the interval  $[0, 4]$  as they are simpler to write down. Moreover, we rescaled the polynomials for simplicity. More precisely the  $k$ th degree polynomial in the list below must be multiplied by the factor  $4/(3 + 2k)$  to obtain the least squares polynomials  $s_k$  as defined in § 2. Note, however that a scaling factor is unimportant if one wants to use these polynomials for preconditioning. The polynomials have been obtained by a simple FORTRAN program based on the kernel polynomial approach described in § 2.2. Finally, recall that a change of variable is necessary if  $b \neq 4$  to map the interval  $[0, b]$  into  $[0, 4]$ , i.e. for a general interval one must take the polynomial  $s_k(4\lambda/b)$ .

$$s_1(\lambda) = 5 - \lambda,$$

$$s_2(\lambda) = 14 - 7\lambda + \lambda^2,$$

$$s_3(\lambda) = 30 - 27\lambda + 9\lambda^2 - \lambda^3,$$

$$s_4(\lambda) = 55 - 77\lambda + 44\lambda^2 - 11\lambda^3 + \lambda^4,$$

$$s_5(\lambda) = 91 - 182\lambda + 156\lambda^2 - 65\lambda^3 + 13\lambda^4 - \lambda^5,$$

$$s_6(\lambda) = 140 - 378\lambda + 450\lambda^2 - 275\lambda^3 + 90\lambda^4 - 15\lambda^5 + \lambda^6,$$

$$s_7(\lambda) = 204 - 714\lambda + 1122\lambda^2 - 935\lambda^3 + 442\lambda^4 - 119\lambda^5 + 17\lambda^6 - \lambda^7,$$

$$s_8(\lambda) = 285 - 1,254\lambda + 2,508\lambda^2 - 2,717\lambda^3 + 1,729\lambda^4 - 665\lambda^5 + 152\lambda^6 - 19\lambda^7 + \lambda^8,$$

$$s_9(\lambda) = 385 - 2,079\lambda + 5,148\lambda^2 - 7,007\lambda^3 + 5,733\lambda^4 - 2,940\lambda^5 + 952\lambda^6 - 189\lambda^7 + 21\lambda^8 - \lambda^9,$$

$$s_{10}(\lambda) = 506 - 3,289\lambda + 9,867\lambda^2 - 16,445\lambda^3 + 16,744\lambda^4 - 10,948\lambda^5 + 4,692\lambda^6 - 1,311\lambda^7 + 230\lambda^8 - 23\lambda^9 + \lambda^{10}.$$

**Acknowledgments.** The author has benefited from valuable discussions with Prof. Martin Schulz, Prof. Stan Eisenstat and Dr. Gerard Meurant.

## REFERENCES

- [1] L. M. ADAMS, *Iterative algorithms for large sparse linear systems on parallel computers*, Ph.D. thesis, Applied Mathematics, Univ. of Virginia, Blacksburg, 1982. Also available as NASA Contractor Report #166027.
- [2] C. C. CHENEY, *Introduction to Approximation Theory*, McGraw-Hill, New York, 1966.
- [3] P. CONCUS, G. H. GOLUB AND G. MEURANT, *Block preconditioning for the conjugate gradient method*, Technical Report LBL-14856, Lawrence Berkeley Lab., 1982; this Journal, 6 (1985), pp. 220-252.
- [4] P. H. DAVIS, *Interpolation and Approximation*, Blaisdell, Waltham, MA, 1963.
- [5] P. F. DUBOIS, A. GREENBAUM AND G. H. RODRIGUE, *Approximating the inverse of a matrix for use on iterative algorithms on vectors processors*, Computing, 22 (1979), pp. 257-268.
- [6] G. H. GOLUB AND R. S. VARGA, *Chebyshev semi-iterative methods, successive overrelaxation iterative methods and second order Richardson iterative methods*, Numer. Math., 3 (1961), pp. 147-168.
- [7] A. L. HAGEMAN AND D. M. YOUNG, *Applied Iterative Methods*, Academic Press, New York, 1981.
- [8] O. G. JOHNSON, C. A. MICCHELLI AND G. PAUL, *Polynomial preconditions for conjugate gradient calculations*, SIAM J. Numer. Anal., 20 (1983), pp. 362-376.
- [9] L. JOHNSON, *Highly concurrent algorithms for solving linear systems of equations*, in Elliptic Problem Solvers II, Proceedings of the Elliptic Problem Solvers Conference, Monterey CA, Jan. 10-12, 1983, G. N. Birkhoff and A. Schoenstadt, eds., Academic Press, New York, 1983, pp. 105-126.
- [10] T. L. JORDAN, *Conjugate gradient preconditioners for vector and parallel processors*, in Elliptic Problem Solvers II, Proceedings of the Elliptic Problem Solvers Conference, Monterey CA, Jan. 10-12, 1983, G. N. Birkhoff and A. Schoenstadt, eds., Academic Press, New York, 1983, pp. 127-139.
- [11] ———, *A guide to parallel computation and some Cray-1 experiences*, in Parallel Computations, Garry Rodrigue, ed., Academic Press, New York, 1982, pp. 1-50.
- [12] 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.
- [13] G. MEURANT, *Vector preconditioning for the conjugate gradient on the CRAY-1 and CDC Cyber 205*, in Proceedings of the 6th International Colloquium on Scientific and Technical Computation, Versailles, France, 1984, INRIA ed., North-Holland, Amsterdam, 1984.
- [14] P. G. NEVAI, *Distribution of zeros of orthogonal polynomials*, Trans. Amer. Math. Soc., 249, 2 (1979), pp. 241-261.
- [15] C. J. PFEIFER, *Data flow and storage allocation for the PDQ-5 program on the Philco-2000*, Comm. ACM, 6 (1963), pp. 365-366.
- [16] H. RUTISHAUSER, *Theory of gradient methods*, in Refined Iterative Methods for Computation of the Solution and the Eigenvalues of Self-Adjoint Boundary Value Problems, Institute of Applied Mathematics, Zurich, Basel-Stuttgart, 1959, pp. 24-49.
- [17] Y. SAAD, *Iterative solution of indefinite symmetric systems by methods using orthogonal polynomials over two disjoint intervals*, SIAM J. Numer. Anal., 20 (1983), pp. 784-811.
- [18] D. C. SMOLARSKI, *Optimum semi-iterative methods for the solution of any linear algebraic system with a square matrix*, Ph.D. thesis, Technical Report UIUCDCS-R-81-1077, Univ. Illinois, Urbana-Champaign, 1981.
- [19] SO CHENG CHEN, *Polynomial scaling in the conjugate gradient method and related topics in matrix scaling*, Ph.D. thesis, Technical Report CS-82-23, Pennsylvania State Univ., University Park, 1982.
- [20] E. L. STIEFEL, *Kernel polynomials in linear algebra and their applications*, U.S. NBS Applied Math. Series, 49 (1958), pp. 1-24.
- [21] H. A. VAN DER VORST, *A vectorizable version of some ICCG methods*, this Journal, 3 (1982), pp. 350-356.
- [22] J. VAN ROSENDALE, *Minimizing inner product data dependencies in conjugate gradient iteration*, Technical Report 172178, ICASE-NASA, 1983.

## AUXILIARY STORAGE METHODS FOR SOLVING FINITE ELEMENT SYSTEMS\*

ALAN GEORGE† AND HAMZA RASHWAN†

**Abstract.** In this paper we consider the direct solution of the system of linear equations  $Ax = b$ , where  $A$  is a large, sparse, symmetric and positive definite matrix. This system is solved using the Cholesky method by factoring  $A$  into  $R^T R$ , where  $R$  is an upper triangular matrix, and then solving  $R^T y = b$  and  $Rx = y$ . We are particularly interested in the case where  $A$  is so large that auxiliary storage must be used to store the Cholesky factor  $R$ .

The approach we use, which fully exploits the sparsity of  $A$ , is based on partitioning the given system into subsystems, each of which is sufficiently small that it can be processed in a relatively small amount of main memory. We give a detailed analysis of the input/output (I/O) traffic generated when our method is applied to a model  $n$  by  $n$  grid problem. The grid is partitioned using an *incomplete nested dissection*, which is a minor modification of the standard nested dissection ordering. Our analysis shows that if we have  $O(n^2)$  main memory, then the total I/O traffic is  $O(n^2 \log n)$ . This result implies that the  $O(n^3)$  numerical computations dominate the I/O traffic. A widely used method for solving such large linear systems is the band method, in which zeros outside the band of  $A$  are exploited. The I/O traffic and arithmetic operations in this case are given by  $O(n^3)$  and  $O(n^4)$ , respectively.

We also consider an improvement for our basic strategy which reduces the I/O traffic to the extent that it is dominated by writing the Cholesky factor alone. This enhancement reduces the I/O traffic associated with storing intermediate results from  $O(n^2 \log n)$  to  $O(n^2 \log \log n)$ . Numerical experiments are also provided to illustrate the performance of our algorithms.

**Key words.** sparse linear equations, auxiliary storage methods, sparse Gaussian elimination, finite element systems

**1. Introduction.** In this paper, we consider the direct solution of the system of linear equations

$$(1.1) \quad Ax = b,$$

where  $A$  is an  $N$  by  $N$ , sparse, symmetric and positive definite matrix. The system (1.1) is solved using Cholesky's method, by factoring  $A$  into  $A = R^T R$ , where  $R$  is an upper triangular matrix. The solution vector  $x$  is then found by solving the two triangular systems:  $R^T y = b$ , and  $Rx = y$ .

When the factorization of a sparse matrix  $A$  is carried out, it usually suffers *fill-in*; that is,  $R + R^T$  has nonzeros in positions which are zero in  $A$ . It is well known (George and Liu [1981a], Rose [1972]) that a judicious permutation of the rows and columns of  $A$  may dramatically reduce the amount of fill-in suffered by  $A$ . Thus, we might consider solving the equivalent system

$$(1.2) \quad (PAP^T)Px = Pb,$$

where  $P$  is a permutation matrix chosen to achieve some combination of the following objectives:

- i) To reduce the amount of fill-in suffered during the factorization step.
- ii) To reduce the number of arithmetic operations required during the factorization and solution steps.
- iii) To permute the matrix  $A$  into a form which enables the efficient use of auxiliary storage.

---

\* Received by the editors November 30, 1983. Research supported in part by Canadian Natural Sciences and Engineering Council grant A8111.

† Department of Computer Science, University of Waterloo, Waterloo, Ontario, Canada N2L 3G1.



When the matrix  $A$  is so large that its Cholesky factor cannot be stored entirely in the main memory of the computer, external storage is used to extend the main memory so that such large problems can be solved.

A *nested dissection* ordering (George [1977]) is effective in reducing fill-in and arithmetic computation in solving sparse positive definite linear systems arising in finite element and finite difference problems. Such nested dissection orderings are not commonly employed in conjunction with auxiliary storage despite their desirable low fill-in and low arithmetic operation counts. The main reason for this seems to be that the Cholesky factors corresponding to these orderings tend to have their nonzeros *scattered*, and therefore require *complicated* data structures and do not seem to be appropriate in connection with the use of auxiliary storage (Tinney [1969], Reid [1974]).

Our approach is based on partitioning the given system into subsystems, each of which is sufficiently small that it can be processed (factored) in a relatively small amount of main memory. The partitioning we use in this paper is provided by an *incomplete nested dissection*, which is a minor modification of the standard nested dissection ordering. In this paper we consider the analysis and implementation of methods which utilize such a partitioning, without compromising the low operations count of nested dissection orderings. For a model grid problem, we show that the arithmetic computations dominate the input/output (I/O) traffic.

An outline of the paper is as follows. In § 2 we review nested dissection and incomplete nested dissection orderings and partitionings for a model  $n$  by  $n$  grid problem. In § 3 we describe a method which uses nested dissection partitioning to solve large sparse linear systems. A detailed analysis of the I/O traffic generated by this method is given in § 4, and the results show that the numerical computations dominate the I/O traffic for the model  $n$  by  $n$  grid problem. In § 5 we consider an important enhancement over the scheme of § 3. This method achieves a low amount of I/O traffic, which is dominated by that required to write out the Cholesky factor alone, and hence it seems to be a near-optimal method. Numerical experiments are provided in § 6, and further extensions are considered in § 7.

**2. A review of nested dissection orderings and partitionings.** Following George [1977], let  $V$  be the set of vertices of the  $n$  by  $n$  mesh, where for convenience we assume  $n = 2^k - 1$  for some integer  $k$ . Let  $S_0 = S_0^1$  be the set of vertices on the vertical mesh line which divides the mesh into two equal parts  $R_1^1$  and  $R_1^2$ , where  $V - S_0 = R_1^1 \cup R_1^2 = R_1$ . Numbering the vertices in  $R_1^1$  followed by those in  $R_1^2$ , followed finally by those in  $S_0$ , induces the block structure in the associated matrix  $A$  as shown in (2.1). Here we assume there is a one variable  $x_r$  associated with each vertex of the mesh, and  $a_{rs} \neq 0$  if variable  $r$  and variable  $s$  are associated with vertices of the same small square in the mesh. For more details, see George [1977].

$$(2.1) \quad A = \begin{pmatrix} A_{11} & 0 & A_{13} \\ 0 & A_{22} & A_{23} \\ A_{13}^T & A_{23}^T & A_{33} \end{pmatrix}.$$

The Cholesky factor of  $A$  is given by

$$(2.2) \quad R = \begin{pmatrix} R_{11} & 0 & R_{13} \\ & R_{22} & R_{23} \\ & & R_{33} \end{pmatrix},$$

where  $A_{11} = R_{11}^T R_{11}$ ,  $A_{22} = R_{22}^T R_{22}$ ,  $R_{13} = R_{11}^{-T} A_{13}$ ,  $R_{23} = R_{22}^{-T} A_{23}$ , and  $R_{33}^T R_{33} = \bar{A}_{33} =$

$A_{33} - A_{13}^T A_{11}^{-1} A_{13} - A_{23}^T A_{22}^{-1} A_{23}$ . Note that the blocks  $A_{11}$  and  $A_{22}$  are *independent*; that is, the factorization of  $A_{11}$  has no effect on that of  $A_{22}$ , and vice versa.

Now choose vertex sets  $S_1^j \subset R_1^j, j = 1, 2$ , consisting of vertices lying on the horizontal mesh line that divides  $R_1^j$  into two equal parts. If we number the variables associated with the vertices in  $R_1^j - S_1^j$  before those associated with  $S_1^j, j = 1, 2$ , and then the remaining vertices as before, we induce the structure shown in (2.3).

$$(2.3) \quad A = \begin{pmatrix} A_{11} & & & & A_{15} & & A_{17} \\ & A_{22} & & & A_{25} & & A_{27} \\ & & A_{33} & & & A_{36} & A_{37} \\ & & & A_{44} & & A_{46} & A_{47} \\ A_{15}^T & A_{25}^T & & & A_{55} & & A_{57} \\ & & A_{36}^T & A_{46}^T & & A_{66} & A_{67} \\ A_{17}^T & A_{27}^T & A_{37}^T & A_{47}^T & A_{57}^T & A_{67}^T & A_{77} \end{pmatrix}.$$

We may continue this process until no further dissection can be accomplished, obtaining a *nested dissection ordering* with the associated vertex partitioning  $S_i^j$ , where  $j = 1, 2, \dots, 2^i, i = 0, 1, \dots, 2l, l = k - 1$  and  $S_{2^i}^j = R_{2^i}^j$ . A detailed analysis of the fill-in and operation counts of this nested dissection approach may be found in George and Liu [1981a]. We state the main result in the following theorem.

**THEOREM 2.1.** *The number of nonzeros in the Cholesky factor  $R$  associated with a regular  $n$  by  $n$  grid which is ordered by the nested dissection algorithm is given by  $\frac{31}{4}n^2 \log_2 n + O(n^2)$ , and the number of arithmetic operations required during the factorization is given by  $\frac{829}{84}n^3 + O(n^2 \log_2 n)$ .*

These bounds were proven optimal in the asymptotic sense; that is, the order of magnitude in these bounds cannot be reduced<sup>1</sup> (George and Liu [1981a], Hoffman et al. [1973]).

An *incomplete* nested dissection ordering and partitioning is obtained when  $l < k - 1$ . An example with  $k = 4$  and  $l = 2$  is given in Fig. 2.1.

In the following we do not distinguish between vertices of the grid and variables in the associated linear system, since there is a 1-1 correspondence between them. Thus, the set (block)  $S_i^j$  refers to a set of variables which are eliminated together, and  $i > i_1$  implies that variables in  $S_i^j$  are eliminated before those in  $S_{i_1}^{j_1}$ . Sets  $S_i^j$  having the same subscript  $i$  will be said to be “at the same level  $i$ ,” and we make no assumptions about the order in which sets at the same level are eliminated.

Blocks for which  $i = 2l$  will be called “last-level blocks,” and those for which  $i < 2l$  will be called “separator blocks”, for obvious reasons. The ordering of the vertices in the last-level blocks is unspecified, and we assume that they are ordered as though the dissection has been completed, i.e., by nested dissection. It should also be clear that both last-level blocks and separator blocks can be classified into *corner*, *side*, and *interior* blocks in a natural way. For example, in Fig. 2.1 there are 4 last-level corner blocks, 8 last-level side blocks, and 4 last-level interior blocks. There are no interior separator blocks because  $l$  is too small. Fig. 2.2 shows the structure of the block columns of the Cholesky factor for different types of blocks. For more details, see (George, Poole and Voigt [1978]).

A *rooted separator tree* is naturally associated with a nested dissection partitioning as follows: the internal nodes of the tree are the separators, and a node  $S_{i+1}^k$  has the

<sup>1</sup> Here, we consider the standard Cholesky algorithm. However, if the method of Strassen [1969] is used, we may achieve an  $O(n^{\log_2 7})$  operation count (Rose and Whitten [1976]).

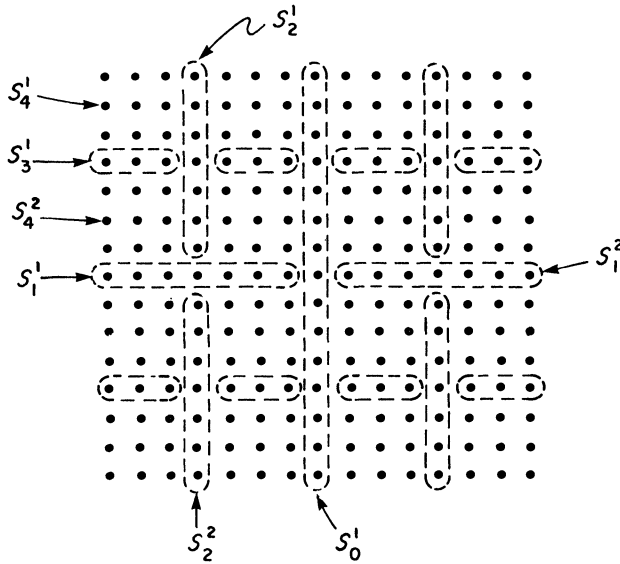


FIG. 2.1. An incomplete nested dissection partitioning of an  $n$  by  $n$  grid, where  $l=2$ ,  $n=2^k-1$ , and  $k=4$ .

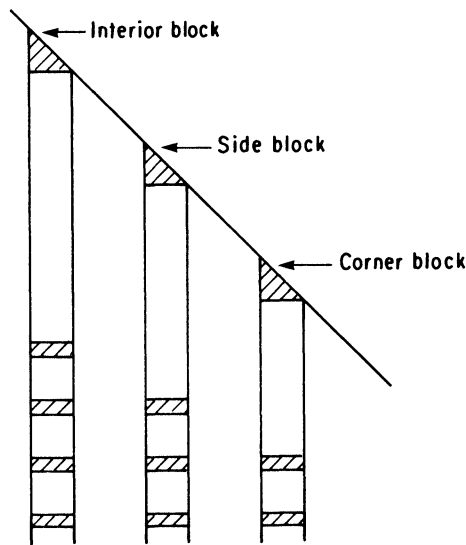


FIG. 2.2. The block structure of the Cholesky factor corresponding to different types of blocks.

node  $S_i^j$  as its *father* if and only if  $R_{i+1}^k \subseteq R_i^j$ . The tree is rooted at  $S_0$ , and its leaves are the last-level blocks (Gilbert [1980]).

**3. Factorization of partitioned matrices using auxiliary storage.** In this section, we present an algorithm which uses sequential access auxiliary storage files for the solution of the linear system of equations  $Ax = b$ , where  $A$  is an  $N$  by  $N$ , sparse, symmetric and positive definite matrix. Most of our discussion will be centered around the factorization step, since this is usually the most expensive and time consuming part of the overall solution process.

Some of the basic observations about the factorization of partitioned sparse matrices are best illustrated using the following simple example. Consider a block 2 by 2 matrix

$$(3.1) \quad A = \begin{pmatrix} B & V \\ V^T & \bar{C} \end{pmatrix},$$

where  $B$  and  $\bar{C}$  are  $r$  by  $r$  and  $s$  by  $s$  respectively, and  $r + s = N$ . The Cholesky factor, correspondingly partitioned, is given by

$$(3.2) \quad R = \begin{pmatrix} R_B & W \\ 0 & R_C \end{pmatrix},$$

where  $R_B$  and  $R_C$  are the Cholesky factors of  $B$  and  $C$  respectively, with  $C = \bar{C} - V^T B^{-1} V$  and  $W = R_B^{-T} V$ .<sup>2</sup> The submatrix  $C = \bar{C} - V^T B^{-1} V = \bar{C} - W^T W$  is often referred to as the *Schur complement* of  $B$  in  $A$ . Note that the submatrices  $R_B$ ,  $W$  and  $W^T W$  can be computed in the *absence* of  $\bar{C}$  (or parts thereof). This is the fundamental observation upon which the frontal method (Irons [1970]) and the generalized element method (Speelpenning [1978]) are based. In the context of sparse matrices, the submatrix  $V$  usually has only a few nonnull columns (say  $\mu$ ), hence the *effective size* of  $\bar{C}$  (i.e., the actual size of the change  $W^T W$ ) will be only  $\mu$  by  $\mu$ .

We now describe how these basic observations are exploited by our method. Let  $A$  be a  $p$  by  $p$  partitioned matrix, with submatrices (blocks)  $A_{ij}$ ,  $1 \leq i, j \leq p$ . This partitioning *induces* an *ordered partition*  $\Omega$  on the set  $X$  of vertices of the corresponding graph  $G = G^A = (X, E)$ , namely,

$$\Omega = (Y_1, Y_2, \dots, Y_p),$$

where  $\cup_{i=1}^p Y_i = X$  and  $Y_i \cap Y_j = \emptyset$  for  $i \neq j$ , and  $Y_i$  is the set of vertices corresponding to the rows and columns of  $A_{ii}$ . The factorization of  $A$  is carried out block by block, and we use  $A_k$  to denote the part of the matrix remaining to be factored after the variable sets  $Y_1, Y_2, \dots, Y_{k-1}$  have been eliminated, where  $A_1 = A$ . The submatrices of  $A_k$  are as shown in (3.3).

$$(3.3) \quad A_k = \begin{pmatrix} A_{k,k}^{(k)} & A_{k,k+1}^{(k)} & \cdots & A_{k,p}^{(k)} \\ & A_{k+1,k+1}^{(k)} & \cdots & A_{k+1,p}^{(k)} \\ & & \ddots & \vdots \\ & \text{symmetric} & & A_{p,p}^{(k)} \end{pmatrix}.$$

At step  $k$ , when variables  $Y_k$  are to be eliminated, we denote by  $\partial Y_k$  the set of columns of  $A_k$  which are not in  $Y_k$ , but have nonzeros in rows corresponding to  $Y_k$ . This set  $\partial Y_k$ , usually referred to as the boundary of  $Y_k$ , has a simple graph theoretical interpretation: the set  $\partial Y_k$  is the set of vertices adjacent to the set  $Y_k$  in the graph  $G^{A_k}$ . This notation is illustrated in Fig. 3.1. For example,  $\partial Y_1 = \{8, 10, 11, 12\}$ , since  $a_{2,8}$ ,  $a_{3,10}$ ,  $a_{5,12}$ , and  $a_{6,11}$  are nonzero in  $A_1 = A$ . The set  $\partial Y_2 = \{11, 12\}$ , because  $a_{8,11}$  and  $a_{8,12}$  become nonzeros in  $A_2$ ; that is, after the elimination of the variable set  $Y_1$ .

Now consider the 3 by 3 example in Fig. 3.2, which is intended to illustrate in general how the computation proceeds. We assume that initially the rows of  $A$  are stored on a sequential access file, which will be usually called the input file. At the first step, rows corresponding to  $Y_1$  and  $\partial Y_1$  are read from the input file, yielding a

<sup>2</sup> Note that we usually do not compute inverses; instead we solve the appropriate linear system. For example to compute  $D = B^{-1} V$ , we solve  $BD = V$ .

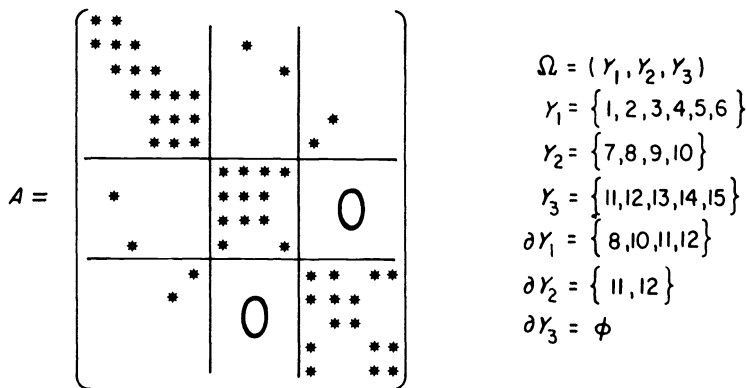


FIG. 3.1. Example of a partitioned matrix.

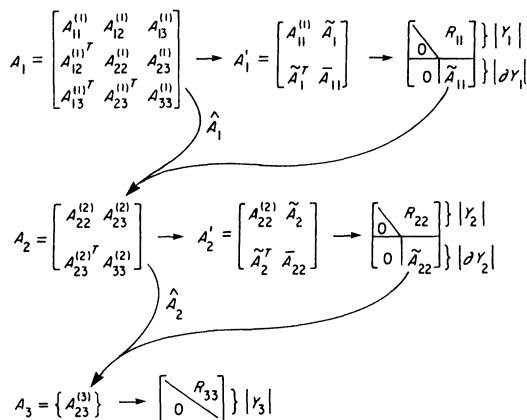


FIG. 3.2. Block factorization of a 3 by 3 partitioned matrix.

matrix  $A'_1$ , which is a “compressed form” of  $A (= A_1)$ :

$$(3.4) \quad A'_1 = \begin{pmatrix} A_{11}^{(1)} & \tilde{A}_1 \\ \tilde{A}_1^T & \bar{A}_{11} \end{pmatrix}.$$

The first  $|Y_1|$  variables are then eliminated, yielding the first  $|Y_1|$  rows of  $R$  which can be output on an “ $R$ -file”. Note that the last  $|\partial Y_1|$  columns of  $R$  are compressed in the same manner as  $\tilde{A}_1$  is with respect to  $(A_{12}^{(1)}|A_{13}^{(1)})$ . We also assume that in the process of reading the data from the input file in order to create the compressed form of  $A_1$  shown in (3.4), that the rows in the matrix

$$(3.5) \quad \begin{pmatrix} A_{22}^{(1)} & A_{23}^{(1)} \\ A_{23}^{(1)T} & A_{33}^{(1)} \end{pmatrix}$$

which correspond to  $\partial Y_1$  are modified to reflect the fact that parts of its rows and columns have been placed in  $\bar{A}_{11}$ . We denote this new matrix by  $\hat{A}_1$ . The method of managing files to be described later shows that this is easy to accomplish.

The matrix  $\tilde{A}_{11} = \bar{A}_{11} - \tilde{A}_1^T A_{11}^{(1)-1} \tilde{A}_1$ , having  $|\partial Y_1|$  rows and columns, must be “expanded” and added into  $\hat{A}_1$ , producing  $A_2$  so that the next major step of the factorization involving the elimination of the variables  $Y_2$  can proceed. In practice we

defer incorporating these modifications until the block to be modified is about to be factored. The matrix  $\hat{A}_{11}$  is written onto the end of the file containing  $\hat{A}_1$ , as indicated in Fig. 3.2. The second and third steps of the factorization are carried out in a similar fashion as depicted in Fig. 3.2.

In general, for a partitioning  $\Omega$  with  $|\Omega| = p$ , we will have  $p$  major steps similar to those depicted in Fig. 3.2, and will have to process a sequence of progressively smaller matrices  $A_1, A_2, \dots, A_p$ . Four sequential access files are used, and their names and functions are now described.

1. *R*-file: accepts the rows of *R* as they are computed.
2. *A*-file: at the beginning of the  $k$ th major step of the computation, this file contains the rows of the matrix  $A_k$  in arbitrary order. This file will occasionally be referenced as the input file.
3. *A'*-file: during the  $k$ th step of the computation, this file contains the matrix  $A'_k$ .
4.  $\hat{A}$ -file: at the beginning of the  $k$ th step, this file is empty. The *A*-file is read,  $A'_k$  is written on the *A'*-file, and  $\hat{A}_k$  is written on the  $\hat{A}$ -file. After the computation of  $R_{kk}$  is done, the rows of  $\tilde{A}_{kk}$  are written on this file, and it then becomes the *A*-file in preparation for the next major step of the factorization.

The reader is reminded that  $\tilde{A}_{kk}$  and  $\hat{A}_k$  are not explicitly combined (or assembled) until such time as the rows they share appear in some  $A'_r$ , with  $r > k$ . This fact must be accounted for in determining the I/O traffic generated during each major step of the factorization algorithm.

With the roles of the files in mind, we now give a step by step description of the factorization algorithm.

#### ALGORITHM 3.1.

**For**  $k = 1, 2, \dots, p$  **do**

1. Rewind the *A*-file, *A'*-file, and the  $\hat{A}$ -file. Read the rows of  $A_k$  from the *A*-file one by one and for each do the following:
  - 1.1 If the row is in  $Y_k$ , write it on the *A'*-file.
  - 1.2 If the row is in  $\partial Y_k$ , write its elements that are in  $Y_k \cup \partial Y_k$  as a row on the *A'*-file, and write the remaining elements as a row on the  $\hat{A}$ -file.
  - 1.3 If the row is not in  $Y_k \cup \partial Y_k$ , write it on the  $\hat{A}$ -file.
2. Using the *A'*-file as input, eliminate the variables in  $Y_k$ . Then write the rows of  $R_{kk}$  on the *R*-file, and those of  $\tilde{A}_{kk}$  on the  $\hat{A}$ -file.
3. Reverse the names of the *A*-file and the  $\hat{A}$ -file.

If the above algorithm is executed successfully, we terminate with the rows of the Cholesky factor *R* stored on the *R*-file, and they may be used in the forward and backward substitution steps to obtain the solution vector  $x$ .

The above algorithm is similar to that described in George et al. [1981b] in connection with solving large sparse linear least squares problems. Note that our algorithm may be viewed as a particular implementation of the generalized element method of Speelpenning [1978], where the elimination of variables is carried out one *block* at a time, instead of one *variable* at a time. Moreover, the auxiliary storage files are assumed to allow only sequential access. The use of sequential access is preferable to random access, since the facilities for random access in programming languages are usually machine dependent. Also note that no particular partitioning is assumed. This algorithm may very well be used in conjunction with any *good* partitioning, provided

that the partitioning is sufficiently fine that each block fits in the available amount of main memory. In the following section we consider one such partitioning which is provided by incomplete nested dissection. Note that the I/O traffic generated in step 1 of Algorithm 3.1 is relatively large, since in each factorization step we read the whole input file.

#### 4. Analysis of the I/O traffic.

**4.1. Preliminaries.** In this section we give a detailed analysis of the amount of data that must be transferred between main memory and auxiliary storage when Algorithm 3.1 is applied to a linear system of equations associated with the model  $n$  by  $n$  grid problem. We assume that the mesh vertices (and the related matrix) are partitioned using an incomplete nested dissection partitioning. The coefficient matrix  $A$  is  $n^2$  by  $n^2$  and has approximately  $9n^2$  nonzeros. However, since  $A$  is symmetric, we need only store its diagonal and upper triangular part, yielding about  $5n^2$  nonzero elements. In what follows we assume that the rows of the upper triangular part of  $A$  (including the diagonal) are stored in *arbitrary* order on a sequential access file, and that for some integer  $l$ , a  $2l$ -level incomplete nested dissection partitioning and ordering is provided, as described in § 2. A simplified version of the analysis given here may be found in George and Rashwan [1982].

The following assumptions are made:

i) The access to the files on auxiliary storage is sequential. This implies that for each partition member processed, we have to read the whole input file, picking up those rows which play a role in the current elimination step.

ii) The elimination process is carried out *one block* at a time irrespective of the block size. This will have the obvious effect of increasing the I/O traffic compared to what might result from eliminating more than one of the smaller blocks at a time (recall that in the nested dissection partitioning of § 2, the size of the separators varies widely across the levels). More important is the implication that we must have enough main memory to carry out the elimination of the largest block (this is the block corresponding to a separator at level one, as can be easily verified by the reader). Thus, we require  $O(n^2)$  main memory, i.e., an amount of main memory proportional to the number of variables in the problem.

iii) The I/O traffic will be measured by the number of nonzeros transferred between main memory and auxiliary storage.

iv) Block elimination is carried out in a level-by-level fashion. Blocks at level  $2l$  (last-level blocks) are eliminated first, followed by blocks at level  $2l-1, \dots$ , and finally block  $S_0$  (the top-level separator block) is eliminated. This is just one possible order for block elimination among others which are equivalent as far as storage requirements and operation counts are concerned.

It should be clear from the description of Algorithm 3.1 that one of the crucial factors in determining the I/O traffic caused by the elimination of  $Y_k$  will be the size of the set  $\partial Y_k$ . In order to see how this is done, consider Fig. 4.1, which is intended to denote part of the  $n$  by  $n$  grid.

A full explanation would take us too far afield here, but given  $Y_k$ , the set  $\partial Y_k$  is precisely the set of vertices which can be reached from a vertex in  $Y_k$  via paths in the mesh having vertices numbered *less* than any of those in  $Y_k$ . Thus, in Fig. 4.1,  $\partial Y_c = Z_a \cup Z_b$  because one can reach vertices in  $Z_a$  or  $Z_b$  via paths in  $Y_a$  or  $Y_b$ . For a complete explanation of this "recipe," see George and Liu [1981a]. The matrices and the I/O traffic generated by the elimination of the vertex sets  $Y_a$ ,  $Y_b$  and  $Y_c$  are depicted in Fig. 4.2.

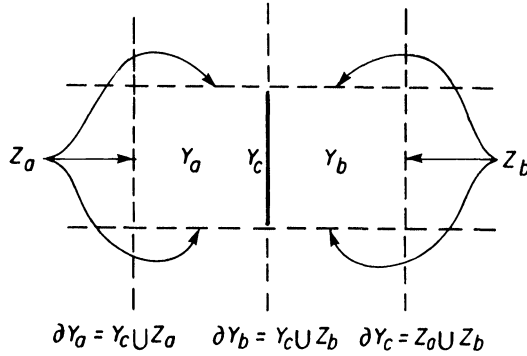


FIG. 4.1. A subgrid showing the "boundaries" of several sets to be eliminated.

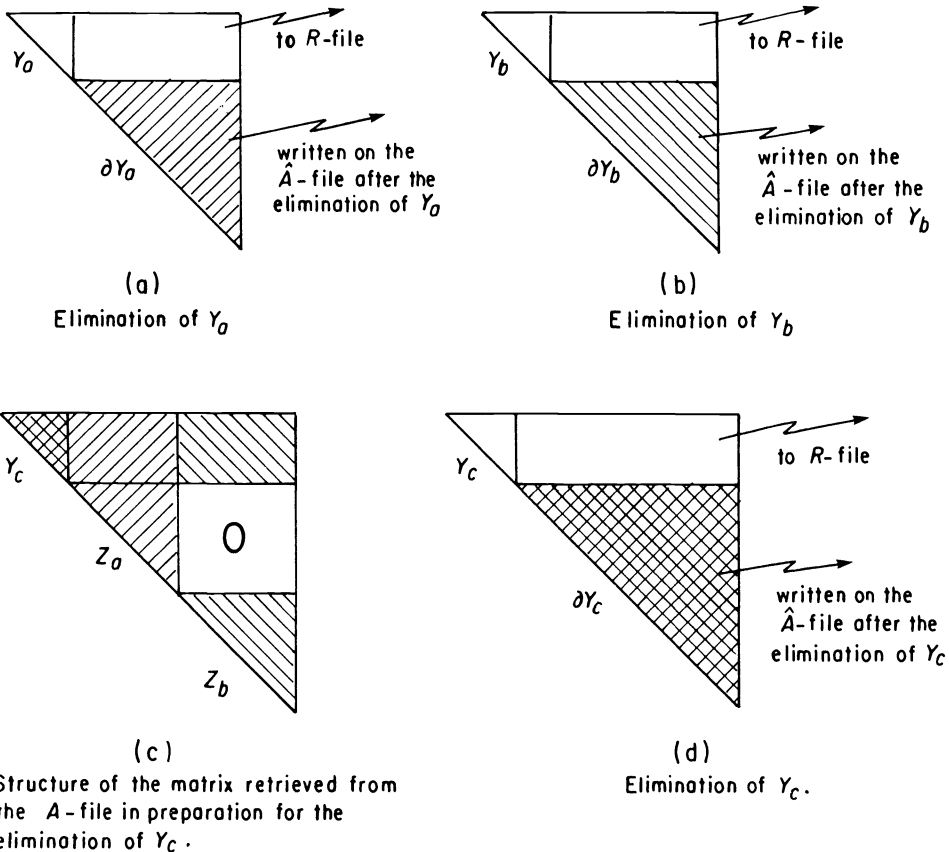


FIG. 4.2. Elimination of vertex sets  $Y_a$ ,  $Y_b$  and  $Y_c$  of Fig. 4.1.

Note that the elimination of  $Y_c$  may proceed even if the rows corresponding to  $\partial Y_c = Z_a \cup Z_b$  are not retrieved from the input file as shown in Fig. 4.3(a), since the rows corresponding to  $Y_c$  alone need be fully assembled.

It may be seen that the elimination of a block  $Y_c$  in the way shown in Fig. 4.2(c) allows for the immediate assembly of the modification submatrix (Schur complement) resulting from the current elimination step with corresponding submatrices resulting from previous elimination steps. On the other hand, the elimination of  $Y_c$  in the absence



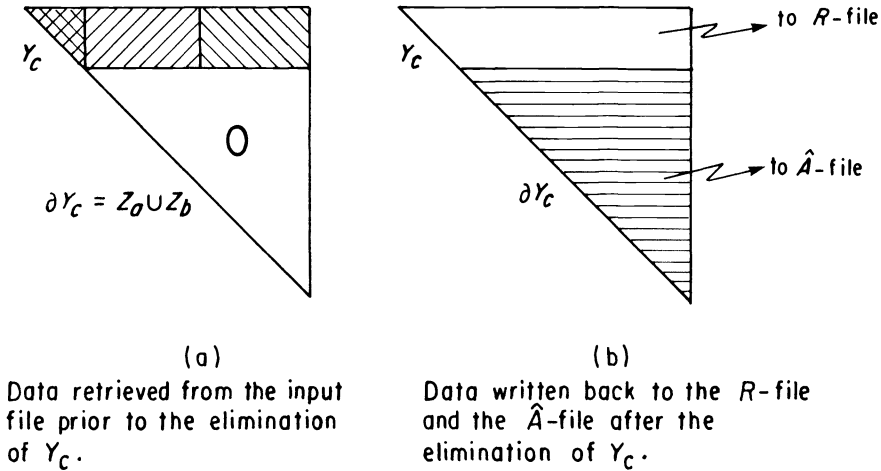


FIG. 4.3. An alternative way to eliminate  $Y_c$ .

of rows corresponding to  $\partial Y_c$  will eventually result in an input file having the three full submatrices corresponding to  $Z_a$ ,  $Z_b$ , and  $\partial Y_c$  stored separately. Thus, using the approach outlined in Fig. 4.2 seems to be advantageous in reducing the size of the input file, and consequently the I/O traffic. It is also interesting to note that the analysis of the I/O traffic becomes much harder and less transparent in the other case, since we have to *remember* not only the results of eliminations in the proceeding level, but also those resulting from all previous levels.

Also observe that the input file is *regenerated* at each level. This may be seen as follows. Consider the elimination of blocks at level  $j$ . As each block is eliminated, the related Schur complements from eliminations at level  $j+1$  are *deleted* from the input file, and subsequently *replaced* by a *new* Schur complement. Thus, when all level  $j$  blocks have been eliminated, the input file will consist entirely of new Schur complements which have been generated at level  $j$ . Note that this observation is not true unless we use the approach indicated in Fig. 4.2.

The above observation is useful in estimating the size of the input file at each level. From now on we will assume that when a block  $Y_c$  is to be eliminated, the rows corresponding to  $\partial Y_c$  are retrieved from the input file in addition to those corresponding to  $Y_c$ .

In order to establish our results about the I/O traffic, we need to estimate the size of the input file as the block elimination proceeds. To this end we study the *changes* in the size of the input file due to the elimination of different types of blocks.

It is helpful to have some notation for certain quantities which are used throughout this section.

$F_i$ —the size of the input file (= number of nonzero elements) *before* eliminating any block at level  $i$ .

$V_i$ —the volume of traffic generated due to the elimination of all blocks at level  $i$ .

$\kappa_i$ —the change in the size of the input file (i.e., the  $A$ -file) resulting from the elimination of a block at level  $i$ .

$\beta_i$ —the size (i.e., number of vertices) of the separator block at level  $i$ .

$\eta_i$ —the number of blocks at level  $i$ .

It is well known that the blocks in the Cholesky factor corresponding to separators are *full* (George [1977]). In this connection we note that the number of nonzeros in

the upper triangular part of a full symmetric submatrix (i.e., a clique) of size  $d$  is given by  $\gamma(d) = \frac{1}{2}d(d+1)$ .

In Lemma 4.1 we give an estimate of the change in the input file size due to the elimination of a separator block.

LEMMA 4.1. *The elimination of a separator block  $Y$  of size  $\lambda$ , which is bordered by boundary segments of size  $\lambda_1$ , and  $\lambda_2$  (as shown in Fig. 4.4(a)) will cause the input file to change in size by  $\kappa(\lambda, \lambda_1, \lambda_2, d) = -\gamma(\lambda + \lambda_1) - \gamma(\lambda + \lambda_2) + \gamma(d)$ , where  $d = |\partial Y|$ .*

*Proof.* Recall that two full Schur complement matrices corresponding to  $Y \cup Z_1$  and  $Y \cup Z_2$  have already been created and stored in the input file before block  $Y$  is about to be eliminated. The elimination of  $Y$  is preceded by retrieving and deleting these two submatrices in Fig. 4.4(b) from the  $A$ -file. After the elimination of  $Y$ , the full submatrix in Fig. 4.4(c) is appended to the end of the  $\hat{A}$ -file. Hence the input file shrinks by  $\gamma(\lambda + \lambda_1) + \gamma(\lambda + \lambda_2)$ , and grows by  $\gamma(d)$ , where  $\lambda = |Y|$ ,  $\lambda_1 = |Z_1|$ ,  $\lambda_2 = |Z_2|$ , and  $d = |\partial Y|$ . This establishes our result.  $\square$

Note that  $d \neq \lambda_1 + \lambda_2$ , even though  $\partial Y = Z_1 \cup Z_2$ , because the sets  $Z_1$  and  $Z_2$  are not disjoint. The points denoted by  $\bullet$  in Fig. 4.4(a) are common to  $Z_1$  and  $Z_2$ . These are the points of intersection of  $Y$  and  $\partial Y$ . The result of Lemma 4.1 is useful, since it depends only on the size of block  $Y$  and the sizes of the associated boundary segments.

For a  $2l$ -level incomplete nested dissection partitioning, we have  $2 \times 2^{2l} - 1$  blocks,  $2^{2l}$  of which are last-level blocks, and  $2^{2l} - 1$  are separator blocks. The following

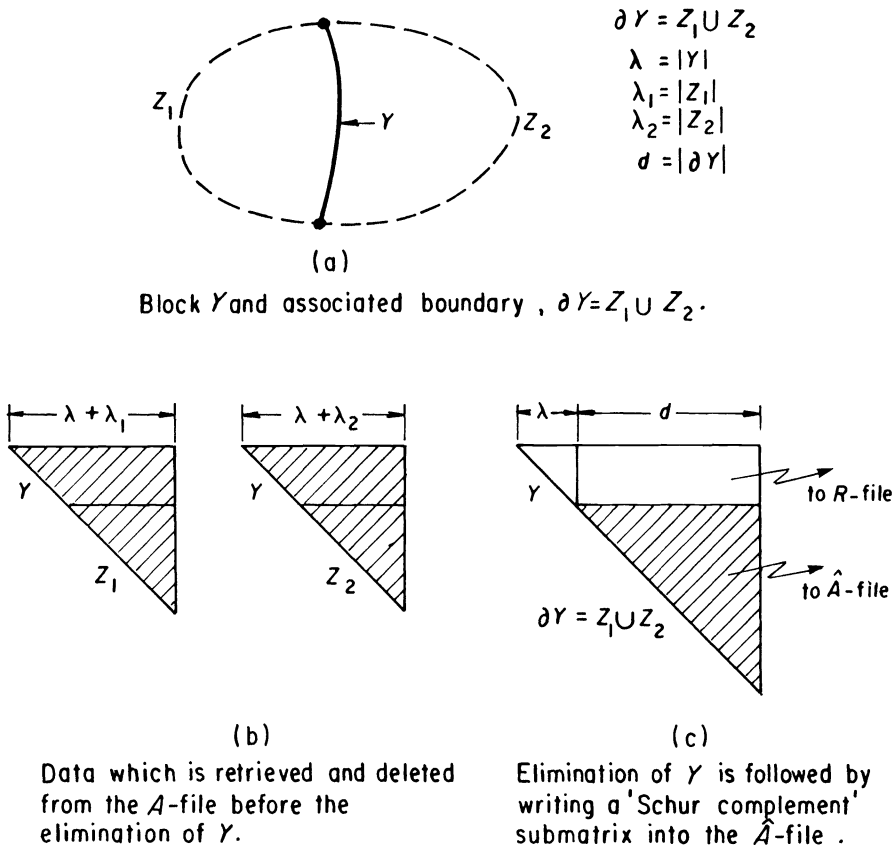


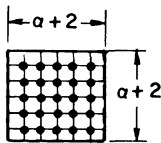
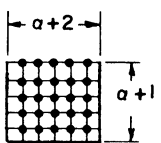
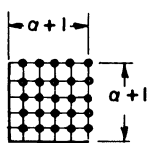


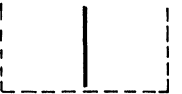
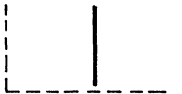


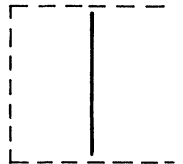

FIG. 4.4. Data transfer between main memory and auxiliary storage due to the elimination of  $Y$ .

observations about the blocks at different levels can be easily established:

a) The last-level blocks (those at level  $2l$ ) are of size  $\alpha$  by  $\alpha$ , where  $\alpha = (n+1)/2^l - 1$ , and there are  $2^{2l}$  such blocks. These blocks are classified into corner, side, and interior blocks. There are 4 corner blocks,  $4(2^l - 2)$  side blocks, and  $(2^l - 2)^2$  interior blocks.

b) The separator blocks at level  $2i+1$ ,  $i=1, 2, \dots, l-1$ , are of size  $\beta_{2i+1} = (n+1)/2^{i+1} - 1$ . These blocks are classified into corner, side, and interior blocks. The side blocks are further classified into two types denoted by  $S_1$  and  $S_2$ , respectively.

TABLE 4.1  
Basic characteristics of different types of blocks.

 <p><math>d = 4\alpha + 4</math> <math>\eta = (2^l - 2)^2</math></p> <p>Interior block</p>	 <p><math>d = 3\alpha + 2</math> <math>\eta = 4(2^l - 2)</math></p> <p>Side block</p>	 <p><math>d = 2\alpha + 1</math> <math>\eta = 4</math></p> <p>Corner block</p>	
<p>Last level blocks of a size <math>\alpha</math> by <math>\alpha</math>.</p>			
 <p><math>\lambda = \beta_{2i+1}</math> <math>\lambda_1 = \lambda_2 = 3\beta_{2i+1} + 4</math> <math>d = 6\beta_{2i+1} + 6</math> <math>\eta = 2(2^i - 2)(2^i - 1)</math></p> <p>Interior block</p>	 <p><math>\lambda = \beta_{2i+1}</math> <math>\lambda_1 = 3\beta_{2i+1} + 4</math> <math>\lambda_2 = 2\beta_{2i+1} + 2</math> <math>d = 5\beta_{2i+1} + 4</math> <math>\eta = 4(2^i - 1)</math></p> <p>Side block (<math>S_1</math>)</p>	 <p><math>\lambda = \beta_{2i+1}</math> <math>\lambda_1 = \lambda_2 = 2\beta_{2i+1} + 2</math> <math>d = 4\beta_{2i+1} + 3</math> <math>\eta = 2(2^i - 2)</math></p> <p>Side block (<math>S_2</math>)</p>	 <p><math>\lambda = \beta_{2i+1}</math> <math>\lambda_1 = 2\beta_{2i+1} + 2</math> <math>\lambda_2 = \beta_{2i+1} + 1</math> <math>d = 3\beta_{2i+1} + 2</math> <math>\eta = 4</math></p> <p>Corner block</p>
<p>Separator blocks at level <math>2i+1</math>.</p>			
 <p><math>\lambda = \beta_{2i}</math> <math>\lambda_1 = \lambda_2 = 2\beta_{2i} + 3</math> <math>d = 4\beta_{2i} + 4</math> <math>\eta = (2^i - 2)^2</math></p> <p>Interior block</p>	 <p><math>\lambda = \beta_{2i}</math> <math>\lambda_1 = \lambda_2 = \frac{3}{2}(\beta_{2i} + 1)</math> <math>d = 3\beta_{2i} + 2</math> <math>\eta = 2(2^i - 2)</math></p> <p>Side block (<math>S_1</math>)</p>	 <p><math>\lambda = \beta_{2i}</math> <math>\lambda_1 = 2\beta_{2i} + 3</math> <math>\lambda_2 = \beta_{2i} + 1</math> <math>d = 3\beta_{2i} + 2</math> <math>\eta = 2(2^i - 2)</math></p> <p>Side block (<math>S_2</math>)</p>	 <p><math>\lambda = \beta_{2i}</math> <math>\lambda_1 = \frac{1}{2}(\beta_{2i} + 1)</math> <math>\lambda_2 = \frac{3}{2}\beta_{2i} + 1</math> <math>d = 2\beta_{2i} + 1</math> <math>\eta = 4</math></p> <p>Corner block</p>
<p>Separator blocks at level <math>2i</math>.</p>			

c) The separator blocks at level  $2i, i = 1, 2, \dots, l-1$  are of size  $\beta_{2i} = (n+1)/2^i - 1$ . These blocks are classified in the same way as those at level  $2i+1$ .

The relevant features of the different types of blocks are summarized in Table 4.1, where solid lines are used to represent separators and broken lines are used to represent the associated boundary. Here, and elsewhere, we use the letter  $C$  to denote corner blocks, the letters  $S_1$  and  $S_2$  to denote the two kinds of side blocks, and finally  $I$  to denote interior blocks. Note that the separators at levels 0 and 1 do not follow the above classification, hence they are treated separately.

**4.2. The I/O traffic.** In this section we give a detailed analysis of the I/O traffic generated during the execution of Algorithm 3.1. The size of the input file plays an important role in deriving a bound on the I/O traffic. To this end, we give an estimate of the size of the input file *before* eliminating blocks at any level.

LEMMA 4.2. *The size of the input file before eliminating separator blocks at levels  $2i+1$  and  $2i$  is given by*

$$(4.1) \quad F_{2i+1} = 8n^2 - 7n^2 2^{-i} + n^2 2^{-2i} + 4n2^i + O(n), \quad i = l-1, l-2, \dots, 0,$$

and

$$(4.2) \quad F_{2i} = 9n^2 - \frac{21}{2} n^2 2^{-i} + 2n^2 2^{-2i} + 3n2^i + O(n), \quad i = l-1, l-2, \dots, 1.$$

*Proof.* Recall that the input file at level  $2i$  consists entirely of those Schur complements resulting from the elimination at level  $2i+1$ . Hence,  $F_{2i}$  is given by

$$F_{2i} = \eta_{2i+1}^C \gamma(d_{2i+1}^C) + \eta_{2i+1}^{S_1} \gamma(d_{2i+1}^{S_1}) + \eta_{2i+1}^{S_2} \gamma(d_{2i+1}^{S_2}) + \eta_{2i+1}^I \gamma(d_{2i+1}^I).$$

Here,  $\gamma(d) = \frac{1}{2}d(d+1)$ . Simplifying this using the symbolic computation system MAPLE (Geddes et al. [1982]) as an aid, we obtain (4.2). The estimate for  $F_{2i+1}$  is obtained in a similar fashion.  $\square$

The expression for  $F_0$  is obtained separately, and is given by  $F_0 = 2\gamma(n) = n^2 + n$ .

Before proceeding further, it is interesting to note that the above result suggests that the size of the input file is  $O(n^2)$  at all levels. This means that the total auxiliary storage required by our algorithm is dominated by that required to store the Cholesky factor  $R$ , which is  $O(n^2 \log n)$ .

In the following we count only the number of nonzeros which are *read* during the execution of step 1 of Algorithm 3.1. Recall that this was the largest portion of the input traffic. Results similar to the ones given here may be similarly established for the output traffic.

The input traffic generated at any level is given by assuming the *current* sizes of the input files *before* the elimination of each block in the level under consideration. It should be obvious that the order of block elimination at each level will have some effect on the volume of the I/O traffic generated. It can be easily verified that the volume of the I/O traffic may be reduced if we eliminate corner blocks first, followed by  $S_2$  side blocks, followed by  $S_1$  side blocks, followed finally by interior blocks. An account of the input traffic generated during the elimination of separator blocks is given by Lemma 4.3.

LEMMA 4.3. *The input traffic generated from the elimination of separator blocks at levels  $2i + 1$  and  $2i$  is given by*

$$(4.3) \quad V_{2i+1} = 17n^2 2^{2i} - 21n^2 2^i + 13n^2 - \frac{41}{4}n^2 2^{-i} + \frac{11}{2}n^2 2^{-2i} - \frac{7}{2}n^2 2^{3i} + O(n^2 2^i),$$

$$(4.4) \quad V_{2i} = \frac{17}{2}n^2 2^{2i} - 14n^2 2^i + 11n^2 - \frac{41}{4}n^2 2^{-i} + 7n^2 2^{-2i} + \frac{5}{2}n^2 2^{3i} + O(n^2 2^i), \quad i = l-1, l-2, \dots, 1.$$

*Proof.* Let  $F_{2i,x}$  be the size of the input file at level  $2i$  before eliminating any block of type  $X$ , where  $X = C, S_2, S_1, \text{ or } I$ . Carrying the elimination in the order indicated earlier, we have  $F_{2i,C} = F_{2i}$ ,  $F_{2i,S_2} = F_{2i,C} - \eta_{2i}^C \kappa_C$ ,  $F_{2i,S_1} = F_{2i,S_2} - \eta_{2i}^{S_2} \kappa_{S_2}$ , and finally  $F_{2i,I} = F_{2i,S_1} - \eta_{2i}^{S_1} \kappa_{S_1}$ , where  $\kappa$  is given by Lemma 4.2. The input traffic is then given by the following summation:

$$V_{2i} = \sum_{X=C,S_2,S_1,I} \sum_{j=1}^{\eta_{2i}^X} (F_{2i,x} + (j-1)\kappa_x).$$

The expression for  $V_{2i+1}$  is obtained in the same manner.  $\square$

$V_0$  and  $V_1$  are computed separately, and are given by  $V_0 = n^2 + n$ , and  $V_1 = \frac{7}{2}(n^2 + n)$ .

Lemma 4.3 has the following simple and intuitive interpretation. The input traffic generated at any level is roughly proportional to the *product* of the *number* of blocks in that level and the *average size* of the input file in the same level. It may also be seen that  $V_{2i+1} > V_{2i}$ ,  $i = l-1, \dots, 1$ . In particular, note that the leading term in the I/O traffic is halved as we move from one level to the next level down in the elimination process. This is mainly because the number of blocks is also halved as we move down the levels. This implies that most of the I/O traffic is generated during the elimination of the small separators. Hence, a substantial reduction in the I/O traffic may be achieved by combining more than one of the small separators together, and effecting their elimination as one block.

The input traffic due to the elimination of all separator blocks is given by adding up the contributions from all levels, and is given by

$$(4.5) \quad V_{\text{sep}} = \frac{17}{2}n^2 2^{2l} - 35n^2 2^l + \frac{1}{6}n^2 + 41n^2 2^{-l} + 24n^2 l - \frac{50}{3}n^2 2^{-2l} + \frac{19}{14}n^2 2^{3l} + O(n^2 2^{2l}).$$

Now, we turn our attention to the elimination of last-level blocks. Let  $g(\alpha_1, \alpha_2) = 5\alpha_1\alpha_2 - 3(\alpha_1 + \alpha_2) + 2$  be the number of nonzeros in the upper triangular part (including the diagonal) of the symmetric matrix corresponding to an  $\alpha_1$  by  $\alpha_2$  grid. Using this fact, we can derive results similar to those established for the separator blocks.

LEMMA 4.4. *The change in the size of the input file due to the elimination of a last-level block of size  $\alpha_1$  by  $\alpha_2$ , whose boundary is  $\partial Y$ , is given by*

$$(4.6) \quad \kappa(\alpha_1, \alpha_2, d) = -g(\alpha_1, \alpha_2) + \gamma(d),$$

where  $d = |\partial Y|$ .

LEMMA 4.5. *The input traffic generated during the elimination of all last-level blocks is given by*

$$(4.7) \quad V_{2l} = \frac{13}{2}n^2 2^{2l} - 14n^2 2^l + \frac{61}{2}n^2 - 49n^2 2^{-l} + 46n^2 2^{-2l} - n^2 2^{3l} + O(n^2 2^{2l} + 2^{4l}).$$

The total input traffic generated during the factorization process may be obtained by adding  $V_{2l}$  and  $V_{\text{sep}}$ . The result is stated as the following theorem.

**THEOREM 4.6.** *The total input traffic generated during the factorization of the matrix associated with an  $n$  by  $n$  grid which is partitioned by a  $2l$ -level incomplete nested dissection partitioning is given by*

$$(4.8) \quad V_{\text{tot}} = 15n^22^{2l} - 49n^22^l + 24n^2l + \frac{92}{3}n^2 - 8n^22^{-l} + n^22^{-2l} + \frac{5}{14}n2^{3l} + O(n2^{2l} + 2^{4l}).$$

It may be seen that the total input traffic is roughly divided between the last-level blocks and the separator blocks, which might be expected, since the blocks are roughly divided among the last-level and the rest of the levels. Observe that the total input traffic is  $O(n^22^{2l}) = O(n^2p)$ , where  $p = |\Omega|$  is the number of partition members.

It is evident from equation (4.8) that using a partitioning which is finer than necessary will increase the I/O traffic substantially. Consider the case of complete nested dissection, where  $l = \log(n + 1) - 1$ . We can see that the I/O is  $O(n^4)$ , which is an order of magnitude higher than the  $O(n^3)$  arithmetic operations. Hence, it is imperative to have  $l$  (or equivalently  $p$ ) as small as possible, consistent with the available amount of main memory. We will shortly see that using  $O(n^2)$  main memory allows us to achieve an  $O(n^2 \log n)$  bound on the I/O traffic.

The I/O traffic generated during the execution of Algorithm 3.1 consists of the following components:

a) I/O generated during step 1, which may be considered as a block *identification* step, where the data belonging to a single block is retrieved from the input file and stored separately on the  $A'$ -file in preparation for the factorization. The output traffic generated during this step is *equal* to the input traffic we have been studying so far, and is thus  $O(n^22^{2l})$ .

b) I/O generated during the execution of step 2, that is the elimination step. In this step, the  $A'$ -file is read *twice*, once to carry out the ordering, and a second time to carry out the actual numerical factorization of the block. It can be shown that the total I/O generated during the execution of this step is  $O(n^2l)$ , in addition to that required to write out the Cholesky factor, which is  $O(n^2 \log n)$ .

From the above discussion, we may conclude that  $V_{\text{tot}}$  as given by (4.8) is in fact a reasonable indication of the total I/O overhead of our algorithm (aside from the I/O associated with storing the Cholesky factor  $R$ ). Also note that even though the expression for  $V_{\text{tot}}$  has a significant negative lower order term, the contributions from this and other low order terms diminish as  $l$  increases. Hence, it seems reasonable to consider only the asymptotic behavior when we “theoretically” compare this algorithm against others. In a practical setting, however, such a comparison should be based on the results of an actual computer implementation.

We now show that the total I/O traffic generated is  $O(n^2 \log n)$ , provided that the main memory available is  $O(n^2)$ . The storage required to factor a separator block

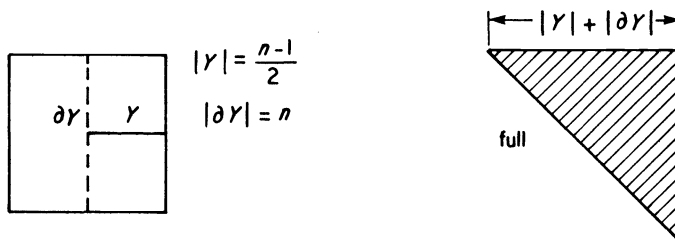


FIG. 4.5. Elimination of a separator block at level one.

of size  $(n - 1)/2$  at level 1 is given by  $\gamma((3n - 1)/2) \approx \frac{9}{8}n^2$ , as shown in Fig. 4.5. Note that no other separator block requires more storage to process.

An estimate of  $l$  (which measures the fineness of the partitioning), under the  $O(n^2)$  main memory assumption is given below.

LEMMA 4.7. *A last-level block of a  $2l$ -level incomplete nested dissection partitioning of the  $n$  by  $n$  grid may be eliminated in  $\frac{9}{8}n^2$  main memory, provided that  $l \geq \frac{1}{2} \log \log n + O(1)$ .*

*Proof.* Using the approach of George and Liu [1981a], it can be shown that the number of nonzeros in the Cholesky factor of the matrix associated with a bordered grid of size  $\alpha$  by  $\alpha$  is given by  $\frac{31}{4}\alpha^2 \log \alpha - 4\alpha^2 + O(\alpha \log \alpha)$ , where the grid vertices are ordered using nested dissection. Hence, to factor a last-level block in main memory, the following inequality must be satisfied.

$$(4.9) \quad \frac{31}{4}\alpha^2 \log \alpha - 4\alpha^2 + O(\alpha \log \alpha) \leq \frac{9}{8}n^2.$$

We are interested in finding the largest  $\alpha$  (or equivalently, the smallest  $l$ ), which satisfies (4.9). Using  $\alpha \approx n/2^l$  and ignoring lower order terms in (4.9), we obtain the following inequality:

$$2^{2l} \geq \frac{62}{9} \log n; \quad \text{hence, } l \geq \frac{1}{2} \log \log n + O(1). \quad \square$$

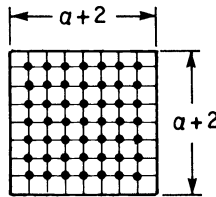


FIG. 4.6. A bordered grid of size  $\alpha$  by  $\alpha$ .

It follows immediately that the total input traffic is roughly given by  $Cn^2 \log n$ , where  $C = \frac{310}{3}$ , which is of the same order as that required to write out the Cholesky factor, namely  $\frac{31}{4}n^2 \log n + O(n^2)$ , but with a much larger constant.

**5. An “optimal” method for file management.**

**5.1. Introductory remarks.** In this section we consider two simple, but effective enhancements for file management. Recall that Algorithm 3.1 was *not* specifically designed for a nested dissection partitioning, since it can handle any partitioning. Thus, a substantial reduction in the I/O traffic can be achieved by exploiting certain features of a nested dissection partitioning.

Consider the elimination of the blocks in a level-by-level fashion. When a block at level  $i$  is eliminated, the resulting Schur complement submatrix plays no role in subsequent eliminations in the  $i$ th level. Thus, writing this Schur complement into the input file, as done by Algorithm 3.1, results in the unnecessary transfer of such a submatrix between main memory and auxiliary storage during subsequent eliminations in the  $i$ th level. Hence, using a separate file, which we call the *S*-file, to store the Schur complements may considerably reduce the I/O traffic. At the end of eliminations in the current level the input file is empty and the *S*-file will contain all the data required for the elimination of blocks in the next level.

Our second enhancement is motivated by the following discussion. Recall that the largest fraction of the I/O traffic of Algorithm 3.1, apart from writing the Cholesky factor  $R$ , was generated during the block *identification* step. This involves reading the

whole input file to identify one block before its elimination. This raises the following question: can we identify the elimination block without reading the whole input file? To answer this question, we look at the reasons behind reading the whole input file in the first place. Apparently, there are two reasons:

1. The rows of the input file were assumed to be stored in *arbitrary* order. This assumption was necessary because the given matrix is usually *re-ordered* before elimination (to reduce fill-in and arithmetic operations), and the structure of the re-ordered matrix hardly resembles that of the original one. Thus, the matrix elements were used in an order different from their order of storage.

2. The order of block elimination in each level was also assumed arbitrary, since all orderings are known to be equivalent in the sense that they produce the same fill-in and operations count.

In the following section, we address these problems and show that using a particular order of block elimination in each level, coupled with a corresponding re-organization of the data in the input file allows us to achieve our goal of eliminating the reading of the whole input file before the elimination of each block.

**5.2. The new algorithm and its I/O traffic.** Consider the four-level incomplete nested dissection partitioning of the  $n$  by  $n$  grid, and the associated separator tree of

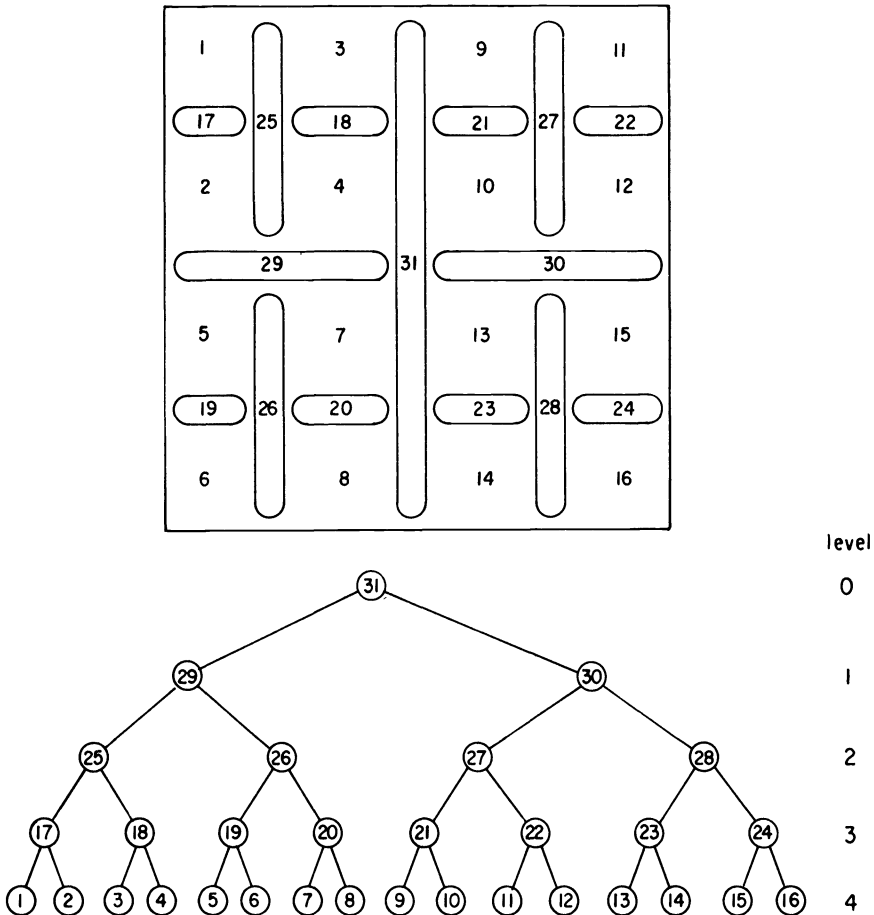


FIG. 5.1. A 4-level incomplete nested dissection partitioning of the  $n$  by  $n$  grid and the associated separator tree.



Fig. 5.1. Note that in a nested dissection partitioning, a node of the tree is labeled before its father. The labeling shown in Fig. 5.1 is obtained using breadth-first search starting at the root of the tree, and then reversing the node labels. This produces a so-called breadth-first nested dissection partitioning (Sherman [1975]). This is a special case of the level-by-level labeling of the tree.

We now show how to make use of this particular labeling scheme in the elimination process. Let  $\Omega = (Y_1, Y_2, \dots, Y_{31})$  be the ordered partition corresponding to the above mentioned tree. Suppose the rows of the input file are sorted so that rows belonging to  $Y_i \cup \partial Y_i$  appear before those belonging to  $Y_{i+1} \cup \partial Y_{i+1}$ ,  $i = 1, 2, \dots, 15$ . We will refer to the set of rows belonging to one set  $Y_i \cup \partial Y_i$  as "segment  $i$ ." Fig. 5.2 shows the input file with the reading head positioned at the beginning of segment  $i + 1$ .

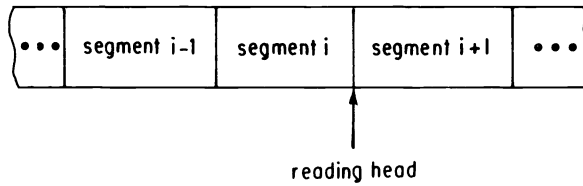


FIG. 5.2. The input file with the reading head positioned after the elimination of the  $i$ th last-level block.

The elimination of the  $i$ th last-level block leaves the reading head of the input file positioned at the beginning of the  $(i + 1)$ st segment. Hence, when we are about to eliminate the  $(i + 1)$ st block, we find that the reading head of the input file is in the correct position, and we may proceed to read the  $(i + 1)$ st segment without the need for doing any search.<sup>3</sup> The same argument applies to all last-level blocks. For this conclusion to be valid, we must use a separate file to store the resulting Schur complements. Fig. 5.3 shows this separate file ( $S$ -file), with its reading head in the initial position.

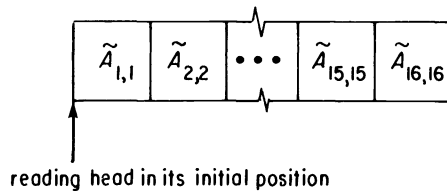


FIG. 5.3. The  $S$ -file before starting elimination in the third level.

When all last-level blocks have been eliminated, the  $S$ -file becomes the input file for the third level. The elimination of blocks in this level may proceed as before, with one basic difference, namely, *two consecutive* segments rather than one must read to effect the elimination of each block. The eliminations in all other levels are similar.

The preceding discussion suggests that Algorithm 3.1 should be modified to accept a breadth-first nested dissection partitioning. To simplify our presentation, we assume that the input file has been already sorted and that a breadth-first nested dissection partitioning is provided. These issues and the extension to irregular grids will be considered in some detail in the following section. Four sequential files are required,

<sup>3</sup> That is, no additional search apart from the initial sorting of the input file, which we have ignored so far.

and they are described below.

1. *R*-file: accepts the rows of *R* as they are computed.
2. *A'*-file: at the *k*th elimination step this file contains the rows of  $A'_k$ , which is the current elimination block.
3. *S*-file: this is a separate file to store the Schur complements. At the beginning of elimination in each level, this file is empty. It receives the rows of the Schur submatrices resulting from the current level. The roles played by this file and the *A*-file are interchanged after processing each level.
4. *A*-file: at the beginning of the *k*th step of the computation, this file contains the rows corresponding to the remaining blocks in the current level. These rows are already distributed into sorted segments according to the order of block elimination. To effect the elimination of a block, one (or two) segments are copied from this file into the *A'*-file. This file becomes empty when a whole level is eliminated.

ALGORITHM 5.1.

1.  $kstop = 0$ .
2. **For** each level  $i$ ,  $i = 2l, 2l-1, \dots, 0$  **do**
  - 2.1 Rewind the *A*-file, and the *S*-file.  
Set  $kstart = kstop + 1$ , and  $kstop = kstart + 2^i - 1$ .
  - 2.2 **For**  $k = kstart, kstart + 1, \dots, kstop$  **do**
    - 2.2.1 Rewind the *A'*-file.
    - 2.2.2 **If**  $i = 2l$  **then** number-of-segments = 1  
**else** number-of-segments = 2.
    - 2.2.3 Copy number-of-segments consecutive segments from the *A*-file into the *A'*-file.
    - 2.2.4 Using the *A'*-file as an input file, eliminate the variables in  $Y_k$ . Then write the rows of  $R_{kk}$  on the *R*-file, and those of  $\hat{A}_{kk}$  on the *S*-file.
  - 2.3 Reverse the names of the *A*-file and the *S*-file.

The main difference between this algorithm and Algorithm 3.1 is that the block identification step is now trivial. The data in the input file are organized so that when a block is about to be eliminated, the read/write head is in the correct position to allow us to copy the segment(s) corresponding to the elimination block directly from the *A*-file to the *A'*-file.

The estimation of the I/O traffic generated when Algorithm 5.1 is applied to our model  $n$  by  $n$  grid problem is straightforward. Once again, we only consider the input traffic generated during the block identification step (step 2.2.3). The I/O traffic generated during the other steps may be found in a similar fashion (indeed, it is given by the same expressions shown here).

The input traffic generated in each level is equal to the size of the input file *before* eliminating any block in that level. Thus, using Lemma 4.2, and summing over all levels, we obtain the following result.

**THEOREM 5.1.** *The total input traffic generated during the factorization of the matrix associated with an  $n$  by  $n$  grid using Algorithm 5.1, is given by:*

$$(5.1) \quad V_{tot} = 17n^2l - \frac{51}{2}n^2 + 35n^22^{-l} - 4n^22^{-2l} + O(n2^l + 2^{2l}).$$

This estimate compares quite favorably with that of § 4. The leading term of  $V_{tot}$  is reduced from  $15n^22^{2l}$  to  $17n^2l$ . Note that the I/O traffic is “*optimal*” in the sense

that each block of the input file is read as few times as possible (basically, each block is initially copied from the  $A$ -file to the  $A'$ -file, and then the  $A'$ -file is read twice, once to provide the matrix structure, and once to provide the actual numerical values). Furthermore, the total I/O traffic generated is dominated by that associated with storing the Cholesky factor alone.

An alternative approach which is beneficial in further reducing the I/O traffic is motivated by the following discussion. Our bounds were derived under the assumption that we only eliminate one block at a time in a level-by-level fashion. This implies that after the elimination of a block we write out the rows of the corresponding Schur complement into the  $S$ -file. We can save these intermediate results in main memory provided that we use a depth-first nested dissection, and only output them to auxiliary storage when their volume exceeds the memory area allocated for that purpose. In this way we can effectively eliminate a whole subtree without writing any data other than the rows of the corresponding Cholesky factor and the Schur complement corresponding to the root of the subtree. (We assume that the last-level blocks have been eliminated in the usual manner, otherwise our memory requirement may have to be increased.) The price paid for this reduction in the traffic is an additional overhead resulting from a complex storage management and the shuffling of intermediate data in main memory.

### 5.3. Block re-ordering and input file pre-processing.

**5.3.1. Block re-ordering.** In this section we describe an algorithm for re-ordering the blocks resulting from a nested dissection partitioning of an irregular grid, so that Algorithm 5.1 may be generalized to handle such a problem as well. Throughout this section we assume that the reader is familiar with some basic graph theory notation, e.g. as described in George and Liu [1981a]. The basic idea here is to shuffle the blocks so that they correspond to a breadth-first traversal of the underlying separator tree. To simplify our discussion, we assume that the matrix  $A$  is *irreducible*; thus, the associated graph  $G=(X, E)$  is *connected*. The block reordering algorithm may be considered to consist of the following three basic steps:

- 1) Build the *quotient graph*  $G/\Omega=(\Omega, E_\Omega)$  corresponding to the given partitioning  $\Omega=(Y_1, Y_2, \dots, Y_p)$ .
- 2) Identify (construct) the separator tree associated with  $\Omega$ .
- 3) Find a permutation  $i_1, i_2, \dots, i_p$  of the integers  $1, 2, \dots, p$ , where  $p=|\Omega|$ , so that  $\Psi=(Y_{i_1}, Y_{i_2}, \dots, Y_{i_p})$  corresponds to a breadth-first traversal of the tree.

In what follows, we provide three algorithms to implement the above steps. An example illustrating the application of the algorithms is also provided. Our first step is to form the adjacency structure of the quotient graph  $G/\Omega=(\Omega, E_\Omega)$  from that of  $G=(X, E)$  as follows.

ALGORITHM 5.2.

1. **For** each node  $Y_i, i=1, 2, \dots, p$  **do**
  - 1.1 Set  $Adj_{G/\Omega}(Y_i)=\emptyset$ .
  - 1.2 **For** each vertex  $y \in Y_i$  **do**
    - 1.2.1 **For** each vertex  $z \in Adj_G(y)$  **do**
      - 1.2.1.1 Let  $Y_k$  be the node containing  $z$ .
      - 1.2.1.2 **If**  $Y_k \in Adj_{G/\Omega}(Y_i)$ 
**then**  $Adj_{G/\Omega}(Y_i)=Adj_{G/\Omega}(Y_i) \cup Y_k$ .

In Algorithm 5.3, we analyze the structure of the quotient graph  $G/\Omega$  in order to identify the underlying tree structure. Note that in this algorithm and in Algorithm 5.4

the only graphs which are used are quotient graphs, and for convenience we use the index  $i$  to refer to the node  $Y_i$ . The basic fact exploited by Algorithm 5.3 is that the separator subtree associated with a connected component of  $G/\Omega$  is rooted at the node whose label is largest in this component. Moreover, when such a node is deleted, the corresponding component splits into two or more connected components. The separator tree is represented using a *FATHER* vector of length  $p$ , where  $FATHER(i) = k$  means that node  $k$  is the father of node  $i$ .

ALGORITHM 5.3.

1.  $Z = \{1, 2, \dots, p\}; Z' = \emptyset$ .
2. **While**  $Z \neq \emptyset$  **do**
  - 2.1 For each connected component  $C_i$  in the quotient graph  $G/\Omega(Z)$  **do**
    - 2.1.1 Let  $r_i$  be the node in  $C_i$  of largest label.
    - 2.1.2 **For** each node  $x \in C_i - \{r_i\}$  **do**  
 Set  $FATHER(x) = r_i$ , and  $Z' = Z' \cup \{x\}$ .
  - 2.2  $Z = Z'$ ;  $Z' = \emptyset$ .

In the above algorithm the connected components of  $G/\Omega(Z)$  in step 2.1 may be found using breadth-first-search. In step 2.1.2, the node  $r_i$  is considered the “father” of all nodes in the connected component  $C_i$ . When the algorithm terminates, the *FATHER* vector contains the right information.

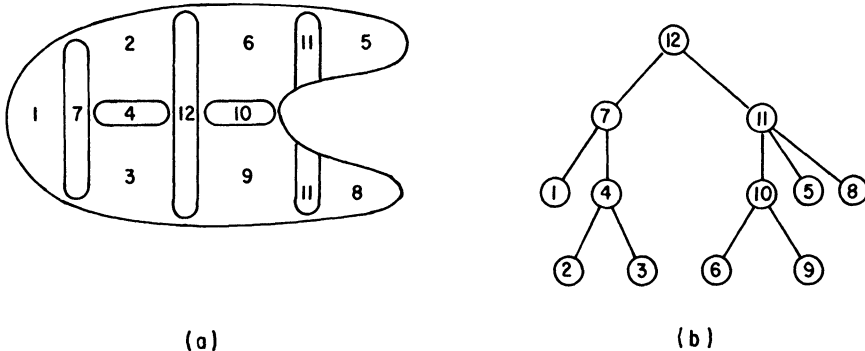
Algorithm 5.4 applies breadth-first-search to the separator tree in order to find the required permutation of the members of  $\Omega = (Y_1, Y_2, \dots, Y_p)$ .

ALGORITHM 5.4.

1.  $\pi = p$ ; mark  $p$  “new”.
2. **While** there exists a node  $x \in \pi$  marked “new” **do**
  - 2.1 Mark  $x$  “old”.
  - 2.2 Insert the sons of  $x$  at the end of  $\pi$ , and mark them “new”.
3. Reverse the order of the elements in  $\pi$ .

The order in which the sons of a given node  $x$  are found and inserted in the ordered list  $\pi$  is immaterial as long as they are added at the end. The order of the elements in  $\pi$  is eventually reversed, since nodes found first are to be eliminated last.

The three phase block reordering algorithm can be easily implemented to run in  $O(|E| + N + h|E_\Omega|)$  time, where  $h$  is the height of the separator tree. This bound is typically  $O(|E|)$ .



A skeleton of a graph  $G$ , showing the partition members  $Y_i, i = 1, 2, \dots, 12$ .

Underlying separator tree.

FIG. 5.4. A dissected graph and its corresponding separator tree.

The following example illustrates the application of the block reordering algorithm. Consider the partitioned graph of Fig. 5.4, with  $\Omega = (Y_1, Y_2, \dots, Y_{12})$ . Also shown in the same figure is the separator tree which we want to identify.

The corresponding quotient graph  $G/\Omega$  is shown in Fig. 5.5, along with a few steps of Algorithm 5.3.

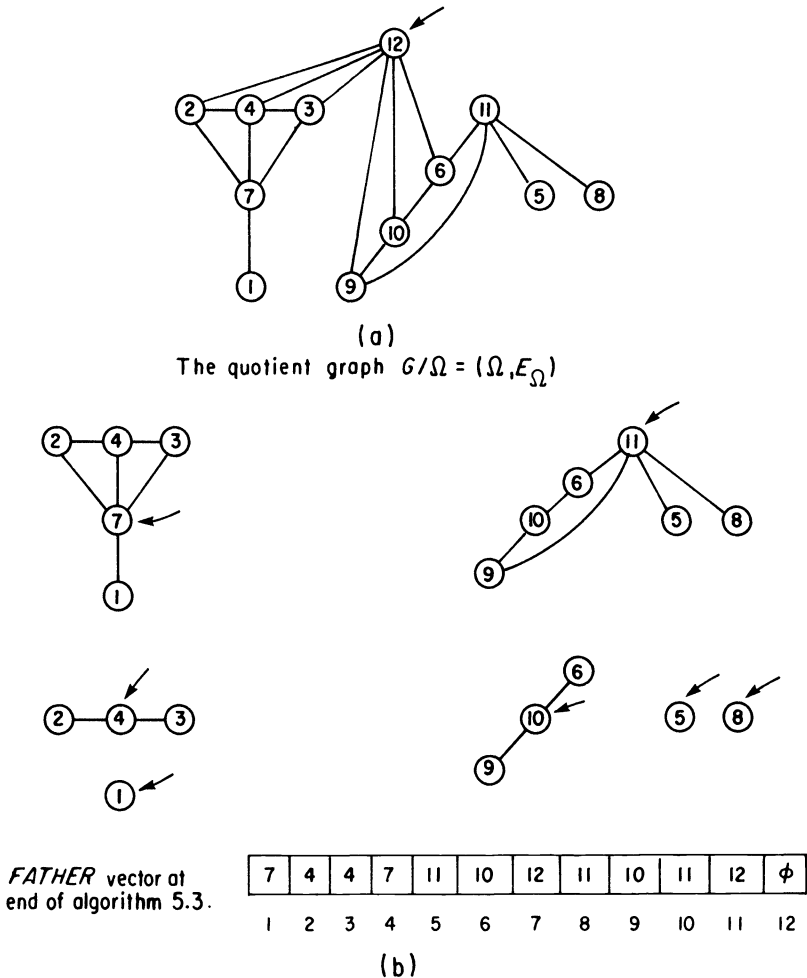


FIG. 5.5. Construction of the father vector from the quotient graph.

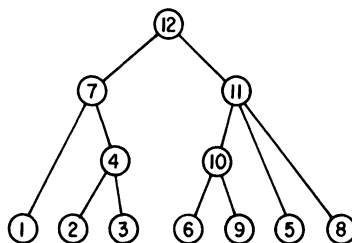


FIG. 5.6. Separator tree with all leaves moved down to the last-level.

Note that some of the leaves do not lie in the last-level of the separator tree. By moving these leaves down to the last-level of the tree as shown in Fig. 5.6, we may simplify the input file preprocessing step which follows.

**5.3.2. Input file preprocessing.** In this section, we consider the problem of reorganizing the rows of the matrix  $A$  on the input file. Our goal here is to have the rows which belong to a particular last-level block appear consecutively in one “segment” and the order of these segments in the input file to be *compatible* with the order of block elimination, as described in the previous section.

The input file preprocessing consists of two steps. In the first step, we split rows which span more than one last-level block among their respective blocks. In the second step, the rows of the input file are sorted into the required order.

Recall that the rows of the upper triangular part of  $A$  are stored on a sequential access file, and without loss of generality, we assume that the following format is used:

nsubs; row index; (column subscripts); (corresponding numerical values).

Here, nsubs is the number of nonzeros in the row. For example:

$$5; 7; (8, 7, 17, 18, 16); (-1.0, 4.0, -1.2, -1.5, -1.1),$$

is the way to store the 7th row which has five nonzeros, namely:

$$a_{7,8} = -1.0, \quad a_{7,7} = 4.0, \quad a_{7,17} = -1.2, \quad a_{7,18} = -1.5 \quad \text{and} \quad a_{7,16} = -1.1.^4$$

In the first step, we solve the problem illustrated in the following example. Consider Fig. 5.7, which is a part of an  $n$  by  $n$  grid partitioned into  $\Omega = (Y_1, Y_2, \dots, Y_7)$ , with  $Y_1, Y_2, Y_3,$  and  $Y_4$  being the last-level blocks.

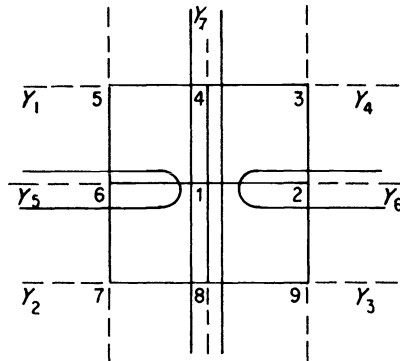


FIG. 5.7. A portion of a partitioned grid.

Our purpose here is to associate each row (or subrow thereof) of the input file with one and only one set  $Y_i \cup \partial Y_i, i = 1, 2, 3,$  or  $4$ . Clearly, a row whose row index belongs to a last-level block poses no problem since it already belongs to one set. On the other hand, a row whose row index belongs to a separator block may have to be split among those last-level blocks that share the same separator as a part of their common boundary, as exemplified by the following row:

$$9; 1; (1, 2, 3, 4, 5, 6, 7, 8, 9); (a_{11}, a_{12}, a_{13}, a_{14}, a_{15}, a_{16}, a_{17}, a_{18}, a_{19}).$$

<sup>4</sup> Incidentally, the nonzero elements belonging to one row need not be stored as one entity (record). Hence, it is quite legitimate to have two records which have the same row number.

This row should be split into four subrows, each of which belongs to one last-level block. A possible splitting is given by:

$$\begin{aligned} 4; 1; (1, 4, 5, 6); (a_{11}, a_{14}, a_{15}, a_{16}) &\in Y_1 \cup \partial Y_1 \\ 2; 1; (7, 8); (a_{17}, a_{18}) &\in Y_2 \cup \partial Y_2 \\ 2; 1; (2, 9); (a_{12}, a_{19}) &\in Y_3 \cup \partial Y_3 \\ 1; 1; (3); (a_{13}) &\in Y_4 \cup \partial Y_4. \end{aligned}$$

Ties might arise during this splitting; e.g.,  $a_{12}$  may be considered as belonging to  $Y_3 \cup \partial Y_3$  or  $Y_4 \cup \partial Y_4$ . Such ties are broken arbitrarily. In Algorithm 5.5, the rows of the input file are read one row at a time, and are split (if necessary) among the last-level blocks. On output, each row is *tagged* by an additional field (*key*) to identify its block number. This key is used by the sorting algorithm which will follow.

ALGORITHM 5.5.

1. Rewind the input and output files.
2. Read the rows of the input file one by one and for each do the following:
  - 2.1 If the row index belongs to a last-level block,  $Y_k$  say, then write the row into the output file, with a key identifying it as belonging to block number  $k$ .
  - 2.2 If the row index does not belong to a last-level block, then split it into subrows, each of which have elements belonging to one set  $Y_k \cup \partial Y_k$ , for some  $k$ , and write each subrow into the output file, after tagging each by its appropriate key.
3. Reverse the names of the input and output files.

The cost of the above algorithm in I/O traffic is essentially reading and writing the input file once. We now turn our attention to the second step in preprocessing the input file. In this step, the rows of the input file are sorted so that the rows which belong to one last-level block appear in sequence as one segment, and that the order of these segments in the input file is the same as the order of block elimination. Our sorting algorithm is a minor modification of *radix sorting* (also known as "bucket" sorting, Aho et al. [1983]). For our purpose here, the buckets are replaced by auxiliary storage files.

We assume that  $k$  scratch files  $f_0, f_1, \dots, f_{k-1}$ ,  $k \geq 2$  are at our disposal. Let  $p'$  be the number of last-level blocks, and for convenience, we assume that  $p' = k^t$ , for some integer  $t$ . Recall that a key is associated with each row to identify its block number. We use block-number-1 as our key, in which case all the keys lie in the range  $0 \dots k^t - 1$ . The basic idea of radix sorting is to consider the keys associated with the rows as  $t$ -digit integers in a base- $k$  representation:  $d_t d_{t-1} \dots d_1$ , where  $0 \leq d_i \leq k-1$ ,  $i = 1, 2, \dots, t$ .

The input file will be sorted in  $t$  passes. In the first pass, the rows of the input file are read one row at a time, and are distributed among the  $k$  scratch files. A row whose associated key has its right most digit  $d_1$  equal to  $j$  is written into  $f_j$ . When the input file is exhausted, the scratch files  $f_1, f_2, \dots, f_{k-1}$  are copied<sup>5</sup> (*concatenated*) to the end of  $f_0$ . The roles of  $f_0$  and the input file are reversed at the end of each pass. All subsequent passes are similar, except that on the  $i$ th pass the rows are distributed

<sup>5</sup> For our sorting to be correct we must append (write) the rows to the end of their respective destination files.

among the destination files based on the value of  $d_i$ . Algorithm 5.6 formalizes this sorting process.

ALGORITHM 5.6.

1. **For**  $i = 1, 2, \dots, t$  **do**
  - 1.1 Read the input file row by row, and for each row **do**
    - 1.1.1 Let  $d_i$  be the  $i$ th digit of the *key* associated with the current row.
    - 1.1.2 Write the current row into  $f_j, j = d_i$ .
  - 1.2 Concatenate  $f_j, j = 1, 2, \dots, k-1$  to the end of  $f_0$ .
  - 1.3 Reverse the names of the input file and  $f_0$ .

Note that the algorithm works correctly, even if  $p'$  is not a power of  $k$ , in which case some of the scratch files would be empty. It can be easily shown that the I/O traffic for preprocessing the input file is  $O(n^2l)$ , which is the same order as the I/O traffic generated during the actual factorization.

We will briefly discuss the modifications required in Algorithm 5.1, so that it can handle irregular grids. Note that, in the case of irregular grids, the resulting separator tree is not necessarily a *complete binary tree*. In Fig. 5.8, the separator tree of Fig. 5.6 is redrawn with the node labels changed to reflect their order of elimination.

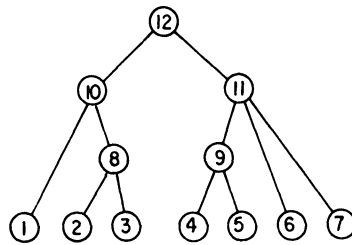


FIG. 5.8. Separator tree of Fig. 5.6, with node labels changed.

The following remarks are useful in arriving at the required modifications:

- 1) The elimination of all last-level blocks is unchanged, since we still have to read one segment from the *A*-file to effect the elimination of each block.
- 2) The fact that the tree is no longer a complete binary tree leads to the following situations:
  - i) When we eliminate blocks at levels other than the last-level, we may encounter certain segments of the *A*-file which *do not* contribute to the elimination in the level under consideration. For instance, the segments corresponding to the Schur complements of blocks (nodes) 1, 6, and 7 in Fig. 5.8 are not used in the second-level. Hence, these segments should be copied directly from the *A*-file to the *S*-file as soon as they are encountered, preserving their relative order.
  - ii) Some nodes may have *more* than two sons. Thus, when such a node is to be eliminated we have to read an equal number of segments from the *A*-file. For example, node 11 of Fig. 5.8 has three sons; so we have to read three segments from the *A*-file to effect the elimination of this node.

The incorporation of the above mentioned changes into Algorithm 5.1 is straightforward.

**6. Numerical experiments.** In this section we present the results of some numerical experiments performed on  $n^2$  by  $n^2$  linear systems resulting from the application of



the standard 9-point finite difference operator to an  $n$  by  $n$  grid. The linear systems were solved using an implementation of Algorithm 3.1 and Algorithm 5.1. As a basis of comparison, the given systems were also solved in main memory without partitioning them into smaller subsystems. Many of the subroutines used were taken from SPAR-SPAK, the University of Waterloo Sparse Linear Equations Package (George and Liu [1979], George, Liu, and Ng [1980]). All the experiments were carried out on an IBM 4341 computer, using the Fortran VS optimizing compiler. The times reported are in seconds, and the storage in single precision words.

The labels used in the tables have the following meaning.

- $S$ —the maximum storage used during the factorization (including overhead storage for pointers, permutations, . . . , etc.),
- $\tau_F$ —the factorization time, including ordering and storage allocation times,
- $\tau_T$ —the total execution time, including I/O processing time, in addition to  $\tau_F$  and related overhead times (e.g. times to record the structure of each block, and to store the actual numerical values in the data structures),
- $\tau_{id}$ —I/O processing time for the block identification step,
- $\tau_{pre}$ —input file pre-processing time (for Algorithm 5.1),
- $\tau_S$ —time for the forward and backward substitution steps, including reading the  $R$ -file,
- $p$ —the number of blocks in the partitioning.

Table 6.1 summarizes the results obtained when the problem is solved in main memory without partitioning. The grid points were ordered using nested dissection. The times required to read the input matrix and to write the Cholesky factor into auxiliary storage are included in the total execution time.

TABLE 6.1  
Problems are solved without partitioning.\*

$N = n^2$	$S$	$\tau_F$	$\tau_T$	$\tau_S$
400	11,613	.96	3.16	2.68
625	19,703	1.82	5.48	4.53
900	29,939	3.07	8.46	6.85
1600	58,453	7.15	17.33	13.14
2500	98,121	13.59	29.79	21.82
3600	147,771	23.12	48.14	32.93
5625	243,000	45.41	86.00	60.18
8100	368,000	77.35	138.00	92.65
10000	465,000	102.73	179.00	114.00

\* The entries in the last three rows of the table are estimates, since we do not have enough storage to solve these large problems without partitioning them.

The entries in Table 6.1 are a clear indication why most people would like to avoid using auxiliary storage if at all possible. Note that the amount of I/O activity is minimal, and we do not expect any of our methods to do any better. However, the picture is not so gloomy, as we will see shortly. The solution time  $\tau_S$  is very large, relative to the factorization time ( $\tau_F$ ), since almost all of it is spent in I/O.

Our next experiment involved solving one "large" problem, using different levels of partitioning. We used a system with  $N = n^2 = 2500$ , and gradually increased the number of partition members (thus, we were able to use less storage).

The information in Table 6.2 shows that our scheme is effective in reducing the storage requirements. The price paid for this is a (large) increase in the total execution

TABLE 6.2  
Solving for  $N = 2500$ , using different levels of partitioning.

$p$	Algorithm 3.1				Algorithm 5.1				
	$S$	$\tau_{id}$	$\tau_T$	$\tau_{id}/p$	$S$	$\tau_{id}$	$\tau_{pre}$	$\tau_T$	$\tau_{id}/\log p$
1	98,121	.	29.79	.	98,121	.	.	29.79	.
7	32,148	20.03	56.67	2.86	31,907	10.51	17.57	50.39	3.74
15	29,014	51.23	98.87	3.42	27,330	15.92	19.64	67.10	4.08
23	20,536	82.03	138.31	3.57	20,001	19.29	27.59	78.63	4.27
29	14,675	105.14	166.52	3.63	14,748	23.39	29.48	85.73	4.81
35	13,121	127.43	192.42	3.64	12,938	24.57	30.73	93.79	4.79
47	12,420	174.41	246.60	3.71	12,371	27.32	33.65	106.01	4.92

time. Note, however that by partitioning the matrix beyond a certain level, the increase in execution time outweighs the marginal reduction in storage, as indicated by the last three rows in the table. We also note that the block identification time ( $\tau_{id}$ ) for Algorithm 5.1 is much less than the corresponding values for Algorithm 3.1. Our analysis shows that the ratio  $\tau_{id}/p$  for Algorithm 3.1 converges to a constant, and indeed this is supported by our experiments. The convergence is very slow due to the existence of large subdominant terms. Similar conclusions may be drawn for Algorithm 5.1, where  $p$  is replaced by  $\log p$ .

In Table 6.3, we illustrate the performance of our algorithms by solving a sequence of problems of increasing size.

TABLE 6.3  
Solving a sequence of partitioned linear systems.

$N = n^2$	$p$	Algorithm 3.1			Algorithm 5.1			
		$S$	$\tau_{id}$	$\tau_T$	$S$	$\tau_{id}$	$\tau_{pre}$	$\tau_T$
400	19	3,089	11.92	22.26	3,126	3.32	5.02	12.48
625	23	4,712	22.29	38.07	4,747	6.03	7.84	20.87
900	25	6,644	33.73	55.79	6,658	8.13	10.96	29.10
1600	27	10,736	63.23	101.31	10,854	13.08	19.09	50.76
2500	29	14,675	105.14	166.52	14,748	23.39	29.48	85.73
3600	31	19,100	157.45	248.86	19,045	31.77	42.30	127.79
5625	33	30,935	284.64	448.55	30,736	49.77	70.75	217.18
8100	35	44,051	433.06	685.68	43,739	71.96	101.78	332.78
10000	37	55,299	560.80	879.59	54,824	86.30	121.36	414.97

Few comments need be made about the information in Table 6.3. The storage requirements of both algorithms are quite modest; in particular, a few vectors of length  $N = n^2$  are used. Algorithm 5.1 seems quite successful in reducing the I/O overhead as attested to by the substantial reduction of  $\tau_{id}$  and  $\tau_T$  over the corresponding entries for Algorithm 3.1. Unfortunately, the preprocessing time  $\tau_{pre}$  is relatively large, hinting that our external sorting procedure is not very efficient. Finally, we note that even though the numerical experiments reported here are for the model  $n$  by  $n$  grid, our subroutines are not specifically designed for that special problem, and indeed results similar to those presented here were obtained for irregular finite element problems.

**7. Concluding remarks and further extensions.** In this paper we have analyzed and compared two schemes for the solution of very large sparse linear systems of equations using auxiliary storage. A detailed analysis of the I/O traffic generated when our first method is applied to a model  $n$  by  $n$  grid problem, where the grid is partitioned using incomplete nested dissection, shows that a substantial reduction in memory requirements can be achieved without compromising the low fill-in and operation counts of the standard nested dissection ordering. In particular, we have shown that using a few vectors of length  $N = n^2$  allows us to achieve an I/O traffic of  $O(n^2 \log n)$ , which is dominated by the  $O(n^3)$  arithmetic computation. We also considered an important enhancement to our basic strategy, which results in reducing the traffic to  $O(n^2 \log \log n)$ , apart from storing the Cholesky factor itself. This further reduction in the I/O traffic is achieved by modifying our algorithm to exploit a certain tree structure associated with nested dissection.

The work presented here can be extended in several directions (Rashwan [1983]) to include:

i) The methods described in this paper assume that  $O(n^2)$  main storage is available. This assumption might limit the size of the largest problem that can be handled. We may relax this assumption by trading a reduction in main memory for some increase in the I/O traffic, and at the same time maintaining the dominance of the  $O(n^3)$  arithmetic computation. This can be achieved as long as the main storage available is not less than  $O(n)$ .

ii) Our analysis can be easily extended to sparse linear systems associated with regular three-dimensional grids. We have also considered linear systems whose graphs are planar or almost planar, which are partitioned and ordered by the generalized nested dissection algorithm of Lipton et al. [1979].

iii) Throughout our discussion, we have assumed that the partitioning of the linear system is given. This might be a reasonable assumption if we were only concerned with simple and regular meshes. In certain engineering and structural design applications, a problem partitioning might arise in a natural manner. A possible solution (the one we used in our implementation) is to perform the partitioning in main memory, since we only require a few integer working vectors of length  $N$ , in addition to the storage of the sparsity structure of the given system. This approach may be applicable if  $N$  is not too large, in which case we have the additional benefit that more than one integer data item can often be packed in one floating point word. This is also possible if extended precision is used during the actual numerical processing. It would certainly be very useful if we can generate the required partitioning using a limited amount of main memory without an unduly high level of I/O traffic. Note that in applications where several systems have the same matrix structure, the cost of the partitioning step is "shared" among all of the systems to be solved, and thus it may be affordable even if its I/O traffic turns out to be relatively large.

#### REFERENCES

- [1] A. V. AHO, J. E. HOPCROFT AND J. D. ULLMAN [1983], *Data Structures and Algorithms*, Addison-Wesley, Reading, MA.
- [2] K. O. GEDDES, G. H. GONNET AND B. W. CHAR [1982], *MAPLE user's manual*, second edition, Research report CS-82-40, Dept. Computer Science, Univ. Waterloo, Waterloo, Ontario.
- [3] J. A. GEORGE [1977], *Numerical experiments using dissection methods to solve  $n$  by  $n$  grid problems*, SIAM J. Numer. Anal., 14, pp. 161-179.
- [4] J. A. GEORGE, W. G. POOLE JR. AND R. G. VOIGT [1978], *Incomplete nested dissection for solving  $n$  by  $n$  grid problems*, SIAM J. Numer. Anal., 15, pp. 662-673.

- [5] J. A. GEORGE AND J. W. H. LIU [1979], *The design of a user interface for a sparse matrix package*, ACM Trans. Math. Software, 5, pp. 134-162.
- [6] J. A. GEORGE, J. W. H. LIU AND E. G. Y. NG [1980], *User's guide for SPARSPAK: Waterloo sparse linear equations package*, Research report CS-78-30 (revised), Dept. Computer Science, Univ. Waterloo, Waterloo, Ontario, Canada.
- [7] J. A. GEORGE AND J. W. H. LIU [1981a], *Computer Solution of Large Sparse Positive Definite Systems*, Prentice-Hall, Englewood Cliffs, NJ.
- [8] J. A. GEORGE, M. T. HEATH AND R. J. PLEMMONS [1981b], *Solution of large-scale sparse least squares problems using auxiliary storage*, this Journal, 2, pp. 416-429.
- [9] J. A. GEORGE AND H. RASHWAN [1982], *Input/output traffic analysis of an auxiliary storage scheme for solving finite element systems*, Proc. 5th International Symposium on Computing Methods for Applied Science and Engineering, Versailles, France, R. Glowinski and J. L. Lions, eds., North-Holland, Amsterdam, pp. 3-25.
- [10] J. R. GILBERT [1980], *Graph separator theorems and sparse Gaussian elimination*, Report STAN-CS-80-833, Dept. Computer Science, Stanford Univ., Stanford, CA.
- [11] A. J. HOFFMAN, M. S. MARTIN AND D. J. ROSE [1973], *Complexity bounds for regular finite difference and finite element grids*, SIAM J. Numer. Anal., 10, pp. 364-369.
- [12] B. M. IRONS [1970], *A frontal solution program for finite element analysis*, Int. J. Numer. Meth. in Engng., 2, pp. 5-32.
- [13] R. J. LIPTON, D. J. ROSE AND R. E. TARJAN [1979], *Generalized nested dissection*, SIAM J. Numer. Anal., 16, pp. 346-358.
- [14] H. RASHWAN [1983], *Auxiliary storage methods for sparse positive definite linear systems*, Ph.D. thesis, Dept. Computer Science, Univ. Waterloo, Waterloo, Ontario.
- [15] J. K. REID [1974], *Direct methods for sparse matrices*, in Software for Numerical Mathematics, D. J. Evans, ed., Academic Press, New York, pp. 29-47.
- [16] D. J. ROSE [1972], *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, pp. 183-217.
- [17] D. J. ROSE AND G. F. WHITTEN [1976], *A recursive analysis of dissection strategies*, in Sparse Matrix Computations, J. R. Bunch and D. J. Rose, eds., Academic Press, New York, pp. 59-84.
- [18] A. H. SHERMAN [1975], *On the efficient solution of sparse systems of linear and nonlinear equations*, Research Report 46, Dept. Computer Science, Yale Univ., New Haven, CT.
- [19] B. SPEELPENNING [1978], *The generalized element method*, Tech. Report UIUCDCS-R-78-946, Dept. Computer Science, Univ. Illinois at Urbana-Champaign.
- [20] V. STRASSEN [1969], *Gaussian elimination is not optimal*, Numer. Math., 13, pp. 354-356.
- [21] W. F. TINNEY [1969], *Comments on using sparsity techniques for power system problem*, in Sparse Matrix Proceedings, R. A. Willoughby, ed., IBM Research Rept. RA1 3-12-69, pp. 25-34.

## A NUMERICAL SIMULATION OF A BINARY ALLOY SOLIDIFICATION PROCESS\*

A. D. SOLOMON†, V. ALEXIADES‡ AND D. G. WILSON†

**Abstract.** We describe the numerical implementation of a mathematical model of a binary alloy solidification process introduced in [1]. The model takes into account the “mushy” zone of dendritic growth between liquid and solid phase regions.

**Key words.** alloy solidification, mathematical model, coupled nonlinear conduction-diffusion system, explicit finite differences

**Introduction.** Simulating the diffusion of solute and transfer of heat in the solidification of an alloy is of key importance to developing control strategies for attaining alloys of desired constitution. Current alloy solidification models are generally devoted to one of two distinct viewpoints: 1) processes describable by spatially extensive diffusion fields for heat and solute; or 2) interfacial processes, occurring locally and accounting for the detailed molecular structures and energy transfers characterizing the freezing process. Work stressing the former is typified by the treatment of Rubinstein [5] or more recently, Tao [7]. The latter area is the focus of extensive work over the last three decades, and is typified by [2], [3], [8].

In [1] we began to bridge the gap between the two approaches in a thermodynamically consistent manner. Using the liquid fraction and the chemical potential in the (possibly dendritic) “mushy” zone, the model in principle obviates the need for empirical models of the mushy zone [6]. In the same sense it extends the earlier weak solution formulation of [9].

The need for a model of the solidification process which admits the possibility of a finite dendritic zone was noted in [10] where a classical Stefan type formulation was shown to be incomplete. In the following lines we present an explicit numerical scheme for the implementation of the model [1]. The scheme can be extended to two and three dimensions; while some questions of convergence and numerical behavior remain, the great difficulties in its implementation lie in the absence of known values of the thermophysical parameters used.

In § 1 we recall the equations underlying the model; these are then used to generate the finite difference model of § 2. Finally in § 3 we present a number of sample calculations for a Cu-Ni binary system.

**1. The mathematical model.** Consider the solidification of a binary alloy consisting of 2 components,  $A$  and  $B$ . We assume the solidification process to be governed locally by an equilibrium phase diagram as shown in Fig. 1. Here  $C = g_s(T)$ ,  $C = g_l(T)$  are the solidus and liquidus curves, respectively;  $C$  is the concentration, measured as the mass fraction of component  $B$ . We wish to model the following solidification process. Assume that the finite slab

$$0 < x < l$$

initially consists of liquid alloy at temperature

$$T_0(x), \quad 0 < x < l,$$

---

\* Received by the editors October 20, 1983. This research was sponsored by the Applied Mathematics Sciences Research Program, Office of Energy Research, U.S. Department of Energy under contract W-7405-eng-26 with the Union Carbide Corporation.

† Mathematics and Statistics Research, Oak Ridge National Laboratory, Oak Ridge, Tennessee 37831.

‡ Mathematics Department, University of Tennessee, Knoxville, Tennessee 37996.

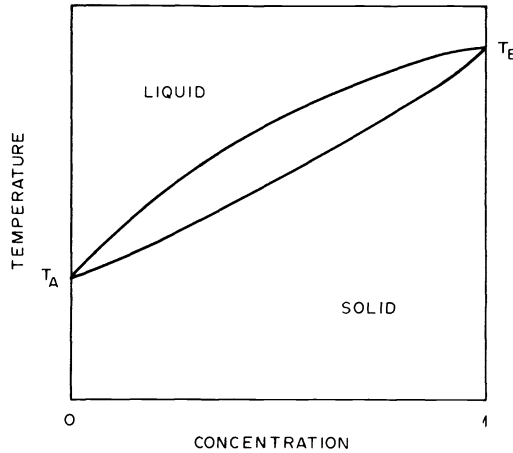


FIG. 1. *Equilibrium phase diagram.*

and concentration

$$C_0(x), \quad 0 < x < l.$$

At  $x = l$  the slab is to be thermally insulated; similarly at  $x = 0, l$ , there is no material flux. Suppose that starting at time  $t = 0$  the temperature at the face  $x = 0$  is set equal to a constant value  $T_s$ , low enough to induce a freezing process beginning at  $x = 0$ . At any later time  $t > 0$ , material at points of the slab may be in one of three states: liquid (L), solid (S) and mushy (or dendritic) (M).

The mushy zone represents a region of constitutional supercooling [2] in which dendrites are separated by liquid zones. Through the width of the zone we may expect liquid and solid fraction to vary—in a manner depending on the orientation and shape of these dendrites.

We define the liquid fraction  $\lambda(C, T)$  as

$$(1) \quad \lambda(C, T) = \begin{cases} 1 & \text{in the liquid } (C \leq g_L(T)), \\ 0 & \text{in the solid } (C \geq g_S(T)), \\ \frac{C - g_S(T)}{g_L(T) - g_S(T)} & \text{in the mush } (g_L(T) < C < g_S(T)). \end{cases}$$

As in [1] we introduce the mass and heat fluxes

$$(2) \quad J = -\{DC_x - \delta T_x\},$$

$$(3) \quad G = -\{KT_x - \beta C_x\},$$

where  $D$  is the material diffusivity,  $K$  the thermal conductivity, and  $\delta$  and  $\beta$  are the Soret and Dufour coefficients. All these transport coefficients depend, in general, on both temperature and concentration and may jump discontinuously from liquid to solid. The primary effects are represented by  $D$  and  $K$ , while the cross effects coefficients  $\delta$  and  $\beta$  are usually much smaller and are commonly neglected. A list of all the thermophysical parameters entering the model appears in § 3.

As it is shown in detail in [1], the conservation laws for mass and energy assume the form

$$(4) \quad C_t = -J_x,$$

$$(5) \quad \rho(c + H\lambda_T)T_t = -G_x - \rho(T\mu_T - H\lambda_C)J_x - \rho\mu_x J.$$

Here, subscripts denote partial derivatives. The density,  $\rho$ , of the alloy is assumed to be constant in order to eliminate material flow;  $c(C, T) = \lambda c_L + (1 - \lambda)c_s$  denotes the specific heat,  $H$  the latent heat of fusion, and  $\mu$  the difference of the chemical potentials of the two components, chosen here as

$$(6) \quad \mu(C, T) = RT \ln \left( \gamma \frac{C}{1 - C} \right),$$

with  $R$  the gas constant and  $\gamma$  the ratio of chemical activities (cf. [4]).

The partial differential equations (4) and (5) are valid globally over the region occupied by the alloy, irrespective of phase. Of course since many of the quantities may not be differentiable across the liquidus and solidus, these equations must be interpreted in a weak (distributional) sense, as discussed in [1].

The boundary and initial conditions for the model are

$$(7) \quad T(0, t) = T_s,$$

$$(8) \quad J(0, t) = 0,$$

$$(9) \quad G(l, t) = 0,$$

$$(10) \quad J(l, t) = 0.$$

**2. Finite difference scheme.** For  $M$  a natural number, let

$$M \Delta x = l,$$

and  $x_j = j\Delta x, j = 0, \dots, M$ . Let  $\Delta t > 0$ , and  $t_n = n\Delta t, n = 0, 1, 2, \dots$ . For any function  $f(x, t)$ , we denote

$$f_j^n = f(x_j, t_n).$$

The numerical scheme we use is explicit in time, updating the concentration vector  $C_j^n$  and temperature vector  $T_j^n, j = 0, \dots, M$  at time step  $n$ , to their values  $C_j^{n+1}, T_j^{n+1}$  at time step  $n + 1$ .

Assume  $C_j^n, T_j^n, j = 0, \dots, M$  to be known. Following (1), we define the liquid fraction array  $\lambda_j^n, j = 0, \dots, M$  via

$$(11) \quad \lambda_j^n = \begin{cases} 1 & \text{for } (C_j^n, T_j^n) \text{ a liquid point on the phase diagram,} \\ 0 & \text{for } (C_j^n, T_j^n) \text{ a solid point on the phase diagram,} \\ \frac{C_j^n - g_s(T_j^n)}{g_L(T_j^n) - g_s(T_j^n)} & \text{for } (C_j^n, T_j^n) \text{ a mushy point on the phase diagram.} \end{cases}$$

If  $(C_j^n, T_j^n)$  is a mushy point on the phase diagram, then we consider the interval

$$x_j - \frac{1}{2} \Delta x < x < x_j + \frac{1}{2} \Delta x$$

to consist of intermixed solid and liquid, at a proportion determined by  $\lambda_j^n$ , in thermal equilibrium for the temperature  $T_j^n$ . Hence a liquid volume  $\lambda_j^n \Delta x$  at temperature  $T_j^n$  and concentration  $g_L(T_j^n)$  and a solid volume  $(1 - \lambda_j^n) \Delta x$  at the same temperature  $T_j^n$  and concentration  $g_s(T_j^n)$  are assumed to fill the interval. We further assume that solid and liquid concentration gradients induce diffusion from solid to solid regions and from liquid to liquid regions in neighboring mesh intervals, while no material flow

will occur between solid and liquid regions. Hence for  $j=0, \dots, M$  we define

$$(12) \quad CS_j^n = \begin{cases} C_j^n & \text{if } \lambda_j^n = 0, \\ g_s(T_j^n) & \text{if } 0 < \lambda_j^n < 1, \\ 0 & \text{if } \lambda_j^n = 1, \end{cases}$$

$$(13) \quad CL_j^n = \begin{cases} C_j^n & \text{if } \lambda_j^n = 1, \\ g_L(T_j^n) & \text{if } 0 < \lambda_j^n < 1, \\ 0 & \text{if } \lambda_j^n = 0. \end{cases}$$

We define the effective solid and liquid material fluxes from node  $j$  to node  $j+1$  as

$$(14) \quad JS_{j+1/2}^n = -\frac{1}{\Delta x} \{D_s(CS_{j+1}^n - CS_j^n) - \delta_s(T_{j+1}^n - T_j^n)\},$$

$$(15) \quad JL_{j+1/2}^n = -\frac{1}{\Delta x} \{D_L(CL_{j+1}^n - CL_j^n) - \delta_L(T_{j+1}^n - T_j^n)\}.$$

Hence we define the effective material flux from node  $j$  to node  $j+1$  as

$$(16) \quad J_{j+1/2}^n = \min(\lambda_j^n, \lambda_{j+1}^n)JL_{j+1/2}^n + \min(1 - \lambda_j^n, 1 - \lambda_{j+1}^n)JS_{j+1/2}^n;$$

this corresponds to the assumption that  $\min(\lambda_j^n, \lambda_{j+1}^n)$  is a measure of the degree of direct liquid-liquid contact between node intervals  $j, j+1$ , and similarly for the solid. Thus the final discretization of (4) is

$$(17) \quad \frac{C_j^{n+1} - C_j^n}{\Delta t} = -\left\{ \frac{J_{j+1/2}^n - J_{j-1/2}^n}{\Delta x} \right\}, \quad j = 1, \dots, M - 1,$$

and updates the concentration to time step  $n+1$ .

The discretization of the energy conservation equation (5) is carried out in a similar manner. For  $j=0, \dots, M$  let

$$A_j^n = \rho(c + H\lambda_T)_j^n$$

where  $c = \gamma c_L + (1 - \lambda)c_s$ .

Now rewrite (5) as

$$(18) \quad AT_t = -G_x - \rho(T\mu_T - H\lambda_C - \mu)J_x - \rho(\mu J)_x.$$

Set

$$B = \rho(T\mu_T - H\lambda_C - \mu);$$

for our present choice (6) of the chemical potential,  $T\mu_T - \mu = 0$  and from (1) we see that

$$(19) \quad B = -\rho H / (g_L(T) - g_s(T)).$$

Thus (18) becomes

$$(20) \quad AT_t = -G_x - BJ_x - \rho(\mu J)_x,$$

which upon discretization yields

$$(21) \quad A_j^n \frac{(T_j^{n+1} + T_j^n)}{\Delta t} = -\left\{ \frac{G_{j+1/2}^n - G_{j-1/2}^n}{\Delta x} \right\} - B_j^n \left\{ \frac{J_{j+1/2}^n - J_{j-1/2}^n}{\Delta x} \right\} - \rho \left\{ \frac{(\mu J)_{j+1/2}^n - (\mu J)_{j-1/2}^n}{\Delta x} \right\}, \quad j = 1, \dots, M - 1.$$



The terms  $J_{j+1/2}$  are already known from (16). On the other hand we treat each of the terms of  $G = -KT_x + \beta C_x$  separately as follows.

Considering the liquid and solid fractions of adjacent mesh intervals as indicating the extent of solid-liquid, liquid-liquid and solid-solid contact, we define the thermal conductivity  $K_{j+1/2}^n$  as

$$(22) \quad K_{j+1/2}^n = |\lambda_{j+1}^n - \lambda_j^n| \frac{2K_s K_L}{K_s + K_L} + \min(\lambda_j^n, \lambda_{j+1}^n) K_L + \min(1 - \lambda_j^n, 1 - \lambda_{j+1}^n) K_s.$$

The first term on the right of (22) measures the contribution from the solid-liquid contact and results by averaging the resistivities  $1/K_s$  and  $1/K_L$ . Then we define

$$(23) \quad \langle KT_x \rangle_{j+1/2}^n = K_{j+1/2}^n \left\{ \frac{T_{j+1}^n - T_j^n}{\Delta x} \right\}.$$

Similarly, let

$$(24) \quad WL_{j+1/2}^n = \frac{1}{\Delta x} \{ \beta_L (CL_{j+1}^n - CL_j^n) \},$$

$$(25) \quad WS_{j+1/2}^n = \frac{1}{\Delta x} \{ \beta_s (CS_{j+1}^n - CS_j^n) \}$$

with  $CS, CL$  given by (12), (13).

Then

$$(26) \quad \langle \beta C_x \rangle_{j+1/2}^n = \min(\lambda_j^n, \lambda_{j+1}^n) WL_{j+1/2}^n + \min(1 - \lambda_j^n, 1 - \lambda_{j+1}^n) WS_{j+1/2}^n,$$

whence  $G_{j+1/2}^n$  is completely defined:

$$(27) \quad G_{j+1/2}^n = -\langle KT_x \rangle_{j+1/2}^n + \langle \beta C_x \rangle_{j+1/2}^n.$$

The expression  $\langle \mu J \rangle_{j+1/2}^n$  is defined as

$$(28) \quad \langle \mu J \rangle_{j+1/2}^n = \frac{\mu_{j+1}^n + \mu_j^n}{2} J_{j+1/2}^n.$$

The initial conditions are

$$T_j^0 = T_0(x_j), \quad C_j^0 = C_0(x_j), \quad j = 0, \dots, M.$$

On the other hand the boundary conditions yield updating equations at  $j = 0$  and  $j = M$  of the form

$$(29) \quad (C_0^{n+1} - C_0^n) \frac{\Delta x}{2} = -\Delta t J_{1/2}^n,$$

$$(30) \quad T_0^n = T_s,$$

$$(31) \quad (C_M^{n+1} - C_M^n) \frac{\Delta x}{2} = \Delta t J_{M-1/2}^n,$$

$$(32) \quad \rho (c_M^n + H_M^n \lambda_{T_M}^n) \frac{\Delta x}{2} (T_M^{n+1} - T_M^n) = \Delta t \{ G_{M-1/2}^n + \rho (T_M^n \mu_{T_M}^n - H_M^n \lambda_{C_M}^n) J_{M-1/2}^n + \frac{\rho}{2} (\mu_{j-1}^n - \mu_j^n) J_{M-1/2}^n \}.$$

Theoretically, the values of all the thermophysical parameters entering the model ( $c, H, D, K, \delta, \beta$ ) can be determined experimentally as functions of the state ( $C, T$ ), at least in the liquid and solid regions. In practice, this is very difficult to do and the available data are scarce. In particular, inside the mushy region it is essentially impossible to measure quantities like  $D, K$ , etc., and therefore we have to amplify the model by incorporating in it detailed descriptions of the transport processes in terms of measurable quantities.

For simplicity in the presentation we shall assume that the physical parameters  $c, D, K, \delta, \beta$  have (possibly different) constant values in the liquid and solid denoted by subscripts  $L$  and  $S$ . For real systems it is already difficult to find even such piecewise constant values. If however one knows the full dependence of, say,  $D_s$  on concentration and temperature, then one can replace  $D_s$  in (14) by, for example,

$$D_s = \frac{1}{2}[D_s(CS_{j+1}^n, T_{j+1}^n) + D_s(CS_j^n, T_j^n)].$$

**3. Some numerical experiments.** A program implementing the scheme of § 2 has been run for a system based on a Cu-Ni alloy. As for many other binary systems, the solidus and liquidus curves for Cu-Ni may be adequately approximated by parabolas. Let  $\tau = (T - T_A)/(T_B - T_A)$ . Then we can approximate the solidus and liquidus curves by quadratics of the form

$$C = g_L(T) = [\alpha + (1 - \alpha)\tau]\tau,$$

$$C = g_S(T) = [\beta + (1 - \beta)\tau]\tau.$$

The phase diagram of Cu-Ni is well approximated by choosing [11]

$$T_A = 1356 \text{ K}, \quad \alpha = .7956,$$

$$T_B = 1728 \text{ K}, \quad \beta = 1.555.$$

The thermophysical parameters about whose values we have some confidence are:

Parameter	Name	Value	Units
$c_L$	liquid alloy specific heat	.4	KJ/KG-K
$c_s$	solid alloy specific heat	.4	KJ/KG-K
$\rho$	density	$8 \times 10^3$	Kg/m <sup>3</sup>
$K_L$	liquid alloy conductivity	.1	KJ/M-s-K
$K_s$	solid alloy conductivity	.3	KJ/M-s-K
$H$	latent heat of alloy	200	KJ/Kg
$T_A$	copper melt temperature	1356	K
$T_B$	nickel melt temperature	1728	K
$\alpha$	Phase diagram parameter	.7956	
$\beta$	Phase diagram parameter	1.555	

Thermophysical parameters about which we are less certain are:

Parameter	Name	Value	Units
$D_s$	solid material diffusivity	?	m <sup>2</sup> /s
$D_L$	liquid material diffusivity	?	m <sup>2</sup> /s
$\beta_s$	coupling term in (3) (solid phase)	?	KJ/m-s
$\beta_L$	coupling term in (3) (liquid phase)	?	KJ/m-s
$\delta_s$	coupling term in (2) (solid phase)	?	m <sup>3</sup> /s <sup>2</sup> -K
$\delta_L$	coupling term in (2) (liquid phase)	?	m <sup>3</sup> /s <sup>2</sup> -K
$\gamma$	chemical activity ratio	?	

Let us describe two runs typical of those made thus far.

**Run 1.** In this case the coupling terms  $\beta$ ,  $\delta$  are taken as zero. Values chosen were:

- |  |  |
|--|--|
| $c_s = c_L = .4 \text{ KJ/Kg-K}$                 | $\gamma = 1$   |
| $\rho = 8 \times 10^3 \text{ Kg/m}^3$            | $T_A = 1356 \text{ K}$   |
| $K_L = .1 \text{ KJ/m-s-K}$                      | $T_B = 1728 \text{ K}$   |
| $K_s = .3 \text{ KJ/m-s-K}$                      | $l = \text{length of slab} = .25 \text{ m}$                                |
| $H = 200 \text{ KJ/Kg}$                          | $\Delta x = \text{mesh width} = 0.1 \text{ m}$                             |
| $D_s = 1 \times 10^{-6} \text{ m}^2/\text{s}$    | $\Delta t = \text{time mesh width} = .25 \text{ s}$                        |
| $D_L = 1 \times 10^{-5} \text{ m}^2/\text{s}$    | $M = \text{number of space mesh intervals} = 25$                           |
| $\delta_L = 0 \text{ m}^3/\text{s}^2 - \text{K}$ | $T_{\text{init}} = \text{uniform initial temperature} = 1600 \text{ K}$    |
| $\delta_s = 0 \text{ m}^3/\text{s}^2 - \text{K}$ | $T_{\text{wall}} = \text{imposed (cold) wall temperature} = 100 \text{ K}$ |
| $\beta_L = 0 \text{ KJ/m-s}$                     | $C_{\text{init}} = \text{uniform initial weight fraction} = .2$            |
| $\beta_s = 0 \text{ KJ/m-s}$                     |  |

In Fig. 2 we see the phase change history over 570 s of simulated time. The mushy zone ( $M$ ) is expanding steadily; the anomaly at  $t = 495 \text{ s}$  is due to slight reduction (at the fourth decimal place) of liquid fraction.

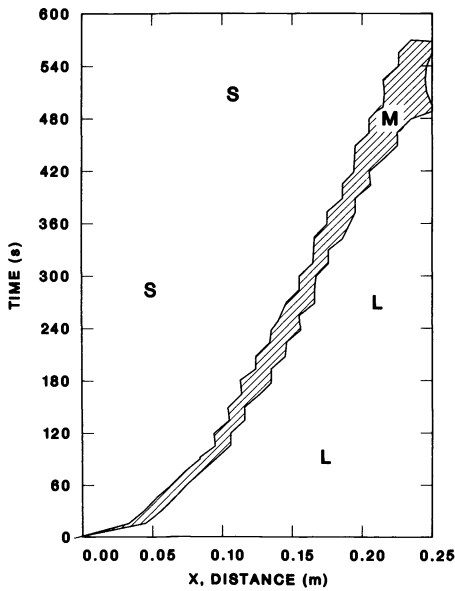


FIG. 2. Phase change history for Run 1.

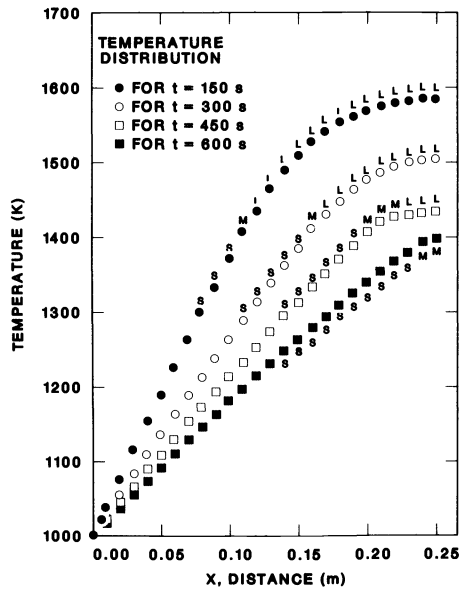


FIG. 3. Temperature distributions for Run 1.

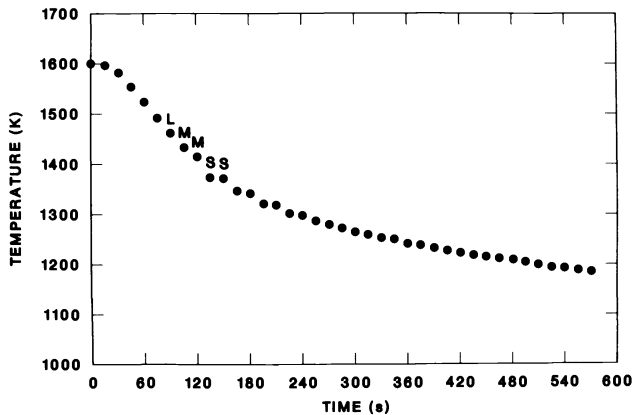


FIG. 4. Temperature history at  $x = .1$  for Run 1.

In Fig. 3 we see the temperature distributions at times 150, 300, 450 and 600 s. Note that the temperature is essentially linear through the solid.

In Fig. 4 we see the temperature history at  $x = .1$  over the first 570 seconds. We again believe that numerical noise, typical for weak solution methods, is the reason for the “jumpiness” of the values following solidification.

In Fig. 5 we see the final concentration profile (Ni) of the system at  $t = 637.5$  s. Note the slight rise of the concentration in position up to  $x \approx .17$ . We note too that mass is conserved in our code for all cases.

Fig. 6 shows the weight fraction history at  $x = .1$ . Similarly, Fig. 7 shows the state history of the node at  $x = .1$  at 30 s intervals.

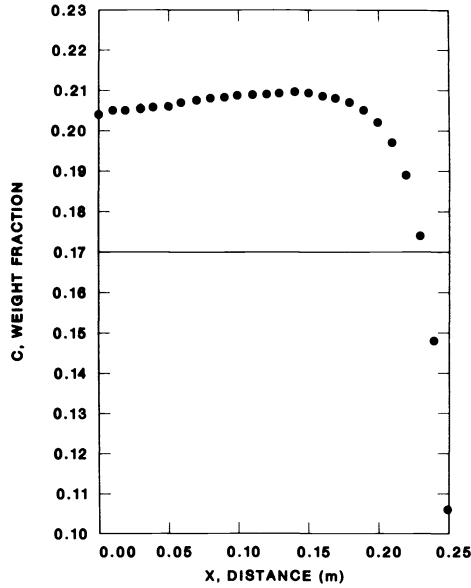


FIG. 5. Final weight fraction distribution at time  $t = 637.5$  s for Run 1.

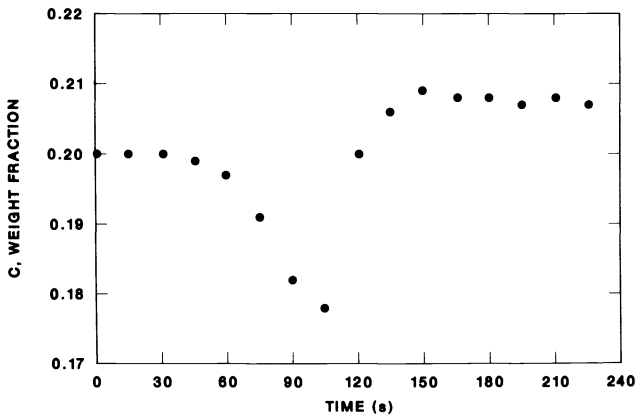


FIG. 6. Weight fraction history at  $x = .1$  for Run 1.

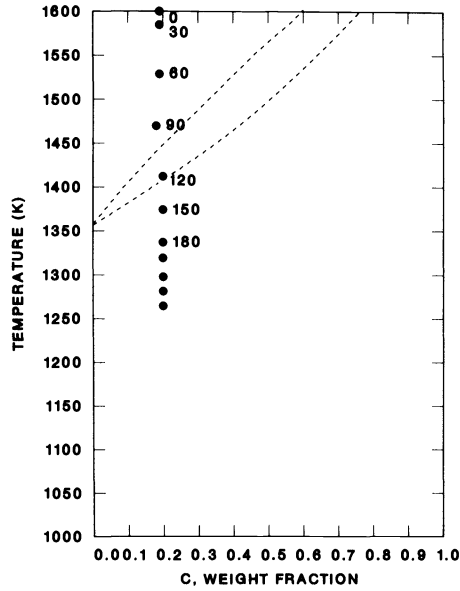


FIG. 7. History of state at  $x = .1$  for Run 1.

Run 2 was made with the same thermophysical parameters as those of Run 1, except for the following:

Parameter	Run 1	Run 2
$D_s$	$1 \times 10^{-6}$	$1 \times 10^{-8}$
$D_L$	$1 \times 10^{-5}$	$1 \times 10^{-6}$
$\delta_L$	0	$1 \times 10^{-12}$
$\delta_s$	0	$1 \times 10^{-12}$
$\beta_L$	0	$1 \times 10^{-2}$
$\beta_s$	0	$1 \times 10^{-2}$

Hence direct coupling is now introduced. The results obtained are shown in Figs. 8-13.

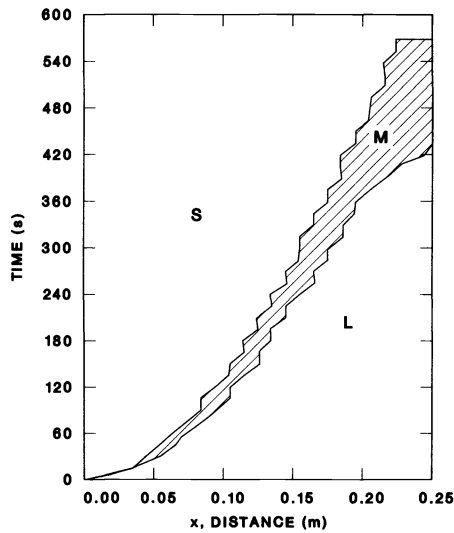


FIG. 8. Phase change history for Run 2.

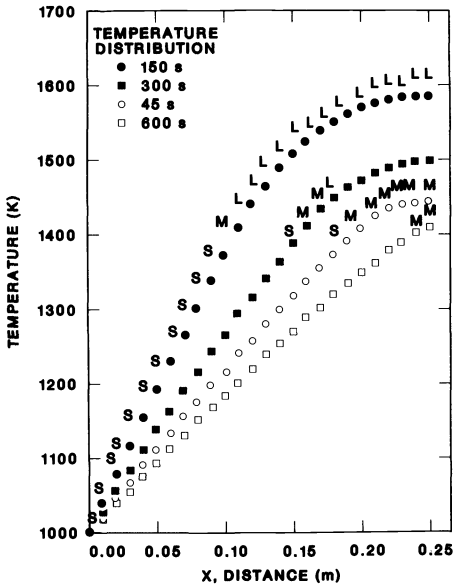


FIG. 9. Temperature distribution for Run 2.

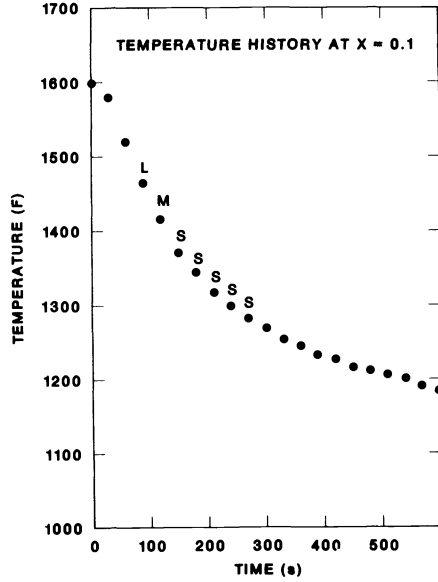


FIG. 10. Temperature history at  $x = .1$  for Run 2.

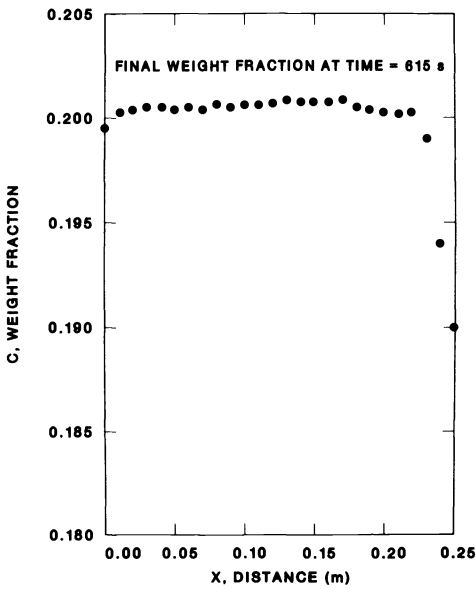


FIG. 11. Final weight fraction at time = 625 s for Run 2.

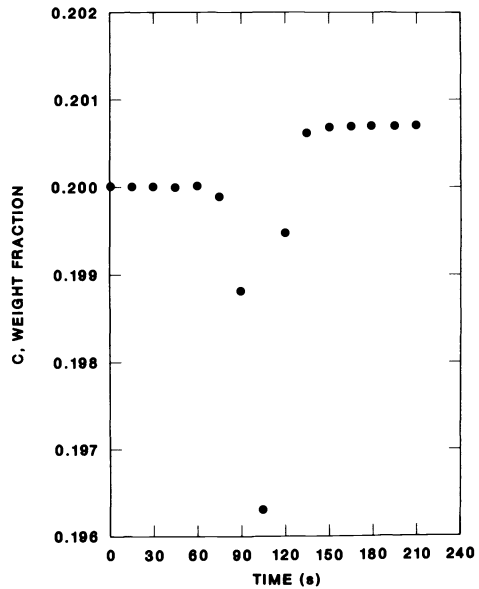


FIG. 12. Weight fraction history at  $x = .1$  for Run 2.

A comparison of Figs. 2 and 8 shows that decreasing  $D_s$  broadens the mushy zone very much, as is expected. We note from Figs. 3 and 9 that the material cools more slowly with  $D_s$ ,  $D_L \neq 0$ .

**4. Conclusions and future goals.** In conclusion, we note that the model we have proposed in [1] can be implemented in a direct fashion, and yields expected results while conserving total energy and mass. Future developments of this model will include: a) resolution of the steady state for the model; b) extension of the model to make

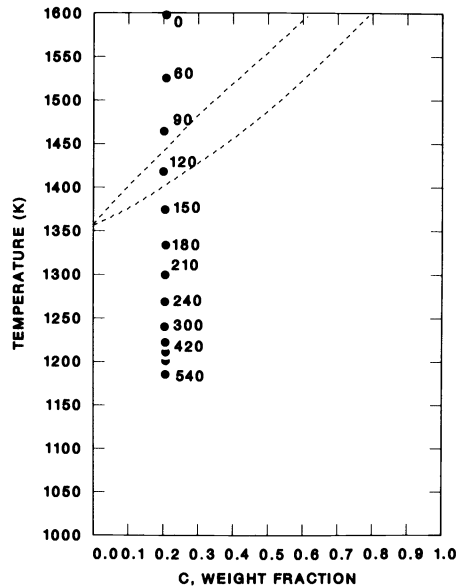


FIG. 13. State history at  $x = .1$  for Run 2.

possible comparisons with available experimental data; c) analysis of the numerical scheme; d) approaches to well-posedness of the underlying mathematical model, and convergence of the numerical method.

**Acknowledgments.** The authors wish to express their gratitude to Dr. Rohit Trivedi of the Ames Laboratory for his advice and encouragement of the work described. Similarly we would like to express our deep appreciation to Al Geist for programming the scheme and obtaining the numerical results and to Tammy Reed for typing and preparing this paper.

*Note added in proof.* An improved description of the basic model is to appear in V. Alexiades, D. G. Wilson and A. D. Solomon, *Macroscopic global modeling of binary alloy solidification processes*, Quart. Appl. Math.

Currently the model is being implemented on a mercury-cadmium-telluride alloy, of which the thermophysical properties strongly depend on temperature and concentration. Results will be reported in a forthcoming ORNL Report.

#### REFERENCES

- [1] V. ALEXIADES, D. WILSON AND A. SOLOMON, *Modeling binary alloy solidification processes*, Union Carbide Corporation, Report ORNL/CSD-117, 1983.
- [2] J. CHRISTIAN, *The Theory of Transformations in Metals and Alloys*, Pergamon, Oxford, 1965.
- [3] W. MULLINS AND R. SEKERKA, *Stability of a planar interface during solidification of a dilute binary alloy*, J. Appl. Phys., 35 (1964), pp. 444-451.
- [4] R. H. PARKER, *An Introduction to Chemical Metallurgy*, 2nd ed., Pergamon, Oxford, 1978.
- [5] L. RUBINSTEIN, *The Stefan Problem*, AMS Translations, Vol. 27, American Mathematical Society, Providence, RI, 1971.
- [6] A. SOLOMON, D. WILSON AND V. ALEXIADES, *A mushy zone model with an exact solution*, Letters in Heat and Mass Transfer, 9 (1982), pp. 319-324.
- [7] L. TAO, *On solidification of a binary alloy*, Quart. Appl. Math., 33 (1980), pp. 211-225.
- [8] R. TRIVEDI, *Theory of dendritic growth during the directional solidification of binary alloys*, J. Crystal Growth, 49 (1980), pp. 219-232.

- [9] D. WILSON, A. SOLOMON AND V. ALEXIADES, *A model of binary alloy solidification*, Union Carbide Corporation, Report ORNL/CSD-97, 1982.
- [10] ———, *A shortcoming of the explicit solution for the binary alloy solidification problem*, Letters in Heat and Mass Transfer, 9 (1982), pp. 421-428.
- [11] D. WILSON, A. LACEY AND A. SOLOMON, *Composition of solidified binary alloy from a simple solidification model*, Union Carbide Corporation, Report ORNL/CSD-66, 1980.



## STUDIES IN NUMERICAL NONLINEAR INSTABILITY I. WHY DO LEAPFROG SCHEMES GO UNSTABLE?\*

J. M. SANZ-SERNA†

**Abstract.** It is well known that leapfrog (explicit mid-point) discretizations of partial differential equations may have unbounded solutions for any choice of mesh-sizes, (even for choices satisfying conditions for linear stability). We provide a means for forecasting the qualitative behaviour of the computed leapfrog points, thus explaining the dynamics of the nonlinear instability phenomenon.

**Key words.** nonlinear instability, leapfrog schemes, dynamical systems

**1. Introduction.** Leapfrog (explicit mid-point) schemes are often used in those meteorological or oceanographic computations where the interest lies in monitoring the global evolution of physical magnitudes over long periods of time. In these circumstances, nondissipative leapfrog schemes may be more advisable than some dissipative alternative methods [14]. However, the lack of dissipativity, while preventing gross global losses of energy, vorticity, etc., . . . , entails some disadvantages from the stability point of view. Here the word stability must be understood to refer to the behaviour of the numerical solution for *fixed* values of the mesh-sizes, as the number of computed time-levels grows. (As distinct from the notions of Lax-Richtmyer stability or Dahlquist stability [16], which provide conditions related to the concept of convergence as the mesh-sizes tend to zero.)

In linear, constant coefficient problems with suitable boundary conditions, the stability of leapfrog schemes is easily investigated, for in such cases the discrete equations are solvable in closed form, via Fourier analysis. Often a relationship between the various mesh-sizes (the so-called linear stability condition) can be derived which ensures boundedness of the numerical solution as the number of computed levels grows.

In linear, variable coefficient or nonlinear problems describing wave phenomena, it may happen that the leapfrog solution is unbounded for *any choice of meshsizes* (even for choices which satisfy the linear stability conditions associated with all the problems obtainable from the given one by linearizing and freezing the coefficients.) This fact was first noted by Phillips [17] in the nonlinear case and Miyakoda [13] in the variable coefficient case. Phillips attributed this offending behaviour (the so-called *nonlinear instability*) to the presence of *aliasing*. However Arakawa [1] constructed a continuous-time difference scheme for the inviscid vorticity transport equation which suffers from aliasing and yet exactly conserves vorticity, its square and kinetic energy, thus ensuring boundedness of the computed solution. (See Morton [14] for an excellent survey of the role played by *quadratic conserved quantities* and its relation with Galerkin's method.) When Arakawa's scheme is discretized in time by means of the leapfrog technique, the quadratic invariants are only *approximately* conserved and nonlinear stability can arise, (cf. [19]). In practice, leap-frog schemes must be supplemented by filtering or artificial viscosity [12] in order to prevent the onset of nonlinear instability. A modification of the leapfrog technique which is free from the occurrence of blowups has been introduced and analyzed in [19], [21], [22].

In this paper we present a technique, whereby *the qualitative behaviour of leapfrog approximations can be forecast* a priori (or explained a posteriori). In particular we provide an explanation of the nonlinear instability phenomenon.

---

\* Received by the editors October 10, 1983, and in revised form July 11, 1984.

† Departamento de Ecuaciones Funcionales, Facultad de Ciencias, Universidad de Valladolid, Valladolid, Spain.

Our research was inspired by a paper by Ushiki [26].

Little is known concerning the qualitative behaviour, for fixed values of the mesh-sizes, of discretizations of nonlinear evolutionary problems. In the ordinary differential equation field, interest has been centered around the issue of *contractivity* (see Dekker and Verwer [4] for a thorough summary). The experience gained by Ushiki [26] and the contents of the present paper, appear to suggest that the investigation of stability properties may benefit from an interaction with the field of *dynamical systems*. However our study does not rely heavily on concepts of that field and the reader only needs to be familiar with the basic elementary techniques in the qualitative study of ordinary differential systems (see among others [7]).

Another useful connection is that between nonlinear instability in numerical analysis and nonlinear stability in fluid mechanics, explored by Newell [15].

An outline of the paper is as follows. Section 2 contains the main idea. The linear and nonlinear ordinary differential equation cases are described in §§ 3 and 5, respectively. Partial differential equations are investigated in § 6. The fourth section is devoted to some technical results and the seventh contains several concluding remarks.

**2. The augmented system.** We consider initial value problems for the system

$$(2.1) \quad \mathbf{y}' = \mathbf{f}(\mathbf{y}),$$

where a prime denotes differentiation with respect to  $t$  and  $\mathbf{f} : \mathbb{R}^d \rightarrow \mathbb{R}^d$  is a smooth ( $C^4$  say) function. The system (2.1) is discretized by means of the mid-point rule

$$(2.2) \quad \mathbf{y}_{n+2} = \mathbf{y}_n + 2h\mathbf{f}(\mathbf{y}_{n+1}),$$

where the step-length  $h$  is positive. If we fix a solution  $\mathbf{y}(t)$  of (2.1) and choose  $\mathbf{y}_0, \mathbf{y}_1$ , close to  $\mathbf{y}(0), \mathbf{y}(h)$  respectively, then each point  $\mathbf{y}_n, n = 2, 3, \dots$  generated by (2.2) will be “close” to the point  $\mathbf{y}(nh), n = 2, 3, \dots$ . In more (but not too) precise terms, if we consider a *family*  $\mathbf{y}_0^h, \mathbf{y}_1^h$  of starting points, with  $h$  ranging in an interval  $(0, h_0), h_0 > 0$ , then [10, p. 22]

$$(2.3) \quad \lim_{\substack{h \rightarrow 0 \\ nh = a}} \mathbf{y}_n^h = \mathbf{y}(a) = \mathbf{y}(nh),$$

provided that

$$(2.4) \quad \lim_{h \rightarrow 0} \mathbf{y}_0^h = \lim_{h \rightarrow 0} \mathbf{y}_1^h = \mathbf{y}(0).$$

However, this *convergence* property does not imply that for a given, *fixed* value of  $h$ , and given  $\mathbf{y}_0, \mathbf{y}_1$  (close to  $\mathbf{y}(0), \mathbf{y}(h)$ , respectively), the sequences in  $\mathbb{R}^d$

$$(2.5) \quad \mathbf{y}_0, \mathbf{y}_1, \mathbf{y}_2, \dots, \mathbf{y}_n, \dots,$$

$$(2.6) \quad \mathbf{y}(0), \mathbf{y}(h), \mathbf{y}(2h), \dots, \mathbf{y}(nh), \dots,$$

exhibit the same *qualitative behaviour*. In fact, it is well known [6, p. 241] that if (2.1) is linear

$$(2.7) \quad \mathbf{y}' = A\mathbf{y}$$

and the spectrum of  $A$  is contained in  $\{z : \text{Re } z < 0\}$ , so that  $\lim_n \mathbf{y}(nh) = \mathbf{0}$ , the leapfrog points  $\mathbf{y}_n$  will, in general, be unbounded no matter how close  $\mathbf{y}_0, \mathbf{y}_1$  are to  $\mathbf{y}(0), \mathbf{y}(h)$ .

In this paper a method is given for describing the qualitative behaviour of (2.5) for fixed  $h$ . We start by introducing the sequence in  $\mathbb{R}^{2d}$

$$(2.8) \quad [\mathbf{y}_0, \mathbf{y}_1], [\mathbf{y}_2, \mathbf{y}_3], \dots, [\mathbf{y}_{2n}, \mathbf{y}_{2n+1}], \dots$$

It is obvious that the knowledge of (2.8) implies that of (2.5) and vice versa. Nevertheless (2.8) is more advantageous in that each “augmented” vector  $[y_{2m}, y_{2m+1}]$  is obtained from the previous vector  $[y_{2n-2}, y_{2n-1}]$  by means of the *one-step* recursion

$$(2.9) \quad [y_{2m}, y_{2m+1}] = T[y_{2n-2}, y_{2n-1}],$$

where  $T: \mathbb{R}^{2d} \rightarrow \mathbb{R}^{2d}$  is the mapping

$$(2.10) \quad T[\mathbf{p}, \mathbf{q}] = [\mathbf{p} + 2h\mathbf{f}(\mathbf{q}), \mathbf{q} + 2h\mathbf{f}(\mathbf{p} + 2h\mathbf{f}(\mathbf{q}))].$$

One iteration of (2.9) is equivalent to *two* iterations of (2.2). The main point of this paper consists of the observation that (2.10) provides a *stable*, first-order *consistent*, one-step method for the numerical study of the system in  $\mathbb{R}^{2d}$ ,

$$(2.11) \quad \mathbf{q}' = \mathbf{f}(\mathbf{q}), \quad \mathbf{q}' = \mathbf{f}(\mathbf{p}),$$

which we call *augmented system* associated with the *original system* (2.1). More precisely, if we fix a solution  $\mathbf{p}(t), \mathbf{q}(t)$  of (2.11), then

$$(2.12) \quad \lim_{\substack{h \rightarrow \infty \\ 2nh = a}} T^n[\mathbf{p}_0^h, \mathbf{q}_0^h] = [\mathbf{p}(a), \mathbf{q}(a)]$$

provided that  $\mathbf{p}_0^h, \mathbf{q}_0^h$  tend to  $\mathbf{p}(0), \mathbf{q}(0)$ .

(The proof of this convergence result is trivial and will not be given here.) We claim that the qualitative behaviour of (2.8) is governed by the qualitative behaviour of sequences

$$(2.10) \quad [\mathbf{p}(0), \mathbf{q}(0)], [\mathbf{p}(h), \mathbf{q}(h)], \dots, [\mathbf{p}(nh), \mathbf{q}(nh)], \dots,$$

where  $\mathbf{p}(t), \mathbf{q}(t)$ , is a solution of the augmented system (2.11).

Before we justify our claim, which concerns the fixed  $h, n \rightarrow \infty$  behaviour of the computed solutions, let us examine the  $h \rightarrow 0, nh$ -fixed behaviour. We consider again a family of step-lengths  $h, 0 < h < h_0$ ; fix a solution  $\mathbf{y}(t)$  of the original system and choose  $\mathbf{y}_0^h, \mathbf{y}_1^h$  satisfying (2.4). This implies, after (2.3), convergence of  $\mathbf{y}_n^h$  towards  $\mathbf{y}(nh)$  ( $nh$  constant).

On the other hand, consideration of the augmented recursion (2.9), implies after (2.12) that

$$(2.13) \quad \lim_{\substack{h \rightarrow \infty \\ 2nh = a}} \mathbf{y}_{2n}^h = \mathbf{p}(a), \quad \lim_{\substack{h \rightarrow \infty \\ 2nh = a}} \mathbf{y}_{2n+1}^h = \mathbf{q}(a)$$

where  $\mathbf{p}(t), \mathbf{q}(t)$  is the solution of the augmented system satisfying  $\mathbf{p}(0) = \mathbf{q}(0) = \mathbf{y}(0)$ . Now it is obvious that this solution is given by  $\mathbf{p}(t) = \mathbf{q}(t) = \mathbf{y}(t)$ , so that (2.13) is just a restatement of (2.3). Thus, in the study of the  $h \rightarrow 0, nh$ -fixed case the introduction of the augmented system does not bring any new information. It can be proved that this conclusion is not altered if orders of convergence are taken into account. Incidentally, we note that only *diagonal solutions* of the augmented system (i.e. solutions with  $\mathbf{p}(t) \equiv \mathbf{q}(t)$ ) have appeared in our argument: clearly there is a one-to-one correspondence between diagonal solutions of the augmented system and solutions of the original problem (2.1). It will turn out that in the study of the fixed  $h$  behaviour *nondiagonal solutions* of the augmented system will play an important role.

We now fix a solution  $\mathbf{y}(t)$  of (2.1), fix a “small” value of  $h$  and choose fixed starting vectors  $\mathbf{y}_0, \mathbf{y}_1$  “close” to  $\mathbf{y}(0), \mathbf{y}(h)$  respectively. Our aim is to describe the qualitative behaviour of (2.5). We rely on the concept of *local error*, as employed in the text by Shampine and Gordon [23, p. 22]. In these authors’ words, when  $\mathbf{y}_n$  and  $\mathbf{y}_{n+1}$  have been computed, the best that the method (2.2) can do is to yield a point

$y_{n+2}$  close to  $u(2h)$ , where  $u(t)$  is the *local solution* defined by

$$u' = f(u), \quad u(0) = y_n.$$

Now, upon Taylor expanding we find that

$$y_{n+2} - u(2h) = 2hf(y_{n+1}) - 2hf(u(h)) + E,$$

where  $E$  can be bounded in the form  $\|E\| \leq Ch^3$ , with  $C$  independent of  $h$ . If we were in the  $h \rightarrow 0$  study, we would argue that  $y_{n+1}$  is close to  $u(h)$  and therefore the next computed point is close to  $u(2h)$ , *i.e.* the computed points tend to follow approximately the *local solutions*. (More precisely if  $y_0 = y(0)$  and  $y_1$  is obtained by means of a one-step, first order method, then  $y_{n+1} - u(h) = O(h^2)$  and  $y_{n+2} - u(2h) = O(h^2)$ .) However for fixed  $h, n$  large,  $y_{n+1}$  and  $u(h)$  can be significantly different and thus  $y_{n+2}$  may not follow the local solution.

Let us consider what happens if we describe the computed points in terms of the augmented iteration (2.9). When  $y_{2n-1}, y_{2n}$  have been computed, the method will attempt to approximate the local solution  $v, w$  defined by

$$\begin{aligned} v' &= f(w), & w' &= f(v), \\ v(0) &= y_{2n-2}, & w(0) &= y_{2n-1}, \end{aligned}$$

and now Taylor expansion yields

$$y_{2n} - v(2h) = E_1, \quad y_{2n-1} - w(2h) = E_2$$

with  $\|E_1\| + \|E_2\| \leq Ch^2$ ,  $C$  constant independent of  $h$ . In other words, the computed points will lie near the local solution of the augmented recursion. Note that these local solutions are in general nondiagonal.

Admittedly the previous discussion has been merely *heuristic* and it is difficult to see how a rigorous proof could be given when the term “qualitative” behaviour has not been mathematically defined. We recall that it is possible to define precisely what is meant by the statement “two differential systems have the same qualitative behaviour” [3, p. 92]. This line of thought is not pursued in this paper. However the linear case is *rigorously* treated in the next section.

**3. The linear case.** In this paragraph we consider the linear system (2.7) and assume for simplicity that  $A$  is real and can be diagonalized by a (possibly complex) linear change of variables. Within this section vectors will be allowed to belong to the complex space  $\mathbb{C}^d$ . We define qualitative behaviour as follows.

**DEFINITION 1.** Let  $(a_n), (b_n)$  be sequences of complex numbers. We say that they are linear with the same qualitative behaviour if they are both identically zero or if they are of the form  $a_n = r \exp(nc), b_n = s \exp(nd)$ , with  $r, c, s, d$  complex numbers  $r, s \neq 0$ ,  $\text{sign Re } c = \text{sign Re } d, \text{sign Im } c = \text{sign Im } d$ .

Note that if  $(a_n), (b_n)$  are linear with the same qualitative behaviour then either  $|a_n| \uparrow \infty, |b_n| \uparrow \infty$ , or  $|a_n| = \text{constant}, |b_n| = \text{constant}$  or  $|a_n| \downarrow 0, |b_n| \downarrow 0$ . Also as  $n \uparrow \infty$ , the arguments of  $a_n, b_n$  are either both increasing or both constant or both decreasing. There are *nine* possible qualitative behaviours for sequences other than the identically zero sequence.

For sequences of vectors we resort to uncoupling changes of variables as follows. ( $e_i$  denotes the  $i$ th column of the identity matrix.)

**DEFINITION 2.** If  $(a_n), (b_n)$  are sequences in  $\mathbb{C}^d$ , we say that they are linear with the same qualitative behaviour, if regular matrices  $M, N$  can be found such that, for

each  $i, 1 \leq i \leq d$  the sequences of components  $\mathbf{e}_i^T M \mathbf{a}_n, \mathbf{e}_i^T N \mathbf{b}_n$  are linear with the same qualitative behaviour in the sense of the previous definition.

**THEOREM 1.** *Let  $\mathbf{y}_0, \mathbf{y}_1$  be given vectors in  $\mathbb{C}^d$  and fix  $h$  such that  $|h \operatorname{Im} \lambda_i| < 1$  for each  $\lambda_i$  in  $\operatorname{Spec}(A)$ , with  $A$  as above. Then there is a solution  $\mathbf{p}(t), \mathbf{q}(t)$  of the augmented system associated with  $\mathbf{y}' = A\mathbf{y}$ , such that the sequence (2.10) and the leapfrog sequence (2.8) are linear with the same qualitative behaviour in  $\mathbb{R}^{2d}$ , and furthermore  $\mathbf{p}(0) = \mathbf{y}_0$ .*

*Remark.* Note that here  $\mathbf{y}_0, \mathbf{y}_1$  can be arbitrary. In practice  $\mathbf{y}_0, \mathbf{y}_1$  approximate  $\mathbf{y}(0), \mathbf{y}(h)$ , with  $\mathbf{y}(t)$  a solution of the original system, and therefore the starting augmented vector  $[\mathbf{y}_0, \mathbf{y}_1]$  will be close to the diagonal of  $\mathbb{R}^d \times \mathbb{R}^d$ , (i.e.  $\mathbf{y}_0, \mathbf{y}_1$  will not be widely different).

*Proof.* As usual, it is enough to consider the scalar equation

$$y' = \lambda y, \quad \lambda \in \mathbb{C}, \quad |h \operatorname{Im} \lambda| < 1$$

with augmented system

$$p' = \lambda q, \quad q' = \lambda p.$$

After the change of variables in  $R^2$  given by

$$p + q = P, \quad p - q = Q,$$

the solutions of the augmented systems are of the form  $P = a \exp(\lambda t), Q = b \exp(-\lambda t)$ , with  $a = p(0) + q(0), b = p(0) - q(0)$ . Therefore, in the  $P, Q$  variables, the sequence in (2.10) becomes

$$(3.1) \quad [a(\exp(\lambda h))^n, b(\exp(-\lambda h))^n].$$

Now the theory of linear, constant coefficient difference equations shows that the sequence of computed leapfrog points (2.5) is given by

$$y_n = cr^n + d(-r^{-1})^n$$

where  $c, d$  depend on  $r, y_0, y_1$  and  $r$  is related to  $h$  and  $\lambda$  through a function  $r = g(H), H = \lambda h$  given by

$$g(H) = H + \sqrt{1 + H^2}.$$

Here  $\sqrt{z}$  denotes the square root of  $z$  with argument in the interval  $(-\pi/2, \pi/2]$ . Double roots of the characteristic equation are ruled out by the condition  $|h \operatorname{Im} \lambda| < 1$ . For the  $n$ th term in the augmented sequence (2.8), we have

$$(3.2) \quad \begin{bmatrix} y_{2n} \\ y_{2n+1} \end{bmatrix} = \begin{bmatrix} cr^{2n} + d(r^{-1})^{2n} \\ crr^{2n} - dr^{-1}(r^{-1})^{2n} \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ r & -r^{-1} \end{bmatrix} \begin{bmatrix} c(r^2)^n \\ d(r^{-2})^n \end{bmatrix}.$$

The matrix in (3.2) is regular. (The values  $r = \pm i$  which render it singular are excluded by the requirement  $|\operatorname{Im} H| < 1$ .) Then a change of variables brings (3.2) to the ‘‘uncoupled’’ form  $[c(r^2)^n, d(r^{-2})^n]$ , which is compared with (3.1), after choosing  $p(0), q(0)$  such that  $p(0) + q(0) = 2c, p(0) - q(0) = 2d$ . For this choice  $p(0) = c + d = y_0$ .

We introduce the following subsets of  $\mathbb{C}$ , in correspondence with the possible nontrivial qualitative behaviours:

- $A_1 = \{0\},$
- $A_2 = \{bi: 0 < b < 1\},$
- $A_3 = \{bi: -1 < b < 0\},$
- $A_4 = \{a: 0 < a < \infty\},$

$$\begin{aligned}
 A_5 &= \{a: -\infty < a < 0\}, \\
 A_6 &= \{a + bi: 0 < a < \infty, 0 < b < 1\}, \\
 A_7 &= \{a + bi: 0 < a < \infty, -1 < b < 0\}, \\
 A_8 &= \{a + bi: -\infty < a < 0, 0 < b < 1\}, \\
 A_9 &= \{a + bi: -\infty < a < 0, -1 < b < 0\}.
 \end{aligned}$$

The proof is concluded if we show that

$$\begin{aligned}
 g(A_1) &= \{1\}, \\
 g(A_2) &= \{e^{i\theta}: 0 < \theta < \pi/2\}, \\
 g(A_3) &= \{e^{i\theta}: -\pi/2 < \theta < 0\}, \\
 g(A_4) &= \{a: 1 < a < \infty\}, \\
 g(A_5) &= \{a: 0 < a < 1\}, \\
 g(A_6) &\subset \{\rho e^{i\theta}: \rho > 1, 0 < \theta < \pi/2\}, \\
 g(A_7) &\subset \{\rho e^{i\theta}: \rho < 1, -\pi/2 < \theta < 0\}, \\
 g(A_8) &\subset \{\rho e^{i\theta}: \rho < 1, 0 < \theta < \pi/2\}, \\
 g(A_9) &\subset \{\rho e^{i\theta}: \rho < 1, -\pi/2 < \theta < 0\}.
 \end{aligned}$$

The conditions relative to  $A_i, i = 1-5$  are verified straightforwardly. For  $A_6$  note that  $g$  is univalued and analytic in the closure  $\bar{A}_6$ , except for the branch point at  $H = i$ . It is therefore sufficient to investigate the behaviour of the boundary of  $A_6$  under the mapping  $g(H)$ . That boundary consists of  $A_1, A_2, A_4$ , already considered, and of the half line  $L = \{a + i, 0 < a < \infty\}$ . A simple computation shows that if  $z$  is in  $L$  then  $\text{Im } z > 1, \text{Re } z > 0$ . This accounts for  $A_6$ . For  $A_7$  use the reflection principle. The remaining subsets (i.e.,  $A_5, A_8, A_9$ ) are easily dealt with via the symmetry  $g(H) = (g(-H))^{-1}$ .

*Remark.* It is very important to emphasize that the solution  $[p(t), q(t)]$  given by the theorem does not in general satisfy  $[p(0), q(0)] = [y_0, y_1]$ . It is easily shown that these two 2-dimensional vectors differ by  $O(h)$  terms for fixed, arbitrary  $y_0, y_1$ .

Before we close this paragraph, we describe some of the properties of linear augmented systems.

**THEOREM 2.** *Let  $A$  be as above. (i) If  $\lambda \neq 0$  is an eigenvalue of  $A$  with multiplicity  $m$ , then  $\lambda, -\lambda$  are eigenvalues of the matrix of the augmented system with multiplicity  $m$  each. If  $\mathbf{a}$  is an eigenvector of  $A$  associated to  $\lambda \neq 0$ , then  $[\mathbf{a}, \pm\mathbf{a}]$  are eigenvectors of the augmented matrix associated to  $\pm\lambda$ .*

*(ii) If  $0$  is an eigenvalue of  $A$  with multiplicity  $m$  then  $0$  is an eigenvalue of the augmented matrix with multiplicity  $2m$ . If  $\mathbf{a} \neq \mathbf{0}$  is in the null space of  $A$  then  $[\mathbf{a}, \mathbf{0}], [\mathbf{0}, \mathbf{a}]$  are linearly independent and lie in the null space of the augmented matrix.*

*Proof.* It is easy and will not be given.

It follows from this theorem that the origin is a stable equilibrium of the augmented system if and only if the spectrum of  $A$  is purely imaginary. Theorem 1 implies then that the mid-point rule is absolutely stable for small  $h$  and linear constant-coefficient problems, if and only if these have purely imaginary spectra—a well-known result.

**4. Some auxiliary results.** We now turn our attention to general *nonlinear* augmented systems (2.11). If  $\mathbf{p}_0, \mathbf{q}_0 \in \mathbb{R}^d$  we denote by  $\mathbf{p}(t; \mathbf{p}_0, \mathbf{q}_0), \mathbf{q}(t; \mathbf{p}_0, \mathbf{q}_0)$ , the solution

of (2.11) such that  $\mathbf{p}(0; \mathbf{p}_0, \mathbf{q}_0) = \mathbf{p}_0; \mathbf{q}(0; \mathbf{p}_0, \mathbf{q}_0) = \mathbf{q}_0$ . Then for fixed  $t > 0$  the mapping  $F_t = [\mathbf{p}_0, \mathbf{q}_0] \rightarrow [\mathbf{p}(t; \mathbf{p}_0, \mathbf{q}_0), \mathbf{q}(t; \mathbf{p}_0, \mathbf{q}_0)]$  is the  $t$ -time flow of the system (2.11). It is defined only at those vectors  $[\mathbf{p}_0, \mathbf{q}_0]$  such that the corresponding solution  $[\mathbf{p}(t; \mathbf{p}_0, \mathbf{q}_0), \mathbf{q}(t; \mathbf{p}_0, \mathbf{q}_0)]$  is defined at time  $t$  (i.e. has not reached infinity before that time).

**THEOREM 3.** (i) For each fixed time  $t$ , the  $t$ -time flow  $F_t$  of any augmented system preserves the volume in  $\mathbb{R}^{2d}$ .

(ii) The equilibria of the augmented system (2.11) are precisely the points  $[\mathbf{a}, \mathbf{b}]$ , with  $\mathbf{a}, \mathbf{b}$  equilibria of the original system. At a diagonal equilibrium  $[\mathbf{a}, \mathbf{a}]$  the eigenvalues of the Jacobian matrix of the augmented system are given by  $\pm\lambda$ , with  $\lambda$  an eigenvalue of the Jacobian matrix at  $\mathbf{a}$  of the original system (2.1).

(iii) If the original system is a gradient system i.e.  $\mathbf{f} = \text{grad } V$ , for a scalar function  $V$ , then the augmented system takes the Hamiltonian form

$$\mathbf{p}' = -\frac{\partial H}{\partial \mathbf{q}}, \quad \mathbf{q}' = \frac{\partial H}{\partial \mathbf{p}}$$

for the Hamiltonian function  $H(\mathbf{p}, \mathbf{q}) = V(\mathbf{p}) - V(\mathbf{q})$ . In this case  $H$  is a first integral of the augmented system, i.e.  $H(\mathbf{p}(t), \mathbf{q}(t)) = \text{constant}$  for solutions of (2.11).

(iv) If the original system has a quadratic first integral  $\mathbf{y}(t)^T M \mathbf{y}(t) = \text{constant}$ , with  $M$  a constant, symmetric matrix, then the augmented system has the first integral  $\mathbf{p}(t)^T M \mathbf{q}(t) = \text{constant}$ .

*Proof.* (ii)–(iv) are easy. For (i) note that  $[\mathbf{f}(\mathbf{q}), \mathbf{f}(\mathbf{p})]$  is a divergence-free field in the  $[\mathbf{p}, \mathbf{q}]$  space [2, p. 69].

For the augmented recursion we have, similarly, the next theorem.

**THEOREM 4.** (i) The mapping  $T$  in (2.10) preserves the orientation and the volume.

(ii) The fixed points of  $T$  are of the form  $[\mathbf{a}, \mathbf{b}]$  with  $\mathbf{a}, \mathbf{b}$  equilibria of  $\mathbf{f}$ .

(iii)  $T$  is one-to-one and onto.

(iv) If the original system has a quadratic first integral  $\mathbf{y}(t)^T M \mathbf{y}(t) = \text{constant}$ , with  $M$  a constant symmetric matrix, then the leapfrog points verify  $\mathbf{y}_n^T M \mathbf{y}_{n+1} = \text{constant}$ ,  $n = 0, 1, \dots$ .

*Proof.* Introduce the map

$$S \begin{bmatrix} \mathbf{p} \\ \mathbf{q} \end{bmatrix} = \begin{bmatrix} \mathbf{q} \\ \mathbf{p} + 2h\mathbf{f}(\mathbf{q}) \end{bmatrix},$$

so that  $T$  is the composition  $S \circ S$ . It is clear that  $S$  is one-to-one and onto and this implies (iii). Furthermore the Jacobian matrix of  $S$  takes the form

$$J = \begin{bmatrix} 0 & I \\ I & 2hD\mathbf{f} \end{bmatrix}$$

with  $D\mathbf{f}$  the Jacobian of  $\mathbf{f}(\mathbf{q})$  w.r.t.  $\mathbf{q}$ . Then  $\det(J) = -1$  and the determinant of the Jacobian matrix of  $T$  equals 1. This proves i), ii) and iv) are trivial.

**5. Nonlinear problems.** In this section we prove, by means of examples, that the qualitative behaviour of the augmented sequence (2.8) is governed by the behaviour of the neighbouring solutions of the augmented system, in the sense that when  $[\mathbf{y}_{2n}, \mathbf{y}_{2n+1}]$  has been computed  $[\mathbf{y}_{2n+2}, \mathbf{y}_{2n+3}]$  lies near the corresponding local solution.

A. Our first example concerns the escalar equation  $y' = y^2$ , whose nonequilibrium solutions are monotonically increasing functions of  $t$ . The origin is the only equilibrium. If  $y(0) < 0$ , then  $y(t)$  tends to 0 as  $t$  tends to  $\infty$ . If  $y(0) > 0$ , then  $y(t)$  reaches in finite time.

The problem  $y' = y^2$ ,  $y(0) = -1$ , is solved by the mid-point method with  $h = 0.1$ ,  $y_0 = -1$ , and  $y_1$  obtained by means of Euler's rule. The computed values  $y_n$  do not exhibit the monotonic behaviour of the points  $y(nh)$ . Some values of  $y_n$  are shown in Table 1.

TABLE 1

$n$	$y_n$
40	-0.308
41	-0.062
42	-0.307
43	-0.043
.....	
80	-0.220
81	0.267
82	-0.206
83	0.276
.....	
120	0.141
121	0.319
.....	
150	0.771
151	0.850
.....	
~180	overflow

The augmented system is, according to Theorem 3, Hamiltonian with  $H(p, q) = \frac{1}{3}(p^3 - q^3)$ . In the  $(p, q)$  plane, solutions of the augmented system are contained in the level sets  $p^3 - q^3 = \text{constant}$  (see Fig. 1). When the computed points are plotted in the

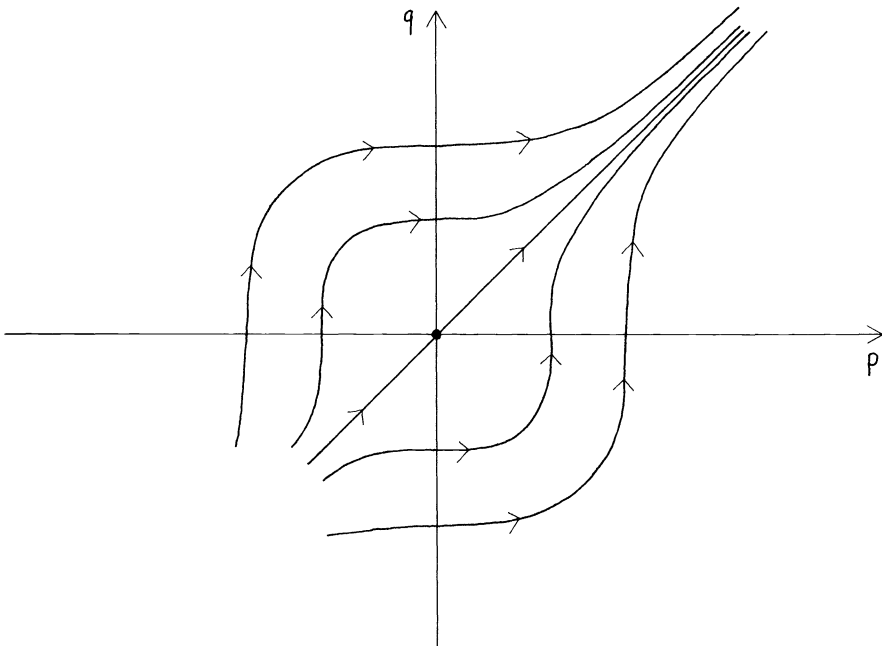


FIG. 1



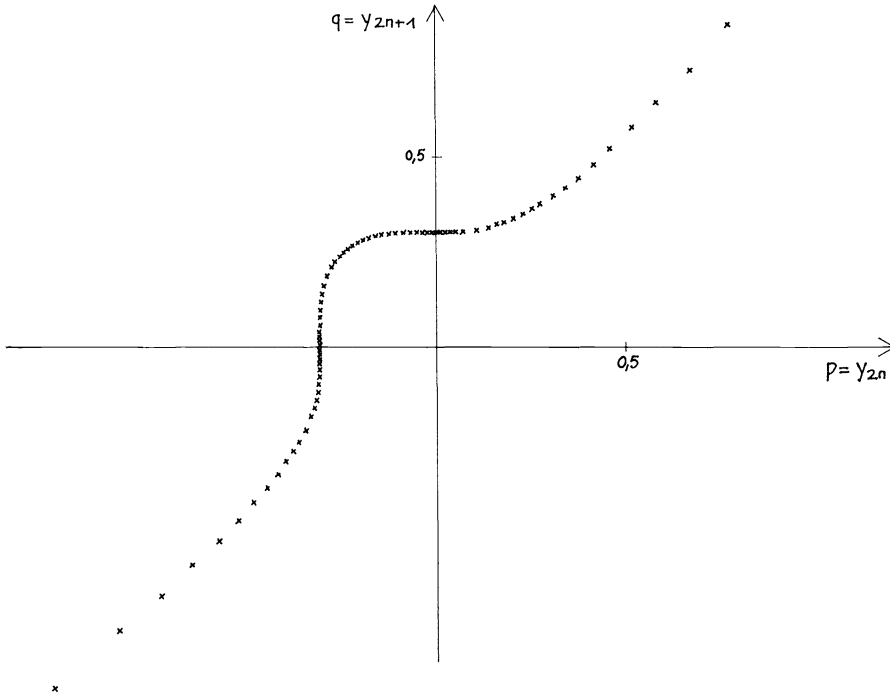


FIG. 2

$(p, q)$  space ( $p = y_{2n}, q = y_{2n+1}$ ) it is clear (Fig. 2) that their behaviour mimics that of solutions of the augmented system.

Note that while the points  $(y_{2n}, y_{2n+1})$  remain near the diagonal, they roughly move towards the origin. In this range of values of  $n$ , the behaviour is that of the original equation  $y' = y^2$  (more precisely that of the diagonal solutions of the augmented system). However the computed points move quickly away from the diagonal and then, as discussed in § 2, the dynamics of the augmented system takes over, leading to a rapid increase of the magnitude of the values  $y_n$ .

The fact that the blowup of leapfrog schemes is preceded by large discrepancies between consecutive values  $y_n, y_{n+1}$  (i.e. nondiagonal behaviour) has been known for a long time [11].

The following remark will be used later: *Any choice of  $y_0, y_1$  leads to a leapfrog sequence with  $\lim y_n = \infty$ .* Strictly speaking there is a curve in the  $(p, q)$ -plane so that if  $(y_0, y_1)$  lies on  $C^-$  then  $(y_{2n}, y_{2n+1})$  lie also on  $C^-$  and  $\lim y_n = 0$ , provided that no round-off is perpetrated.

The curve  $C^-$  plays, in the difference case, the role played by the bisectrix of the third quadrant in the augmented system (Fig. 1). Of course, in practice round-off is always present and so computationally  $\lim y_n = \infty$  for every  $y_0, y_1$ .

The equation  $q = \phi(p)$  for  $C^-$  near the origin is readily obtained by assuming an expansion

$$q = \phi(p) = a_1 p + a_2 p^2 + a_3 p^3 + \dots$$

and imposing the requirement that the coordinates  $T_{(1)}, T_{(2)}$  of the transformed point  $T(p, \phi(p))$  satisfy  $T_{(2)} = \phi(T_{(1)})$ . Thus, we find

$$(5.1) \quad q = \phi(p) = p + hp^2 + h^2p^3 + O(p^5), \quad p \rightarrow 0^-.$$

In order to numerically investigate the role of  $C^-$  the following experiment was carried out.

For  $h = \frac{1}{2}$ , we took  $y_0 = -0.1$  and successively set  $y_1$  equal to  $-0.1, -0.095, -0.09525$ . This corresponds to choosing  $(y_0, y_1)$  on  $C^-$  except for terms  $O(p^2), O(p^3), O(p^5)$  respectively. The smallest value of  $n$  for which  $y_n$  is larger than zero is given in Table 2.

TABLE 2

$y_0$	$y_1$	$N$
-0.1	-0.1	49
-0.1	-0.095	164
-0.1	-0.09525	1,045

It is useful to observe that  $-0.095, -0.09525$  are precisely the values of  $y_1$  furnished by Euler’s rule and the second order Taylor expansion method, respectively.

Returning now to the flow in Fig. 1, it should be pointed out that, due to the divergence-free/area conserving property the behaviour of the contours  $p^3 - q^3 = \text{constant}$  is similar to that of the streamlines in an incompressible flow. Thus these contours are sparsely distributed in the neighbourhood of the equilibrium (stagnation point) and converge near the diagonal of the first quadrant where the magnitude of the velocity of the flow is large. Hence the diagonal  $p = q > 0$  is an attractor of the flow of the augmented system. Analogously for the mapping  $T$ , there exists a curve  $C^+$  in the region  $p, q > 0$  such that if  $(y_0, y_1)$  lies on  $C^+$  then for all integers  $n, T^n(y_0, y_1)$  lies on  $C^+$  with

$$\lim_{n \rightarrow \infty} T^n(y_0, y_1) = (\infty, \infty), \quad \lim_{n \rightarrow \infty} T^{-n}(y_0, y_1) = (0, 0),$$

and  $C^+$  attracts the flow of the discrete recursion defined by  $T$ .

The expression  $q = \psi(p)$  near  $p = 0$  is again given by (5.1). Near  $p = \infty, q = \psi(p)$  has an expansion in powers of  $p^{1/2}$

$$q = 2hp^2 + (2h)^{-1/2}p^{1/2} + O(p^{-1/2}), \quad p \rightarrow \infty.$$

Hence, we expect that for any choice of  $y_0, y_1, h$  and large  $n$

$$y_{2n+1} \approx 2hy_{2n}^2 + (2h)^{-1/2}y_{2n}^{1/2}.$$

In fact, when  $h = 0.1, y_0 = y_1 = -1$ , we find that for  $n = 36, y_{2n} = 157.43, y_{2n+1} = 4984.52$  and the discrepancy between the right- and left-hand sides in the expression above equals  $-0.937$ .

B. We now follow [26], [28] and consider the equation  $y' = y - y^2$ . This has the equilibria  $y = 0$  (unstable) and  $y = 1$  stable. Solutions with  $y(0) > 1$  decrease monotonically towards 1 and solutions with  $0 < y(0) < 1$  increase monotonically towards 1. When  $y(0) < 0$  the solution reaches  $-\infty$  in finite time. (Solutions in closed form are readily available, but quantitative features are disregarded here.)

The problem  $y' = y - y^2, y(0) = 0.5$  was integrated by the mid-point method with  $h = 0.1, y_0 = 0.5$  and  $y_1$  obtained by means of Euler’s rule. Some computed points are shown in Table 3.

Initially the values  $y_n$  grow and approach the equilibrium  $y = 1$ . Then they oscillate with increasing amplitude around that equilibrium. When the oscillations become large enough, the computed points are “attracted” towards the *unstable* equilibrium  $y = 0$ , a surprising behaviour. Later, the  $y_n$  recover the monotonic behaviour and eventually ( $n = 500, 501$ ) a situation very similar to the initial ( $n = 0, 1$ ) is attained.

TABLE 3

$n$	$y_n$
80	.99966
81	.99969
.....	
160	.99988
161	1.00013
.....	
240	.62801
241	1.31896
.....	
320	-.00320
321	.00262
.....	
360	-.00005
361	+.00005
440	.00286
441	.00316
.....	
500	.53534
501	.56013

The strange dynamics of the numerical solution is again readily explained by that of the augmented system. Now, we have  $p' = q - q^2$ ,  $q' = p - p^2$ , with four equilibria  $O = (0, 0)$ ,  $P = (1, 1)$ ,  $C_1 = (0, 1)$ ,  $C_2 = (1, 0)$ . According to Theorem 2, both  $O$  and  $P$  are saddles, while a simple computation shows that  $C_1$  and  $C_2$  are centers.

The augmented system is Hamiltonian with  $H(p, q) = (1/2)(p^2 - q^2) - (1/3)(p^3 - q^3)$ . The contour  $H(p, q) = 0$  consists of the diagonal  $p = q$  and of the ellipse  $3(p + q) = 2(p^2 + q^2 + pq)$ . The latter comprises the unstable manifold of  $P$  and the stable manifold of  $O$ . The phase portrait of the augmented system given in Fig. 3 can now be compared with the plot of the points  $p = y_{2n}$ ,  $q = y_{2n+1}$  given in Fig. 3 ( $n = 0, 5, 10, \dots, 450$ ).

We shall return to this example in § 7.

C. *First integrals.* Our next example concerns the system  $y'_{(1)} = -y_{(1)}y_{(2)}$ ;  $y'_{(2)} = y_{(1)}^2$ . (Bracketed subindices denote components.) The system has the first integral  $y_{(1)}^2 + y_{(2)}^2 = \text{constant}$ , implying that the origin is a stable equilibrium and that the solutions remain bounded as  $t$  increases. The points  $y_{(1)} = 0$ ,  $y_{(2)} = r$  are equilibria, (stable for  $r \geq 0$ , unstable for  $r < 0$ ). In the  $(y_{(1)}, y_{(2)})$ -phase plane a nonequilibrium solution, is represented by a circular arc connecting the initial point  $(y_{(1)}(0), y_{(2)}(0))$  to the equilibrium:

$$y_{(1)} = 0, \quad y_{(2)} = (y_{(1)}^2(0) + y_{(2)}^2(0))^{1/2}.$$

The augmented system

$$\begin{aligned}
 p'_{(1)} &= -q_{(1)}q_{(2)}, \\
 p'_{(2)} &= q_{(1)}^2, \\
 q'_{(1)} &= -p_{(1)}p_{(2)}, \\
 q'_{(2)} &= p_{(1)}^2,
 \end{aligned}
 \tag{5.2}$$

inherits, according to Theorem 3, the first integral

$$p_{(1)}q_{(1)} + p_{(2)}q_{(2)} = \text{constant.} \tag{5.3}$$

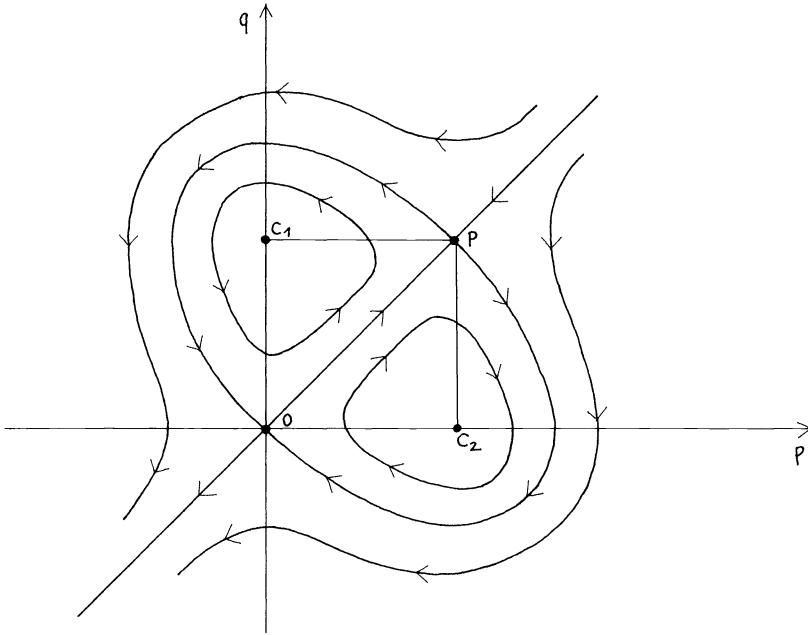


FIG. 3

However the surfaces in the  $(p_{(1)}, p_{(2)}, q_{(1)}, q_{(2)})$ -space whose equation is (5.3) are not bounded: now the first integral does not guarantee boundedness of the solutions. In fact, consider a diagonal point  $\mathbf{p}^0$  with coordinates  $p_{(1)}^0 = q_{(1)}^0 \neq 0, p_{(2)}^0 = q_{(2)}^0$ . From the theorem of continuous dependence on initial values, we conclude that solutions to the augmented system with initial data near  $\mathbf{p}^0$  will reach the neighbourhood of the diagonal equilibrium  $\mathbf{p}^e, p_{(1)}^e = q_{(1)}^e = 0, p_{(2)}^e = q_{(2)}^e = ((p_{(1)}^0)^2 + (p_{(2)}^0)^2)^{1/2}$ . Study of the linearization of this equilibrium shows that, generically, solutions in the neighbourhood of  $\mathbf{p}^e$  leave the regions

$$\{p_{(1)} > 0, p_{(2)} > 0, q_{(1)} > 0, q_{(2)} > 0\},$$

$$\{p_{(1)} > 0, p_{(2)} > 0, q_{(1)} < 0, q_{(2)} < 0\},$$

and enter the regions

$$R_1 = \{p_{(1)} > 0, p_{(2)} > 0, q_{(1)} < 0, q_{(2)} > 0\}$$

and

$$R_2 = \{p_{(1)} < 0, p_{(2)} > 0, q_{(1)} > 0, q_{(2)} > 0\}.$$

Now, it is easily seen that  $R_1, R_2$  are positively invariant for the flow of (5.1), i.e. solutions of (5.1) cannot leave  $R_1$  or  $R_2$ . We conclude that solutions of (5.1) which start near the diagonal, generically enter  $R_1$  or  $R_2$  and remain there. However, it is clear from the signs of the right-hand sides in (5.1) that solutions in  $R_1$  or  $R_2$  have components whose magnitude grow unboundedly. (In fact they reach  $\infty$  in finite time, due to the quadratic interactions.) To sum up, solutions of (5.2) with initial data arbitrarily close to the diagonal will generically reach infinity in finite time.

Once more we found that the augmented system provides a reliable indication as to the dynamics of the mid-point rule solution to the original system. Some computed values ( $h = 0.1, y_{(1)}(0) = 0.2, y_{(2)}(0) = 1.$ ) are shown in Table 4.

TABLE 4

$n$	$\mathcal{Y}_{(1)2n}$	$\mathcal{Y}_{(2)2n}$	$\mathcal{Y}_{(1)2n+1}$	$\mathcal{Y}_{(2)2n+1}$
4	0.090	1.015	0.079	1.017
5	0.075	1.017	0.064	1.018
.....				
15	0.019	1.019	-0.002	1.020
16	0.019	1.019	-0.006	1.021
.....				
40	3.23	3.38	-4.63	4.74
41	7.64	7.68	-16.3	16.4

The dynamics is, for small  $n$ , that of the original system:  $y_{(1)n}$  decreases towards 0 and  $y_{(2)n}$  increases. In the neighbourhood of the equilibrium  $\mathbf{p}^e$  the points enter the region  $R_1$  and once there the behaviour is monotonic towards  $(\infty, \infty, -\infty, \infty)$ .

As noted before, Theorem 4 (iv) ensures the conservation property (5.3). Therefore an unbounded growth of the solution  $y_n$  can only take place if one of the products  $\mathcal{Y}_{(1)n}\mathcal{Y}_{(1)n+1}$  or  $\mathcal{Y}_{(2)n}\mathcal{Y}_{(2)n+1}$  is negative.

**6. Application to partial differential equations.** We now turn our attention to the study of schemes for the numerical solution of evolutionary partial differential equations based on the leapfrog time stepping. It is supposed that the space variables are discretized first [20] (by means of finite differences, finite elements etc. . .) so as to approximate the original equation or system by a system of ordinary differential equations having the time as independent variable. The resulting system is in turn discretized in time by means of the leapfrog technique to obtain the fully discrete set of approximating equations.

For simplicity we restrict ourselves to the model equation [5]:

$$u_t + uu_x = 0, \quad u(0, t) = u(1, t),$$

although our study can be readily extended among others to problems associated with the vorticity/stream-function formulation of the equations of inviscid, incompressible flow.

Let  $r$  be a real parameter. The interval  $[0, 1]$  is divided into  $J$  intervals of equal length  $\Delta x = 1/J$ , by means of a grid  $x_j = j \Delta x, j = 0, 1, \dots, J$  and  $u(x_j, t)$  is approximated by the solutions  $U_j = U_j(t)$  of the system [5]:

$$(6.1) \quad U_j' + (r/2\Delta x)U_j(U_{j+1} - U_{j-1}) + ((1-r)/4\Delta x)(U_{j+1}^2 - U_{j-1}^2) = 0, \\ j = 1, \dots, J-1, \quad U_J = U_0.$$

Error estimates for  $U_j(t)$  are given in [9]. The dynamics of the solutions of (6.1) and of their leapfrog approximations is complicated indeed. A means of achieving some insight is to restrict the attention to  $J$ -dimensional vectors  $[U_0, \dots, U_{J-1}]$  representable as a superposition of a *small* number of *discrete Fourier modes*. The relevant set of modes must be chosen in such a way that it is closed under the quadratic interactions in (6.1), i.e. the product of two modes in the set must belong to the set. This simplifying approach was first taken by N. A. Phillips [17] for the vorticity transport equation. Later references include Richtmyer (reproduced in [18]) and Fornberg [5].

Here two sets of modes are considered:

- (i) *One mode solution.* Assume that  $J$  is a multiple of 3.

Then [5] the system (6.1) has solutions

$$(6.2) \quad U_j(t) = a(t) \sin(2\pi j/3),$$

where the amplitude  $a(t)$  satisfies

$$(6.3) \quad a'(t) = ca^2(t), \quad c = \left(\frac{\sqrt{3}}{8\Delta x}\right)(1-3r).$$

Note that leapfrog discretization of the amplitude equation (6.3) yields, via (6.2), a solution to the fully discrete leapfrog partial difference equations, i.e. Fourier analysis and leapfrog differencing commute.

Fornberg [5] observed that if  $r \neq 1/3$  and  $a(0)$  and  $c$  are of the same sign, then  $a(t)$  reaches infinity at a finite  $O(1/\Delta x)$  time. Hence he concluded that the corresponding leapfrog approximations would grow unboundedly for increasing  $n$ . The condition on the sign of  $a(0)$  means that, for a fixed value of  $r$ , only one among the sign patterns

$$\begin{aligned} &0, +, -, 0, +, -, 0, +, -, \dots, 0, +, -, \\ &0, -, +, 0, -, +, 0, -, +, \dots, 0, -, + \end{aligned}$$

in  $U_0(0), U_1(0), \dots, U_{J-1}(0)$  will lead to the blow up predicted by Fornberg. Upon introducing the new variable  $y = ca$ , the equation (6.3) reads  $y' = y^2$ , so that from our analysis in the previous section we conclude that the blowup in the leapfrog solution will take place (if  $r \neq 1/3$ ) either if the signs of  $a(0)$  and  $c$  agree or not. In other words both sign patterns in  $U_j(0), j = 0, 1, \dots, J-1$  lead to blowup.

(ii) *Two modes solution.* In the case  $r \neq 1/3$  the instability described above is attributable to the space discretization.

Any method employed for the integration in time inherits that offending behaviour and this is particularly so with the mid-point rule, which, as we have seen, enlarges the class of initial conditions leading to blowup.

The value  $r = 1/3$  is special in that it ensures the conservation law  $\sum U_j^2(t) = \text{constant}$ , and therefore forces the boundedness of  $U_j(t), j = 1, \dots, J, t > 0$ . (See Morton [14] or Kreiss and Olinger [8] for a discussion.) Hereafter we set  $r = 1/3$ , assume that  $J$  is multiple of four and look for solutions

$$U_j(t) = a(t)(\sin(\pi j/2) + \cos(\pi j/2)) + b(t) \cos \pi j.$$

The amplitudes  $a, b$  must satisfy

$$a' = \left(\frac{1}{3\Delta x}\right)ab, \quad b' = -\left(\frac{1}{3\Delta x}\right)a^2,$$

a system whose leapfrog discretization entails blowup as shown in the previous section.

The material in this section is covered in detail and expanded in [27].

**7. Integrability. Concluding remarks.** We now return to Example B in § 5. It is important to point out that the similarity between Figs. 3 and 4 is deceptive. In fact the augmented system is an *integrable* Hamiltonian system [2], [25]. This simply means that due to the first integral  $H(p(t), q(t)) = \text{constant}$ , the integration of the system is achievable by quadratures. (In systems with  $d$  degrees of freedom,  $d$  involutive first integrals are required.) For the time  $h$  flow  $F_h$ , integrability implies that all the iterates  $F_h^n(p_0, q_0)$  of a point lie on a curve  $H(p, q) = \text{constant}$ . These curves are closed in the region  $H(p, q) < 0$ . It may be conjectured from Fig. 4, that the mapping  $T$ , (which approximates  $F_h$ ) also defines an integrable system, i.e. a nonconstant function  $S(p, q)$  defined in all  $R^2$  exists so that the iterates  $T^n(p_0, q_0)$  are confined to lie on a level

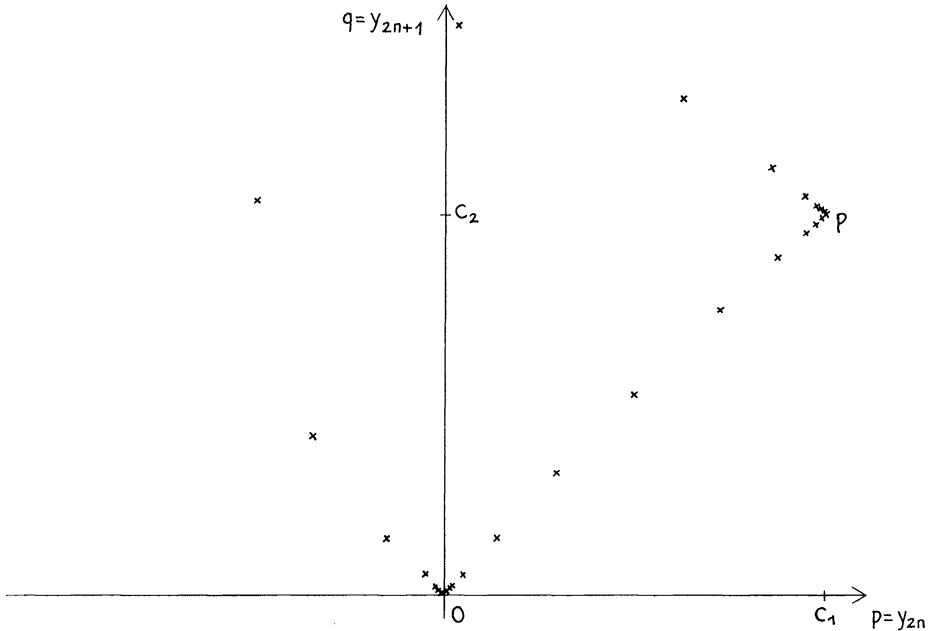


FIG. 4

curve  $S(p, q) = \text{constant}$ . However Ushiki [26] has proved that this is not the case. In fact he shows that the dynamics of  $T$  is *chaotic* in a sense made precise in his paper. This remark does not invalidate our claim that  $(y_{2n+2}, y_{2n+3})$  lies near the local solution through  $(y_{2n}, y_{2n+1})$ ; it only implies that the behaviour of the sequence  $(y_{2n}, y_{2n+1})$ ,  $n = 0, 1, \dots$  may be far more complicated than that of sequences  $(p(nh), q(nh))$ ,  $n = 0, 1, \dots$ . In this regard linear systems are exceptional as shown by Theorem 1.

We would like to recall that there are many available results concerning the behaviour of area preserving mappings [24]. Some of them could be used in order to ascertain the properties of the dynamics of  $T$ . It is expected that such a dynamics will be highly involved. Some of the finer details will be numerically missed due to round-off. Of particular significance is the study of  $T$  near center equilibria, since, as noted before, these are the only nondegenerate equilibria in whose neighbourhood the leapfrog technique is useful. This point will be the subject of a forthcoming paper.

Finally, we observe that theories similar to the one presented in this paper can be constructed for any multistep method having  $r^k - r^{k-2}$  as first characteristic polynomial. (Nystrom and generalized Milne-Simpson methods, in the terminology of [10].) For instance the dynamics associated with Milne's method

$$y_{n+2} - y_n = \frac{h}{3} (f_{n+2} + 4f_{n+1} + f_n),$$

would be governed by the enlarged system

$$p' = \frac{1}{3}(p + 2q), \quad q' = \frac{1}{3}(2p + q).$$

*Note.* Prof. M. N. Spijker has recently let us know that the idea of augmented system had been considered by H. J. Stetter, *Symmetric two-step algorithms for ordinary differential equations*, Computing, 5 (1970), pp. 267-280. However the subject of Stetter's paper is completely different from ours.

## REFERENCES

- [1] A. ARAKAWA, *Computational design for long-term numerical integration of the equations of fluid motion: two-dimensional incompressible flow. Part 1*, J. Comput. Phys., 1 (1966), pp. 119–143.
- [2] V. ARNOLD, *Mathematical Methods of Classical Mechanics*, Springer, New York, 1978.
- [3] ———, *Chapitres supplémentaires de la théorie des équations différentielles ordinaires*, MIR, Moscow, 1980.
- [4] K. DEKKER AND J. G. VERWER, *Stability of Runge-Kutta Methods for Stiff Nonlinear Differential Equations*, North-Holland, Amsterdam, 1984.
- [5] B. FORNBERG, *On the instability of leapfrog and Crank-Nicolson approximations of a nonlinear partial differential equation*, Math. Comput., 27 (1973), pp. 47–57.
- [6] P. HENRICI, *Discrete Variable Methods in Ordinary Differential Equations*, John Wiley, New York, 1962.
- [7] M. W. HIRSCH AND S. SMALE, *Differential Equations, Dynamical Systems and Linear Algebra*, Academic Press, New York, 1974.
- [8] H. O. KREISS AND J. OLIGER, *Methods for the Approximate Solution of Time Dependent Problems*, GARP Publication Series, 10, 1973.
- [9] P.-Y. KUO AND J. M. SANZ-SERNA, *Convergence of methods for the numerical solution of the Korteweg-de Vries equation*, IMA J. Numer. Anal., 1 (1981), pp. 215–221.
- [10] J. D. LAMBERT, *Computational Methods in Ordinary Differential Equations*, John Wiley, London, 1973.
- [11] D. K. LILLY, *On the computational stability of numerical solutions of time-dependent nonlinear geophysical fluid dynamics problems*, Monthly Wealth Rev., 93 (1965), pp. 11–26.
- [12] A. MAJDA AND S. OSHER, *A systematic approach for correcting nonlinear instabilities, the Lax-Wendroff scheme for scalar conservation laws*, Numer. Math., 30 (1978), pp. 429–452.
- [13] K. MIYAKODA, *Contribution to the numerical weather prediction computation with finite difference*, Japan J. Geophys., 3 (1962), pp. 75–190.
- [14] K. W. MORTON, *Initial value problems by finite difference and other methods*, in *The State of The Art in Numerical Analysis*, P. A. H. Jacobs, ed., Academic Press, New York, 1977, pp. 699–756.
- [15] A. C. NEWELL, *Finite amplitude instabilities of partial difference equations*, SIAM J. Appl. Math., 33 (1977), pp. 133–160.
- [16] C. PALENCIA AND J. M. SANZ-SERNA, *Equivalence theorems for incomplete spaces: an appraisal*, IMA J. Numer. Anal., 4 (1984), pp. 109–115.
- [17] N. A. PHILLIPS, *An example of nonlinear computational stability*, *The Atmosphere and the Sea in Motion*, B. Bolin ed., Rockefeller Institute, New York, 1959, pp. 501–504.
- [18] R. D. RICHTMYER AND K. W. MORTON, *Difference Methods for Initial-Value Problems*, Wiley Interscience, New York, 1967.
- [19] J. M. SANZ-SERNA, *An explicit finite-difference scheme with exact conservation properties*, J. Comput. Phys., 47 (1982), pp. 199–210.
- [20] ———, *On finite elements simultaneously in space and time*, Int. J. Numer. Meth. Engrg., 19 (1983), pp. 623–624.
- [21] ———, *Convergence of the Lambert-McLeod trajectory solver and of the CELF method*, Numer. Math., to appear.
- [22] J. M. SANZ-SERNA AND V. S. MANORANJAN, *A method for the integration in time of certain partial differential equations*, J. Comput. Phys., 52 (1983), pp. 273–289.
- [23] L. F. SHAMPINE AND M. K. GORDON, *Computer Solution of Ordinary Differential Equations, The Initial Value Problem*, W. H. Freeman, San Francisco, 1975.
- [24] C. L. SIEGEL AND J. K. MOSER, *Lectures on Celestial Mechanics*, Springer, Berlin, 1971.
- [25] C. SIMÓ, *Integrability: a difficult analytical problem*, *Publicacions de la Secció de Matemàtiques, Universitat Autònoma de Barcelona*, 22 (1980), pp. 71–80.
- [26] S. USHIKI, *Central difference scheme and chaos*, Physica, 4D (1982), pp. 407–424.
- [27] F. VADILLO, *Estabilidad no lineal en ecuaciones en derivadas parciales*, PhD. thesis, Universidad de Valladolid, to appear.
- [28] M. YAMAGUTI AND S. USHIKI, *Chaos in numerical analysis of ordinary differential equations*, Physica, 3D (1981), pp. 618–626.



## DESCRIPTION AND EVALUATION OF A STIFF ODE CODE DSTIFF\*

GOPAL K. GUPTA†

**Abstract.** The paper describes and evaluates DSTIFF, a set of subroutines for solving stiff ordinary differential equations. The code is somewhat similar to the well-known packages LSODE, GEAR and DIFSUB but the present set of subroutines are based on least squares multistep formulas rather than the BDF. The paper describes the formulas used in the code, the structure of the code and the heuristics used, and evaluates its performance. The code seems to be much more efficient than LSODE in solving stiff equations which have Jacobians with eigenvalues having large imaginary parts.

On other problems, DSTIFF is as efficient as LSODE on larger tolerances and somewhat less efficient than LSODE on stringent tolerances.

**Key words.** ordinary differential equations, stiff equations, mathematical software, multistep methods

**1. Introduction.** Several codes are now available for solving stiff ordinary differential equations. Enright, Hull and Lindberg (1975) and Enright and Hull (1976) have tested codes for solving stiff equations and Shampine and Gear (1979) discuss some of the available codes. Shampine and Gear also discuss why there is a need to distinguish a special class of problems termed stiff and describe the common characteristics of methods used for solving stiff equations. For a more recent survey of codes and techniques used in them, we refer the reader to Gupta, Sacks-Davis and Tischer (1983).

Most of the codes available for solving stiff equations may be divided into the following three classes:

(a) *The BDF codes.* The first code using the Backward Differentiation Formulas (BDF) was designed by Gear (1971). This code, DIFSUB, uses formulas up to order 6. Several attempts have been made to improve this code and several variants of the code are now available. The best known variants are GEAR by Hindmarsh (1974) and EPISODE by Byrne and Hindmarsh (1975). GEAR and EPISODE themselves have several versions. For example a version is available for solving problems with banded Jacobians and another for linearly implicit equations of the form  $Ay' = g$ ,  $A$  a square matrix. A new version of GEAR called LSODE is now available from Hindmarsh (1980). Several other variants are known to exist. Test results for the codes cited above indicate that the BDF codes are the most efficient for solving stiff equations except when the eigenvalues of the Jacobian of the differential equations have large imaginary parts. For such oscillatory problems, the BDF codes become very inefficient because of the poor stability properties of the 5th and 6th order BDF. To partly overcome this problem, several BDF codes restrict the maximum order of the formula used to 5.

(b) *The second-derivative codes.* To overcome the stability problems of the BDF, Enright (1972) suggested the use of the second-derivative formulas (SDF); linear multistep formulas which include second-derivative terms. Enright designed the first SDF code, SDBASIC, using formulas of orders up to 9. Other SDF codes have been designed by Addison (1979) and Sacks-Davis (1980). Although the SDF used in the above codes have much better stability than the BDF and therefore overcome the problem of the BDF codes, they have three serious disadvantages. These are: requirement of analytic Jacobians, necessity of more linear algebra at each step than in the BDF codes, and the difficulty in solving some problems with large coupling between the equations. The last problem arises because the corrector iterations used in the SDF codes assume that  $\partial^2 f / \partial^2 y$  terms for the given differential equations are negligible.

\* Received by the editors November 13, 1979, and in final revised form July 10, 1984.

† Department of Computer Science, Monash University, Clayton, Victoria 3168, Australia.

Testing of the SDF codes indicates that they are not really very efficient when compared to the BDF codes like GEAR except for solving small systems at stringent tolerances or for solving problems with Jacobians having large imaginary parts for which the BDF codes become very inefficient. As noted above, the biggest problem in using the SDF codes is that the analytic Jacobian must be provided. However, if the Jacobian is available, the SDF codes can also solve nonstiff equations efficiently because the truncation error coefficients of the SDF of Enright (1972) are quite small.

(c) *The Runge-Kutta codes.* Implicit Runge-Kutta formulas (RKF) suitable for solving stiff equations have been derived and some codes based on implicit RKF were tested by Enright et al. (1975) and Enright and Hull (1976). The amount of linear algebra required at each step in using a fully implicit RKF is so large, that these formulas cannot be expected to be efficient even for small systems of equations. Progress has recently been made in reducing the amount of linear algebra involved in using the RKF and a code, STRIDE, has been designed by Butcher, Burrage and Chipman (1979) based on singly-implicit RKF of Burrage (1978). The formulas used in STRIDE are  $A(\alpha)$ -stable,  $\alpha \cong 83^\circ$  (Widlund, 1967). To solve a system of  $N$  equations, STRIDE needs to do about the same amount of  $O(N^3)$  operations per step as a code based on the BDF but several additional  $O(N^2)$  operations per step must be performed. Kaps and Rentrop (1979) take a different approach in designing two codes GRK4T and GRK4A based on embedded generalized RKF (also called the Rosenbrock methods) of order 4. The formula used in GRK4A is A-stable while that in GRK4T is  $A(89.3^\circ)$ -stable. These codes evaluate the Jacobian and then solve a linear system of order  $N$  with four right-hand sides at each step. Preliminary testing by Addison (1980) and results presented by Kaps and Rentrop (1979) indicate that the codes are reliable and efficient in solving problems whose Jacobians have eigenvalues near the imaginary axis. We therefore expect these codes to be useful in solving small systems of stiff equations but the additional linear algebra involved at each step would make them inefficient for large systems.

In addition to the above, a code based on cyclic formulas of orders up to 7, STINT, has been designed by Tendler, Bickart and Picel (1978). Another code, TRAPEX, based on extrapolation was tested by Enright et al. (1975) and Enright and Hull (1976). Also Skeel and Kong (1977) have shown that blended formulas using the Adams formulas and the BDF together are suitable for solving stiff equations. The numerical testing has shown that TRAPEX is not an efficient code. Tendler, Bickart and Picel (1978) show that STINT is reliable but not as efficient as GEAR. Only a few test results of a blended formulas code are available. The blended formulas approach does overcome one of the problems of the SDF codes in that it does not require analytic Jacobians. Blended formulas however still need somewhat more linear algebra per step than the BDF codes.

The present paper describes a code-based on linear multistep formulas derived using least squares. The code, DSTIFF, overcomes the weakness of the BDF codes in solving problems that have decaying highly oscillatory solution components and is almost as efficient and reliable as the BDF codes on most other problems.

**2. The formulas used.** We have noted that the BDF codes like GEAR and LSODE are very efficient for solving many stiff problems but have trouble on systems that have Jacobian with eigenvalues close to the imaginary axis. This difficulty arises because of the poor stability of the BDF of orders 5 and 6 near the imaginary axis. To overcome this Wallace and Gupta (1973), Gupta and Wallace (1975) and Gupta (1975), (1976) have derived linear multistep formulas having much better stability than the BDF. In

fact some formulas presented in Gupta (1975), (1976) are very close to being A-stable even at orders 6 and 7. But these formulas with good stability are not very efficient for solving stiff equations because the formulas have large truncation error coefficients and tend to have roots close to unity at  $h\lambda = \infty$ . This is in contrast to the BDF which for orders 5 and 6 have poor stability near the imaginary axis, small truncation error coefficients and zero roots at  $h\lambda = \infty$ . There are, of course, a large number of formulas somewhere between the two extremes and we have derived several sets in Gupta (1975). We have chosen a set based on least squares to be included in DSTIFF. This set is similar to Adams-Moulton formulas (AMF) but instead of interpolation we use the least squares approximation. To clarify this, we first define the predictor corrector method as

$$P_n(x) = P_{n-1}(x) + \delta_n C((x - x_n)/h)$$

where we want to compute a polynomial  $P_n(x)$  of degree  $m$  at  $x_n$  so that  $y_n = P_n(x_n)$  may be computed.  $P_{n-1}(x)$  is the polynomial approximation at  $x_{n-1}$  and  $C$  is a fixed polynomial of degree  $m$  characteristic of the particular multistep formula. For the AMF of order  $m$ ,  $C$  is such that

$$C(-1) = 0, \quad C'(-k) = 0, \quad k = 1, 2, \dots, m-1.$$

For the formulas we use for solving stiff equations, we require that  $C$  of order  $m$  satisfy the condition  $C(-1) = 0$  and  $C'$  be a least squares approximation to points  $(0, 1), (-1, 0), (-2, 0), \dots, (-N, 0)$  where  $N \geq m-1$  and is so chosen that the formula is stiffly stable and has small truncation error. These formulas were labeled FLS in Gupta and Wallace (1975). We retain this name for them although the values of  $N$  used for some formulas in the present set are different than those in that paper.

In Table 1, we present some of the characteristics of the formulas up to order 10. Higher order formulas have been derived but their inclusion in the code did not seem to affect the subroutine's performance very much. We will show in the last section that it may be desirable to further restrict the maximum order used to 7.

TABLE 1  
Truncation error coefficients and stability of formulas FLS.

Order	$\alpha$	$D$	$k_{m+1}$	Modulus of largest root at $h\lambda = \infty$	Value of $N$
1	90.00	0.0	0.500	0.0	—
2	90.00	0.0	0.083	1.0	1
3	86.46	-0.075	0.242	0.32	3
4	80.13	-0.282	0.374	0.43	5
5	73.58	-0.606	0.529	0.567	7
6	67.77	-1.218	0.724	0.878	9
7	65.53	-1.376	1.886	0.898	12
8	64.96	-1.149	7.686	0.790	16
9	62.78	-2.086	16.737	0.989	19
10	63.74	-1.223	133.955	0.878	26

In Table 1,  $\alpha$  is the angle defined by  $A(\alpha)$ -stability (Widlund (1967)).  $D$  is the most negative  $\text{Re}(h\lambda)$  value on the stability curve.  $k_{m+1}$  is the truncation error coefficient, and  $N$  is the parameter used in the least squares approximation above.

**3. Order and step-size changing.** It is assumed that the reader is familiar with the Nordsieck representation of the predictor-corrector methods. At any point  $x_n$ , let  $a_n$  be the vector of scaled derivatives of the approximating polynomial  $P_n(x)$  when a step-size of  $h_n$  is being used. We have (for example, refer to Gear (1971, p. 216))

$$a_{n+1} = Aa_n + lw$$

where  $A$  is a pascal triangle,  $l$  is a corrector vector depending on the multistep formula being used and  $w$  is the correction necessary.

In DIFSUB, Gear (1971, § 9.3) uses the following techniques in step-size and order changing. Let the present order be  $q$  and the present step-size be  $h_n$ .

(i)  $a_n$  has  $q + 1$  elements. The last element  $a_{n,q+1} \simeq h_n^q y^{(q)} / q!$  Therefore the local truncation error, which is equal to  $k_{q+1} h_n^{q+1} y^{(q+1)} + O(h_n^{q+2})$ , can be estimated by

$$k_{q+1} \cdot q! \nabla a_{n,q+1}$$

( $\nabla a_{n,q+1}$  being the correction applied to the last element). Similarly, the estimate of the local truncation error for order  $q - 1$  is given by

$$k_q \cdot a_{n,q+1} \cdot q!$$

and at order  $q + 1$  by

$$k_{q+1} \cdot q! \cdot (\nabla^2 a_{n,q+1}).$$

Let us call these three estimates  $E_q, E_{q-1}$  and  $E_{q+1}$  respectively.

(ii) Let  $h_{n+1} = \alpha_k h_n$  where  $k = q - 1, q, q + 1$ . New step-sizes for orders  $q, q - 1$  and  $q + 1$  are now calculated. If the user specified error tolerance is  $\epsilon$ , the ratios,  $\alpha_k$ , are

$$\alpha_q = \frac{1}{1.2} \left[ \frac{\epsilon}{\|E_q/w\|} \right]^{1/(q+1)},$$

$$\alpha_{q-1} = \frac{1}{1.3} \left[ \frac{\epsilon}{\|E_{q-1}/w\|} \right]^{1/q},$$

$$\alpha_{q+1} = \frac{1}{1.4} \left[ \frac{\epsilon}{\|E_{q+1}/w\|} \right]^{1/(q+2)},$$

where for each member of the system there is a weight  $w$  and  $\| \cdot \|$  is the  $L_2$ -norm. The order to be used on the next step is selected corresponding to the largest of these  $\alpha$ 's.

(iii) The following heuristics are also used.

- (a) If the largest  $\alpha$  value is less than 1.1, the step-size is not increased.
- (b) The step-size is not changed for  $q + 1$  steps after the last change except if a step fails.
- (c) The order can not be increased if a step fails.
- (d) If the step-size is not increased after computing  $\alpha$ 's, the step-size and the order is not changed for 10 steps except if a step fails.
- (e) If a step fails three times, the order is set to one.
- (f) If the corrector iterations do not converge then the step-size is reduced by a factor of 4.

In the main integrator of DSTIFF  $E_q, E_{q-1}$  and  $E_{q+1}$  are computed as above but the  $\alpha$ 's are computed as follows:

(i) If  $\epsilon > E_q$ , that is, after a successful step we have

$$\alpha_q = \frac{1}{1.05} \left[ \frac{\epsilon_q}{\|E_q/w\|} \right]^{1/(q+1)},$$

$$\alpha_{q-1} = \frac{1}{1.05} \left[ \frac{\varepsilon_{q-1}}{\|E_{q-1}/w\|} \right]^{1/q},$$

$$\alpha_{q+1} = \frac{1}{1.05} \left[ \frac{\varepsilon_{q-2}}{\|E_{q+1}/w\|} \right]^{1/(q+2)},$$

where  $\varepsilon_k = \varepsilon/k^{1/2}$ ,  $k = q-1, q, q+1$ .

Instead of the  $L_2$ -norm, we use the  $L_\infty$ -norm. The order is then selected corresponding to the largest  $\alpha$ . The main reason for changing the heuristics is that we want to encourage the code to go to as high an order as possible. In the BDF codes, order 5 or 6 is reached quickly because the truncation error coefficients of the higher order formulas are smaller. For the FLS formulas used in DSTIFF, the higher order formulas have larger truncation error coefficients as shown in Table 1.

(ii) If  $\varepsilon < E_q$ , then the unsuccessful step must be repeated and the new step-size is based on

$$\alpha_q = \frac{1}{1.05} \left[ \frac{\varepsilon_q}{\|E_q/w\|} \right]^{2/(q+1)},$$

$$\alpha_{q-1} = \frac{1}{1.05} \left[ \frac{\varepsilon_{q-1}}{\|E_1/w\|} \right]^{2/q}.$$

This change is desirable because the need for reducing the step-size indicates that the polynomials approximating the solutions are not very good approximations at the present point. Therefore the step-size needs to be reduced by a larger factor than would be reduced by formulas in (i) above.

(iii) The following heuristics are used:

- (a) If the largest  $\alpha$  value is less than 1.025 then the step-size is not increased.
- (b) The step-size is not changed for  $q+1$  steps after the last change except if a step fails.
- (c) The order cannot be increased if a step fails.
- (d) If a step fails twice and the order being used is more than 3, the order is reduced by one.
- (e) If a step fails three times, then the order is set to one.
- (f) If the corrector iterations do not converge then the step-size is reduced by a factor of 2 and the step-size and the order is not changed for the next 10 steps (except if a step fails).
- (g) DIFSUB, GEAR and LSODE use the fixed-step interpolation when the step-size is varied. This means that the coefficients of the formulas do not change when the step-size is changed. In DSTIFF also we use the fixed-step interpolation most of the time, but we use a technique called the "average-step interpolation" discussed by Gupta and Wallace (1979) whenever the step-size is reduced. New coefficients of the formulas are computed for  $(q+2)/3$  steps after a step-size reduction takes place.

**4. The corrector iterations.** To compute the correction necessary so that the approximating polynomial at the new point  $x_{n+1}$  satisfies the differential equation, an iterative procedure must be used. For stiff equations, the simple iterations do not converge and the modified Newton's iterations must be used. Using the Nordsieck notation, the iterations may be expressed as follows (Gear (1971, p. 207)):

$$a_{n+1,(m+1)} = a_{n+1,(m)} - l[-I + hl_0J]^{-1}F(a_{n+1,(m)}).$$

$a_{n+1,(m)}$  is the  $m$ th approximation to the vector  $a_{n+1}$ .  $l$  is the corrector vector with first

coefficient  $l_0$ .  $J$  is the Jacobian, and  $F$  is a function such that for a vector  $a$ ,  $F(a) = hf(a_0) - a_1 \cdot F(a) = 0$ , if the vector  $a$  satisfies the differential equation.

As is common with other stiff codes, DSTIFF evaluates  $J$  only when necessary, and the matrix  $-I + hl_0J$  is stored as its  $LU$ -decomposition. The present version of DSTIFF does not have any sparse facilities to solve large problems efficiently.

An important part of the corrector iterations technique is the convergence test used in terminating the iterations. In DIFSUB, the iterations are terminated if the following test is satisfied:

$$\left\| \frac{y_{n+1,(m+1)} - y_{n+1,(m)}}{w} \right\| \leq \frac{\epsilon l_0}{2N(q+2)}.$$

$w$  is a weight and the  $L_\infty$ -norm is used.  $\epsilon$  is the user-specified error tolerance and  $N$  is the number of equations. The constant  $l_0/2N(q+2)$  is empirical and used to ensure that the implicit equations are solved somewhat more accurately than  $\epsilon$ .

Hindmarsh (1974) uses a slightly different convergence test in GEAR in an attempt to reduce the number of iterations. The test is based on the assumption that corrector iterations converge linearly and the rate of convergence is almost constant. Let

$$d_m = \frac{y_{n+1,(m+1)} - y_{n+1,(m)}}{w},$$

$$c_m = \frac{d_m}{d_{m-1}}.$$

$w$  is a weight depending on the error criterion and the  $L_2$ -norm is used in computing  $d_m$ .  $c_m$  is the rate of convergence. The following convergence test is used:

$$2c_m d_m \leq \frac{\epsilon l_0}{2(q+2)}.$$

$2c_m d_m$  is an estimate of the correction at the next iteration. The rate of convergence,  $c_m$ , is not available at the first iteration of each new step and the last value of  $c_m$  from the previous step is used. DSTIFF also uses this test of corrector convergence. In LSODE, the convergence test has been modified further.  $d_m$  is computed as above but the RMS-norm is used instead of the  $L_2$ -norm. The convergence test seems to be the following:

$$1.5c_m (d_m q! k_{m+1} l_m) \leq \frac{\epsilon}{2(q+2)},$$

where  $k_{m+1}$  is the truncation error coefficient of the formula being used and  $l_m$  is the last coefficient of the formula vector  $l$ . In addition, if the rate of convergence  $c_m$  becomes greater than 2 on the second or the third iteration, the code assumes that the iterations are diverging.

Recently Shampine (1979), (1980) has discussed the problem of testing for corrector convergence and pointed out some weaknesses of the tests discussed above.

**5. The structure of DSTIFF.** DSTIFF consists of eight subroutines, and the user provides two subroutines and the calling program as discussed in Gupta (1979). Out of the eight subroutines in DSTIFF, five subroutines (SDRIVE, DEC, SOL, PSET, INTERP) are very similar to those used by Hindmarsh and Byrne in GEAR and EPISODE. The remaining three subroutines (SSTIFF, METHOD, NEWHQ) are quite different. We discuss the subroutines briefly.

(1) SDRIVE is the driver subroutine for the main integrator SSTIFF. SDRIVE checks the parameters passed to it and calls SSTIFF repeatedly till the integration reaches the end-point specified by the user. The calling sequence of DSTIFF is the same as EPISODE and GEAR.

(2) DEC and SOL are subroutines for solving linear equations.

(3) SSTIFF is the main integrator. It takes one step and returns control to the calling routine. On the first call, the subroutine sets the order to one and computes a suitable step-size. At the end of one step of integration, the subroutine computes, if necessary, new step-size and order to be used for the next step.

(4) METHOD specifies the coefficients of the formulas being used and the truncation error coefficients of the formulas. The three-dimensional array PERTST used in DIFSUB and GEAR has not been used so that the subroutine is easier to understand.

(5) NEWHQ is called by SSTIFF when new step-size and order needs to be computed. Separating the step-size and order changing techniques into a subroutine makes it easier to understand and modify the package.

(6) PSET is called by SSTIFF to process the Jacobian. PSET numerically evaluates the Jacobian if necessary and calls DEC to obtain a  $LU$  decomposition of the linear equations.

(7) INTERP is an interpolation routine used when the user wants the solution at intermediate points.

The overall structure of DSTIFF, therefore, looks like Fig. 1.

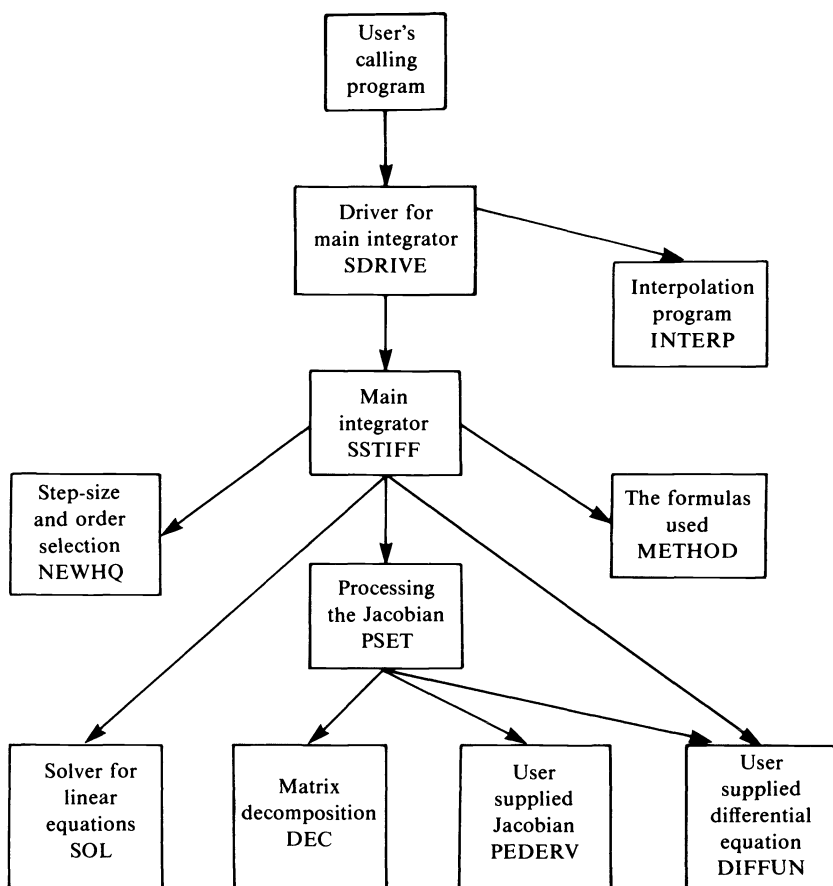


FIG. 1

**6. Performance evaluation of DSTIFF.** Enright, Hull and Lindberg (1975), Enright and Hull (1976), and Addison (1981) have tested several subroutines for solving stiff equations. As we have noted, the BDF codes have been found to be the most efficient. We therefore compare the performance of DSTIFF with LSODE, the most recent BDF code. Tables 2 through 6 compare the performance of LSODE and DSTIFF in solving the test problems in the testing package described in Enright (1979). The testing package New Detest differs from the package used in Enright et al. (1975) in that the code being tested does not need to be modified and normalised statistics, if required, are provided. New Detest includes a set of 30 test problems consisting of 25 test problems in five classes *A*, *B*, *C*, *D* and *E* used in Enright et al. (1975) and 5 additional chemical kinetics problems in class *F*. We have presented the overall results for each of the six problem classes. (New Detest was run on an IBM 3031/6 at the Asian Institute of Technology, Bangkok.) Absolute error criterion was used in the testing.

TABLE 2  
*The total number of function evaluations (overall results of each problem class).*

Problem class		A	B	C	D	E	F	Overall
DSTIFF	10** <sup>-2</sup>	346	1,001	804	378	534	2,135	5,198
	10** <sup>-4</sup>	957	2,462	1,715	962	1,207	2,927	10,230
	10** <sup>-6</sup>	1,970	4,157	2,891	1,686	2,235	4,633	17,572
LSODE	10** <sup>-2</sup>	250	3,613	528	300	381	1,198	5,820
	10** <sup>-4</sup>	535	3,672	1,028	623	712	2,086	8,656
	10** <sup>-6</sup>	1,007	4,622	1,711	1,267	1,375	3,634	13,616

TABLE 3  
*The total number of Jacobian evaluations and LU decompositions (overall results of each problem class).*

Problem class		A	B	C	D	E	F	Overall
DSTIFF	10** <sup>-2</sup>	61	72	78	60	59	159	489
	10** <sup>-4</sup>	88	90	104	96	82	145	605
	10** <sup>-6</sup>	106	98	118	130	101	217	770
LSODE	10** <sup>-2</sup>	59	206	86	67	75	183	676
	10** <sup>-4</sup>	86	248	127	112	100	229	902
	10** <sup>-6</sup>	119	289	160	166	144	346	1,224

TABLE 4  
*The total time in seconds (overall results of each problem class).*

Problem class		A	B	C	D	E	F	Overall
DSTIFF	10** <sup>-2</sup>	1.741	3.335	2.560	0.742	1.130	5.131	14.640
	10** <sup>-4</sup>	6.511	10.434	6.375	2.099	3.191	9.123	37.733
	10** <sup>-6</sup>	14.118	22.416	12.376	4.515	6.993	16.789	77.207
LSODE	10** <sup>-2</sup>	1.664	18.288	2.159	0.930	1.261	3.969	28.271
	10** <sup>-4</sup>	3.765	21.490	4.355	1.958	2.499	7.032	41.100
	10** <sup>-6</sup>	7.100	26.282	7.331	4.169	5.049	12.678	62.609



TABLE 5  
The total number of steps taken (overall results of each problem class).

Problem class		A	B	C	D	E	F	Overall
DSTIFF	10** <sup>-2</sup>	204	484	390	173	250	918	2,419
	10** <sup>-4</sup>	620	1,176	866	467	607	1,349	5,085
	10** <sup>-6</sup>	1,189	2,139	1,526	869	1,180	2,364	9,267
LSODE	10** <sup>-2</sup>	187	2,804	395	184	255	803	4,628
	10** <sup>-4</sup>	444	3,123	805	406	542	1,473	6,793
	10** <sup>-6</sup>	832	3,989	1,392	922	1,100	2,767	11,002

TABLE 6  
The local error deception (overall results of each problem class).

Problem class		Fraction of steps on which local error > tolerance						Overall
		A	B	C	D	E	F	
DSTIFF	10** <sup>-2</sup>	0.000	0.004	0.010	0.006	0.020	0.024	0.014
	10** <sup>-4</sup>	0.000	0.006	0.016	0.004	0.013	0.020	0.011
	10** <sup>-6</sup>	0.004	0.008	0.006	0.000	0.011	0.019	0.010
LSODE	10** <sup>-2</sup>	0.059	0.339	0.096	0.098	0.141	0.173	0.258
	10** <sup>-4</sup>	0.027	0.443	0.063	0.059	0.110	0.145	0.256
	10** <sup>-6</sup>	0.053	0.359	0.093	0.060	0.043	0.145	0.192

We note that LSODE and DSTIFF are similar codes in many ways. Both codes use Nordsieck representation of the linear multistep formulas and both carry out about the same amount of work per step. Therefore the statistics provided by New Detest are a good basis for comparing the two codes.

We note that DSTIFF is very efficient, specially at larger tolerances. The main reason for the much better overall performance of DSTIFF is the problem B5 which is solved by DSTIFF very efficiently.

We note that DSTIFF failed to successfully integrate problem E5 at tolerances of  $10^{-2}$  and  $10^{-4}$ . Problem E5 is a semi-stable chemical kinetics problem that can become unstable if a computed solution becomes negative. Shampine (1981) and Enright and Hull (1976) discuss the difficulty of solving such problems.

On the basis of the above testing we can say that:

(1) The code DSTIFF seems to be more reliable than LSODE since the deception of DSTIFF is much smaller than for LSODE.

(2) The number of Jacobian evaluations are smaller for DSTIFF than for LSODE because LSODE reevaluates Jacobians regularly even if there is no need for it.

(3) DSTIFF solved the problems in Class B much more efficiently than LSODE. This is because the problem B5 was solved by DSTIFF in 134, 358 and 615 steps at the three tolerances respectively, while LSODE took 2351, 2397 and 2620 steps respectively. Problem B5 has eigenvalues very close to the imaginary axis ( $-10 \pm 100i$ ).

(4) On problems other than those which have large imaginary eigenvalues, LSODE is usually more efficient than DSTIFF, but is usually less reliable in keeping the local error below the user specified tolerance.

(5) Somewhat surprisingly, in spite of the higher order formulas used by DSTIFF, LSODE seems to be more efficient at stringent tolerances. This is partly because of

the large truncation error coefficients of the higher order formulas. Also the roots at infinity of the higher order formulas are close to 0.9.

(6) The average number of function evaluations per step is close to 2.0 for DSTIFF while for LSODE, it is only 1.25. This is because of somewhat different stopping criterion used in the codes. It also may be partly due to slightly larger leading coefficients of the higher order formulas in DSTIFF. We note however that if the corrector tests of both codes were to be changed to the test recommended by Shampine (1979) to improve reliability, we expect both codes to take about two function evaluations per step. Therefore the change probably will have much more impact on the performance of LSODE than on the performance of DSTIFF.

(7) The CPU time taken per step on the average is close to 0.006 seconds for LSODE while for DSTIFF, the times are 0.0060, 0.0074 and 0.0083 seconds for the three tolerances respectively. The time taken per step is larger for DSTIFF because of the larger number of function evaluations per step and, at stringent tolerances, due to the higher order formulas used.

Another set of tests were run to evaluate the performance of DSTIFF in solving stiff equations when analytic Jacobian is not available and the code must approximate the Jacobian by finite differencing. Table 7 gives the overall results for DSTIFF and LSODE. DSTIFF failed to complete the integration for Problem F4 at  $TOL = 10^{-6}$  and therefore results for the tolerance  $10^{-6}$  for LSODE also do not include the cost of solving F4.

TABLE 7  
Overall results with numerically approximated Jacobian (\*excluding cost of solving problem F4).

	TOL	Time	Fcn calls	Mat fact	No. of steps
DSTIFF	$10^{-2}$	14.884	7,244	488	2,417
	$10^{-4}$	37.833	12,804	605	5,085
	$10^{-6}$	64.714	16,884	648	7,522 (*)
LSODE	$10^{-2}$	27.797	8,506	652	4,510
	$10^{-4}$	40.942	12,394	882	6,723
	$10^{-6}$	54.887	15,700	1,028	9,007 (*)

TABLE 8  
Overall results when the maximum order in DSTIFF is set to 5, 7 and 10 (excluding class F).

Max order	TOL	Time	Fcn calls	Mat fact	No. of steps
5	$10^{-2}$	9.308	3,210	331	1,551
	$10^{-4}$	24.257	7,042	473	3,857
	$10^{-6}$	50.244	13,769	520	7,631
	Overall	83.809	24,021	1,324	13,039
7	$10^{-2}$	9.208	3,081	331	1,499
	$10^{-4}$	26.375	7,134	448	3,694
	$10^{-6}$	53.204	12,743	565	6,841
	Overall	88.787	22,958	1,344	12,034
10	$10^{-2}$	9.509	3,063	330	1,501
	$10^{-4}$	28.610	7,303	460	3,736
	$10^{-6}$	60.418	12,939	553	6,903
	Overall	98.537	23,305	1,343	12,140

To evaluate the effect of the high order formulas on the efficiency of DSTIFF, we conducted another set of tests. We used DSTIFF with the maximum order of formulas restricted to 5 and then with maximum orders of 7 and 10 to solve the New Detest stiff problems. The results obtained are summarised in Table 8.

For better comparison of results we have excluded class *F* problems from the summary in Table 8. This has been done because for maximum order of 5 we had three failures in Class *F* while maximum order 7 and 10 had only two failures. We note that the results at  $TOL = 10^{-2}$  are not affected very much by setting maximum order to as low as 5. For the more stringent tolerances, maximum order of 7 seems to be better than maximum order of 5 or 10 since the number of function evaluations and the number of steps for maximum order 7 are the lowest.

**7. Concluding remarks.** The code DSTIFF has been shown to be a reliable and efficient code. It is much more efficient than LSODE in solving problems which have Jacobians with eigenvalues having large imaginary parts. On the other problems, DSTIFF is almost as efficient as LSODE on larger tolerances and somewhat less efficient than LSODE on the stringent tolerances.

The evaluation of DSTIFF using New Detest is only one indicator of its performance. For example, the Jacobians in New Detest are so cheap to evaluate that the fact that one code takes smaller numbers of Jacobian evaluations is not really reflected in the CPU time statistics provided by New Detest. We further note that if a code fails on one or more of the problems in New Detest, comparison of codes becomes very difficult. Also it is difficult to say how serious a failure should be considered when some of the problems in the set are very difficult problems. Recently Shampine (1981) has critically looked at the set of problems in New Detest (except class *F*) and it is hoped that New Detest will be modified in line with Shampine's comments.

We hope to further improve DSTIFF and are studying why DSTIFF does not perform better at stringent tolerances. We are looking at the order changing algorithm and are considering reducing the maximum order used in the code.

A copy of the code DSTIFF may be obtained by writing to the author.

**Acknowledgments.** I would like to thank Drs. Ron Sacks-Davis, Larry Shampine and Peter Tischer and two anonymous referees for their comments on an earlier version of this paper. Programming assistance of Kao Chung-Cheng is gratefully acknowledged.

#### REFERENCES

- C. A. ADDISON (1979), *Implementing a stiff method based upon the second derivative formulas*, Technical Report 130, Dept. Computer Science, Univ. Toronto, Toronto, Ontario.  
 ——— (1980), *Results of the 1979 ODE olympics*, unpublished manuscript.
- K. BURRAGE (1978), *A special family of Runge-Kutta methods for solving stiff differential equations*, BIT, 18, pp. 22-41.
- J. BUTCHER, K. BURRAGE AND F. CHIPMAN (1979), STRIDE: *Stable Runge-Kutta integrator for differential equations*, Computational Mathematics Report No. 19, University of Auckland; BIT, 20 (1980) pp. 326-340.
- G. D. BYRNE AND A. C. HINDMARSH (1975), *A polyalgorithm for the numerical solution of ordinary differential equations*, ACM-Trans. Math. Software, 1, pp. 71-96.
- W. H. ENRIGHT (1972), *Studies in the numerical solution of stiff ordinary differential equations*, Technical Report No. 46, Dept. Computer Science, Univ. Toronto, Toronto, Ontario.
- W. H. ENRIGHT, T. E. HULL AND B. LINDBERG (1975), *Comparing numerical methods for stiff systems of ODE's*, BIT, 15, pp. 10-48.
- W. H. ENRIGHT AND T. E. HULL (1976), *Comparing numerical methods for the solution of stiff systems of ODE's arising in chemistry*, in Numerical Methods for Differential Systems, L. Lapidus and W. E. Schiesser, eds., Academic Press, New York, pp. 45-66.

- W. H. ENRIGHT (1979), *Using a test package for the automatic assessment of methods for ODE's in Performance Evaluation of Numerical Software*, L. D. Fosdick, ed., North-Holland, Amsterdam, pp. 199-213.
- C. W. GEAR (1971), *Numerical Initial Value Problems in Ordinary Differential Equations*, Prentice-Hall, Englewood Cliffs, NJ.
- G. K. GUPTA (1975), *New multistep methods for the solution of ordinary differential equations*, Ph.D. Thesis, Dept. Computer Science, Monash Univ. Clayton, Victoria, Australia.
- (1976), *Some new high-order multistep formulae for solving stiff equations*, Math. Comp., 30, pp. 417-432.
- (1979), *DSTIFF: A set of subroutines for solving stiff equations—User's guide part 1*, Technical Report No. 3, Dept. Computer Science, Monash Univ., Clayton, Victoria, Australia.
- G. K. GUPTA, R. SACKS-DAVIS AND P. TISCHER (1983), *A review of recent developments in solving ODE's*, Technical Report No. 24, Dept. of Computer Science, Monash University, Clayton, Victoria, Australia, ACM Computing Surveys, to appear.
- G. K. GUPTA AND C. S. WALLACE (1975), *Some new multistep methods for solving ordinary differential equations*, Math. Comp., 29, pp. 489-500.
- (1979), *A new-step-size changing technique for multistep methods*, Math. Comp., 33, pp. 125-138.
- A. C. HINDMARSH (1974), *GEAR: Ordinary differential equation system solver*, Tech. Report UCID-30001, Rev. 3, Lawrence Livermore Laboratory, Livermore, CA.
- (1980), *LSODE and LSODI, two new initial value ordinary differential equation solvers*, ACM SIGNUM Newsletter, 15, 4, pp. 10-11.
- P. KAPS AND P. RENTROP (1979), *Generalized Runge-Kutta methods of order four with stepsize control for stiff ordinary differential equations*, Numer. Math., 33, pp. 55-68.
- R. SACKS-DAVIS (1980), *Fixed leading coefficient implementation of SD-formulas for stiff ODE's*, ACM Trans. Math. Software, 6, pp. 540-562.
- L. F. SHAMPINE (1981), *Evaluation of a test set for stiff ODE solvers*, ACM Trans. Math. Software, 7, pp. 409-420.
- L. F. SHAMPINE AND C. W. GEAR (1979), *A user's view of solving stiff ordinary differential equations*, SIAM Rev., 21, pp. 1-17.
- L. F. SHAMPINE (1979), *Evaluation of implicit formulas for the solution of ODE's*, BIT, 19, pp. 495-502.
- (1980), *Implementation of implicit formulas for the solution of ODE's*, this Journal, 1, pp. 119-130.
- R. D. SKEEL AND A. K. KONG (1977), *Blended linear multistep methods*, ACM Trans. Math. Software, 3, pp. 326-345.
- J. M. TENDLER, T. A. BICKART AND Z. PICEL (1978), *A stiffly stable integration process*, ACM Trans. Math. Software, 4, pp. 399-403.
- C. S. WALLACE AND G. K. GUPTA (1973), *General linear multistep methods to solve ordinary differential equations*, Aust. Comp. J., 5, pp. 62-69.
- O. B. WIDLUND (1967), *A note on unconditionally stable linear multistep methods*, BIT, 7, pp. 65-70.

## EXPANDED CONVERGENCE DOMAINS FOR NEWTON'S METHOD AT NEARLY SINGULAR ROOTS\*

D. W. DECKER† AND C. T. KELLEY‡

**Abstract.** We consider Newton's method for a class of nonlinear equations for which the derivative is nearly singular at the root. We show that there is a larger than expected domain of attraction for the Newton iterates but that convergence appears to be only linear in most of this domain. We give a modification of Newton's method that improves this slow convergence.

**Key words.** Newton's method, singularity, convergence rate, acceleration, bifurcation

**1. Introduction.** Newton's method has long been widely used in the solution of nonlinear functional equations, which we denote in the form  $F(x) = 0$ , where  $F$  maps some Banach space  $E$  into itself. Starting with some initial guess  $x_0$ , the Newton iterates are defined as

$$(1.1) \quad x_{i+1} = x_i - F'(x_i)^{-1}F(x_i), \quad i = 0, 1, \dots$$

If  $x^*$  denotes a root and the Fréchet derivative  $F'(x)$  is Lipschitz continuous near  $x^*$  and is nonsingular at  $x^*$ , then the iterates will converge quadratically to  $x^*$ , provided  $x_0$  is chosen sufficiently close [12]. That is, there is a  $K > 0$ , such that

$$(1.2) \quad \|x_{i+1} - x^*\| \leq K \|x_i - x^*\|^2, \quad i = 0, 1, \dots$$

Recently, much attention has been directed at the situation where  $F'(x^*)$  is singular [2]-[11], [16], [18]-[21]. The convergence behavior demonstrated is quite varied and depends on the precise singular structure of  $F'$  and higher derivatives. Three features, however, are common to these results:

(i) The initial Newton iterate cannot be chosen arbitrarily within a small ball about  $x^*$  but must be a member of a more restrictive set.

(ii) The overall convergence rate for the iterate error is reduced from quadratic to linear.

(iii) The iterate error may be decomposed into two components:

(1) the error in the direction of the null space of  $F'(x^*)$ ,

(2) the error in the remaining range space directions; with the second component of smaller order than the first. (For example, the range space error proportional to the square of the nullspace error is common [6].)

In light of the slower convergence for singular problems, work has been undertaken to devise modifications of Newton's method that will return quadratic or at least superlinear convergence ([5], [7], [16], [18], [20], [21] and in particular see the excellent survey [9]).

The purpose of this paper is to address the following question: Suppose  $F'(x^*)$  is not singular, but is nearly so; what convergence behavior may be expected of Newton's method? Before considering this question in detail we indicate why nearly singular problems may be considered important. Path following techniques for the

---

\* Received by the editors April 26, 1983, and in revised form June 5, 1984.

† Department of Mathematics, North Carolina State University, Box 5548, Raleigh, North Carolina 27650. The work of this author was supported by the National Science Foundation under grants MCS-81-04254 and MCS-83-00841.

‡ Department of Mathematics, North Carolina State University, Box 5548, Raleigh, North Carolina 27650. Supported by the National Science Foundation under grants MCS-79-02659A01 and MCS-83-00841.

solution of nonlinear equations form an important component of numerical computation in almost every scientific discipline. In such computations the determination of intersecting solution branches is commonly required and hence nearly singular problems will arise naturally in the neighborhood of the branch (singular) points. In addition, a disproportionate share of the computational expense is usually involved in determining the solutions near the bifurcation points as a result of sensitivity to initial guess and slow convergence. Hence improved information on convergence domains and schemes for accelerating convergence for nearly singular problems could prove especially worthwhile.

That such problems must blend regular and singular convergence behavior can be seen in the following manner. Consider a sequence  $x_0, x_1, \dots$  of Newton iterates for a singular problem. If we perturb the problem very slightly to nearly singular, the initial iterates will be largely unchanged and hence will exhibit linear convergence. However since  $F'(x^*)$  is now nonsingular, the iterate behavior must become more regular and quadratic convergence must eventually be achieved.

The present work describes a basic singular problem and its companion nearly singular problem. These two problems, however, are intended to describe the convergence behavior to be expected in the neighborhood of simple bifurcation points. Hence, we first introduce the notation required to demonstrate this correspondence.

The general bifurcation problem may be formulated as

$$(1.3) \quad F(x; \varepsilon) = 0$$

where  $\varepsilon$  is a chosen continuation parameter and  $\varepsilon = 0$  denotes the bifurcation point. We let  $x^*(\varepsilon)$  denote a solution arc through the branch point. For the singular problem we make the following common assumptions [13], [22]. Let  $F'(x^*(0); 0)$  have a one-dimensional null space  $N_0$ ,

$$(1.4) \quad N_0 = \mathcal{N}(F'(x^*(0); 0)) = \text{span} \{ \phi_0 \}, \quad \|\phi_0\| = 1$$

and a range space  $X_0$ , which decompose the space  $E$ ,

$$(1.5) \quad X_0 = \mathcal{R}(F'(x^*(0), 0)), \quad E = N_0 \oplus X_0.$$

Let  $P_{N_0}$  denote a projection onto  $N_0$  parallel to  $X_0$  and  $P_{X_0} = I - P_{N_0}$ . With these assumptions we see  $F'(x^*(0); 0)$  is a Fredholm operator of index zero and hence will remain so for  $\varepsilon$  near zero. In this case it can be shown [2] that the structure of (1.4)–(1.5) is also preserved for small  $\varepsilon$ . That is, one may solve

$$(1.6) \quad F'(x^*(\varepsilon); \varepsilon)\phi(\varepsilon) = \mu(\varepsilon)\phi(\varepsilon)$$

for a continuous eigenvalue–eigenvector pair  $(\mu(\varepsilon), \phi(\varepsilon))$  with  $(\mu(0), \phi(0)) = (0, \phi_0)$ . In addition there is a codimension one subspace  $X(\varepsilon)$  which satisfies

$$(1.7) \quad F'(x^*(\varepsilon); \varepsilon)X(\varepsilon) = X(\varepsilon)$$

and

$$(1.8) \quad E = N(\varepsilon) \oplus X(\varepsilon)$$

where

$$(1.9) \quad N(\varepsilon) = \text{span} \{ \phi(\varepsilon) \}, \quad \|\phi(\varepsilon)\| = 1.$$

Hence one may define  $P_N(\varepsilon)$  to be the projection onto  $N(\varepsilon)$  parallel to  $X(\varepsilon)$  and  $P_X(\varepsilon) = I - P_N(\varepsilon)$ .

To simplify the notation we first assume  $\mu(\varepsilon) = \varepsilon$ . (Alternately, one could view this as choosing the small eigenvalue of the linearized problem as the new continuation parameter near the branch point.) Second, when discussing the nearly singular problem the explicit  $\varepsilon$  dependence ( $x^*(\varepsilon)$ ,  $N(\varepsilon)$ , etc.) is suppressed, while the singular problem is distinguished by a zero subscript ( $x_0^*$ ,  $N_0$ , etc.).

It is in this fashion that the two problems, singular and nearly singular, may be employed to illustrate the convergence behavior to be expected in the neighborhood of simple branch points. The following two sections determine the performance of Newton's method when applied to these two problems, which we now roughly summarize. (The singular case has been analyzed previously and is included here for completeness and comparison [6], [19].)

For the singular problem, if  $F$  is smooth, and a natural nondegeneracy condition on  $F''$  is satisfied, an initial iterate chosen from a pair of truncated cones centered at  $x_0^*$  (with symmetry axes in the  $\phi_0$  direction) will generate a sequence converging to  $x_0^*$ . The convergence is linear with overall rate  $\frac{1}{2}$ . For essentially no additional computational expense, an acceleration technique which modifies every other iterate can be shown to converge superlinearly to  $x_0^*$  [16].

For the nearly singular problem it is a simple matter to demonstrate a ball of radius the order of  $\varepsilon$  in which quadratic convergence to  $x^*$  is achieved. It is shown, however, that there is also a truncated cone in which convergence (although with initially linear rate) can be assured. For small  $\varepsilon$  this canonical region approaches one of the cones of the singular problem and hence becomes dramatically larger than the domain of quadratic convergence as  $\varepsilon$  shrinks to zero. This situation is presented pictorially in Fig. 1. Here we have a two-dimensional representation of the Banach space  $E$  (with axes in the  $N_0$  and  $X_0$  directions) and the chosen solutions  $x^*(\varepsilon)$  are then sketched in a neighborhood of the bifurcation point  $\varepsilon = 0$  as a curve in  $\varepsilon - E$  space. (The second bifurcation branch is indicated by dashed lines and the following construction could be applied to this curve as well.) It is expected that  $F'(x; \varepsilon)$  will also be singular away from the bifurcation point and in general the singular set will be a smooth manifold through  $(x^*(0), 0)$ . Under the assumption made on  $F''$  in the next section the  $\varepsilon = 0$  slice of this surface is transverse to the  $N_0$  direction and such a manifold is included in the figure. From the sketch it is then clear that a pair of cones in  $E$  centered about  $N_0$  directions (the second indicated by dashed lines) are regions

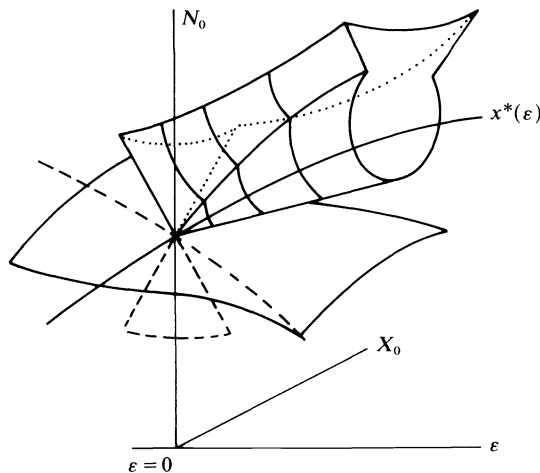


FIG. 1

of invertibility of  $F'$  for  $\varepsilon = 0$ ; while for  $\varepsilon \neq 0$  only one cone may be chosen in order to avoid the singular manifold. Finally the Newton iterate domain of attraction is then composed as the union of this conical region and the usual quadratic convergence ball.

For acceleration of the Newton iterates, it is shown that starting in the conical domain but well outside the quadratic ball the problem behaves like a singular problem and the singular acceleration modification may be employed. As the convergence proceeds the behavior becomes more regular and a test is presented which determines from iterate information when modification should be terminated. When employed, however, a modified step is assured to be an improvement, and the test is such that in the limit of  $\varepsilon = 0$  the modification is never terminated and overall superlinear convergence is achieved.

The proofs of these results for the nearly singular problem are contained in § 4. In § 5 an example is given which illustrates both the larger conical domain of attraction and the improved convergence achieved in this domain with the use of the proposed acceleration scheme.

**2. The singular problem.** The type of singular problem described in this section has been considered by several authors [3], [6]-[8], [18], [19] with the present formulation following that of [6].

We define the operators

$$(2.1) \quad \begin{aligned} A(x) &= P_{X_0} F'(x) P_{X_0}, & B(x) &= P_{X_0} F'(x) P_{N_0}, \\ C(x) &= P_{N_0} F'(x) P_{X_0}, & D(x) &= P_{N_0} F'(x) P_{N_0}, \end{aligned}$$

with  $A(x_0^*) = P_{X_0} F'(x_0^*) P_{X_0} \equiv \hat{F}$  which is invertible when viewed as an operator from  $X_0$  into itself. Then we have the following [6].

LEMMA 2.2. For  $\rho$  sufficiently small and  $\|x - x_0^*\| \leq \rho$ ,  $F'(x)$  is nonsingular if and only if

$$(2.3) \quad \tilde{D}(x) \equiv D(x) - C(x)A(x)^{-1}B(x)$$

is invertible (viewed as an operator from  $N_0$  into itself). Further, for  $\rho$  sufficiently small and  $\tilde{D}(x)$  invertible we have

$$(2.4) \quad \begin{aligned} F'(x)^{-1} &= P_{X_0}(A(x)^{-1} + A(x)^{-1}B(x)\tilde{D}(x)^{-1}C(x)A(x)^{-1})P_{X_0} \\ &\quad - P_{X_0}A(x)^{-1}B(x)\tilde{D}(x)^{-1}P_{N_0} \\ &\quad - P_{N_0}\tilde{D}(x)^{-1}C(x)A(x)^{-1}P_{X_0} + P_{N_0}\tilde{D}(x)^{-1}P_{N_0}. \end{aligned}$$

This result hence shifts the question of invertibility of  $F'(x)$  to that of  $\tilde{D}(x)$ .

We set  $\tilde{x} = x - x^*$  and define  $\beta_k(x)$  as any operator on  $E$ , vector in  $E$ , or scalar whose norm is at least  $O(\|\tilde{x}\|^k)$ . A region that will prove to be a domain of attraction for Newton's method we define as

$$(2.5) \quad W(\rho, \theta) = \{x \in E \mid 0 < \|\tilde{x}\| \leq \rho, \|P_{X_0}\tilde{x}\| \leq \theta\|P_{N_0}\tilde{x}\|\}.$$

The singular structure and the convergence behavior for this problem then rests upon

$$(2.6) \quad \lambda_0 \phi_0 \equiv P_{N_0} F''(x_0^*)(\phi_0, \phi_0)$$

for we have [6].

THEOREM 2.7. Assume  $\lambda_0 \neq 0$ . Then there is a  $\rho_0 > 0$ ,  $\theta_0 > 0$  such that  $F'(x_0)^{-1}$  exists for all  $x_0 \in W(\rho_0, \theta_0)$ . Further, all subsequent Newton iterates remain in this set and



converge to  $x_0^*$  with rate determined by

$$(2.8) \quad \|P_{x_0} \tilde{x}_i\| \leq K_1 \|\tilde{x}_{i-1}\|^3, \quad \text{some } K_1 > 0, \quad i \geq 1,$$

and

$$(2.9) \quad \lim_{i \rightarrow \infty} \frac{\|P_{N_0} \tilde{x}_i\|}{\|P_{N_0} \tilde{x}_{i-1}\|} = \frac{1}{2}.$$

This result illustrates the three basic features of singular problems described in the introduction. This result also provides an example of the convergence behavior at a simple branch point when the bifurcation is transverse. (That is, both branches exist for both  $\varepsilon > 0$  and  $\varepsilon < 0$  [22].) A second type of bifurcation, called super (sub) critical, or pitchfork occurs when one branch exists only for  $\varepsilon > 0$  or  $\varepsilon < 0$ . It is possible for such branch points to be irregular singular points in the sense of Griewank and Osborne [11] in which case the behavior of Newton's method may be extremely complex. On the other hand pitchfork bifurcation may occur when the problem is such that  $F''(x^*)(\cdot, \cdot) \equiv 0$  but  $P_N F'''(x^*)(\phi_0, \phi_0, \phi_0) \neq 0$ , and in this case the conclusions of Theorem 2.7 remain valid with  $\frac{1}{2}$  replaced by  $\frac{2}{3}$  in (2.9) [6]. Note that  $F''(x^*) \equiv 0$  certainly holds if  $F(x) = -F(-x)$ .

For an acceleration scheme we first consider

$$(2.10) \quad \begin{aligned} y &= x - F'(x)^{-1}F(x), \\ z &= y - 2F'(y)^{-1}F(y). \end{aligned}$$

If  $x \in W(\rho_0, \theta_0)$ , then so is  $y$  and it is a simple matter to show  $\|P_N z\| = \beta_2(x)$ . The problem, however, is that  $z$  may not be in  $W(\rho, \theta)$  and hence a correction term (larger than  $\beta_2(x)$ ) is added to remedy this defect. This is contained in [16].

**THEOREM 2.11.** *Assume  $\lambda_0 \neq 0$  and  $\alpha \in (0, 1)$  is given. Then if  $x_0 \in W(\rho_0, \theta_0)$  for  $\rho_0, \theta_0$  sufficiently small and  $y_n, \delta_n$  and  $x_{n+1}$  are given for  $n \geq 0$  by*

$$(2.12) \quad \begin{aligned} y_n &= x_n - F'(x_n)^{-1}F(x_n), \\ \delta_n &= F'(y_n)^{-1}F(y_n), \\ x_{n+1} &= y_n - (2 - \|\delta_n\|^\alpha)\delta_n, \end{aligned}$$

then  $x_n \in W(\rho_0, \theta_0)$  for all  $n$  and there is a  $K_1 > 0$  so that

$$\|\tilde{x}_{n+1}\| \leq K_1 \|\tilde{x}_n\|^{1+\alpha}, \quad n = 0, 1, \dots$$

**3. The nearly singular problem.** As outlined in the introduction, the structure of the singular problem is assumed to be such that

$$(3.1) \quad \begin{aligned} F'(x^*)\phi &= \varepsilon\phi, \quad \|\phi\| = 1, \quad N = \text{sp}\{\phi\}, \\ F'(x^*)X &= X, \quad E = N \oplus X, \end{aligned}$$

with  $P_N$  a projection onto  $N$  parallel to  $X$  and  $P_X = I - P_N$  (and with the explicit  $\varepsilon$  dependence of the subspaces and projections suppressed). The operators  $A(x)$ , etc. are defined for this problem in the same fashion as (2.1) and hence Lemma 2.2 may be applied as well.

We first estimate the radius of the ball about  $x^*$  in which the Newton iterates can be assured to converge. The standard requirement for an initial iterate  $x_0$  to generate a convergent Newton sequence may be written as [12]

$$(3.2) \quad L\|F'(x_0)^{-1}\| \|F'(x_0)^{-1}F(x_0)\| < \frac{1}{2}$$

where  $L$  is a Lipschitz bound on  $F'(x)$  near  $x_0$ . From (2.3) we see  $\tilde{D}(x) = \varepsilon P_N + \beta_1(x)$  and setting  $r = \|\tilde{x}\|$  we have  $\tilde{D}(x)$  invertible and  $\|\tilde{D}(x)^{-1}\| \leq k_1/\varepsilon$  provided

$$(3.3) \quad r < k_2\varepsilon, \quad 0 \leq \varepsilon \leq \bar{\varepsilon}$$

for some constants  $k_1, k_2 > 0$  and some chosen  $\bar{\varepsilon} > 0$ . From this and the expression (2.4) for  $F'(x)^{-1}$  we have

$$(3.4) \quad \|F'(x_0)^{-1}P_N\| \leq \frac{k_3}{\varepsilon}, \quad \|F'(x_0)^{-1}P_X\| \leq k_4$$

for some  $k_3, k_4 > 0$  and any  $x_0$  with  $\|\tilde{x}_0\| \leq r$  where  $r, \varepsilon$  satisfy (3.3). Now

$$(3.5) \quad F(x_0) = \varepsilon P_N \tilde{x}_0 + P_X \hat{F} P_X \tilde{x}_0 + \beta_2(x_0)$$

and since

$$(3.6) \quad \|F'(x_0)^{-1}F(x_0)\| \leq \|F'(x_0)^{-1}P_N\| \|P_N F(x_0)\| + \|F'(x_0)^{-1}P_X\| \|P_X F(x_0)\|$$

we see (3.2) is satisfied provided

$$(3.7) \quad \frac{1}{\varepsilon} \left( k_5 r + k_6 \frac{r^2}{\varepsilon} \right) < 1$$

for some  $k_5, k_6$  where  $r, \varepsilon$  satisfy (3.3). The relation (3.7) is quadratic in  $r/\varepsilon$  and is satisfied provided  $r/\varepsilon < k_7$  for some  $k_7 > 0$ . Hence there is a constant  $\nu > 0$  such that for all  $0 \leq \varepsilon \leq \bar{\varepsilon}$  the Newton iterates will converge quadratically to  $x^*$  provided  $x_0$  is chosen from

$$(3.8) \quad Q(\varepsilon) = \{x \in E \mid 0 \leq \|\tilde{x}\| \leq \nu\varepsilon\}.$$

For our purposes the constant  $\nu$  is unimportant, the desired conclusion being that the radius of the quadratic convergence ball is  $O(\varepsilon)$ .

We now construct a region of invertibility (modeled after the singular region) which for  $\varepsilon$  small is much larger than  $Q(\varepsilon)$ . We first note that all operator norms, say  $\|P_N F''(x^*)(\phi, P_N \cdot)\|$  are assumed to be the supremum over (the suppressed  $\varepsilon$  for  $0 < \varepsilon \leq \bar{\varepsilon}$  for some fixed  $\bar{\varepsilon} > 0$ . Defining

$$(3.9) \quad \begin{aligned} \lambda\phi &= P_N F''(x^*)(\phi, \phi), \\ \mu\phi &= P_N \tilde{x}, \\ s &= \text{sgn}(\varepsilon\lambda), \end{aligned}$$

we consider the conical region

$$(3.10) \quad W_s(\rho, \theta) = \{x \in E \mid 0 < \|\tilde{x}\| \leq \rho, \|P_X \tilde{x}\| \leq \theta \|P_N \tilde{x}\|, \text{sgn}(\mu) = s\}.$$

From (2.3) we see

$$(3.11) \quad \tilde{D}(x) = \varepsilon P_N + D_1(x) + D_2(x) - C(x)A^{-1}(x)B(x) + \beta_3(x)$$

where  $D_j(x) \equiv P_N F^{(j+1)}(x^*)(\tilde{x}^j, P_N \cdot)$ ,  $j = 1, 2, \dots$  (and similarly later for  $A_j(x)$ ,  $B_j(x)$ , etc.). With the definitions (3.9)

$$(3.12) \quad \tilde{D}(x)\phi = (\varepsilon + \lambda\mu)\phi + P_N F''(x^*)(P_X \tilde{x}, \phi) + \beta_2(x).$$

From this we see  $\tilde{D}(x)$  invertible would require  $|\lambda\mu| < \varepsilon$  but if one chooses  $\text{sgn}(\mu) = s$  the leading term of (3.12) will never vanish. Now since  $\|\tilde{x}\| \leq (1 + \theta)\mu \leq 2\mu$ , for  $\theta < 1$ , we have

$$(3.13) \quad \|\tilde{D}(x) - (\varepsilon P_N + D_1(x))\| \leq \gamma_1 \mu^2$$

for some constant  $\gamma_1$  and  $\|\tilde{x}\| \leq \rho$  sufficiently small. If we set

$$(3.14) \quad \kappa = \max_{\|P_N \tilde{x}\|=1} \|P_N F''(x^*)(P_N \tilde{x}, P_N \cdot)\|$$

we see  $D(x)$  will be invertible provided

$$(3.15) \quad |\varepsilon + \lambda\mu| - (\kappa\theta|\mu| + \gamma_1\mu^2) > 0.$$

Now assuming that  $\lambda \neq 0$  and  $x \in W_s(\rho, \theta)$ , then  $|\varepsilon + \lambda\mu| = |\varepsilon| + |\lambda\mu| \geq |\lambda\mu|$  and hence (3.15) can be assured provided

$$(3.16) \quad \kappa\theta|\mu| + \gamma_1\mu^2 < \frac{|\lambda\mu|}{2}.$$

Notice that (3.16) places restrictions on  $\rho, \theta$  that guarantee invertibility of  $\tilde{D}(x)$  independent of  $\varepsilon \leq \bar{\varepsilon}$ . Hence we may state the next lemma.

LEMMA 3.17. *There exist  $\rho > 0, \theta > 0$ , independent of  $\varepsilon \leq \bar{\varepsilon}$  such that if  $\lambda \neq 0$  and  $x \in W_s(\rho, \theta)$ ,  $\tilde{D}(x)^{-1}$  exists and may be defined as*

$$(3.18) \quad \tilde{D}(x)^{-1}\phi \equiv \frac{1}{(\varepsilon + \bar{\lambda}\mu)}\phi$$

where  $\bar{\lambda} = \bar{\lambda}(x)$  and satisfies

$$(3.19) \quad \frac{|\lambda\mu|}{2} < |\bar{\lambda}(x)\mu| < \frac{3|\lambda\mu|}{2}.$$

From this lemma and Lemma 2.2 we have  $F'(x)$  invertible for  $x \in W_s(\rho, \theta)$  where  $\rho, \theta$  are chosen to insure satisfaction of (3.13), (3.16). In this case the subsequent iterate

$$(3.20) \quad y = x - F'(x)^{-1}F(x)$$

is well defined and will satisfy [6]

$$(3.21) \quad \tilde{y} = \frac{1}{2}F'(x)^{-1}\{(A_1(x) + B_1(x) + C_1(x) + D_1(x))\tilde{x} + \beta_3(x)\}.$$

Lemma 3.17 and the expression (2.4) for  $F'(x)^{-1}$  can then be employed to show  $y \in W_s(\rho, \theta)$  and hence the process may be repeated. This is contained in the following theorem:

THEOREM 3.22. *Assume  $\lambda \neq 0$ , and  $0 \leq \varepsilon \leq \bar{\varepsilon}$ . Then there are continuous functions  $\rho = \rho(\varepsilon) > 0, \theta = \theta(\varepsilon) > 0$ , monotonically increasing as  $\varepsilon \downarrow 0$  such that if  $x_0 \in W_s(\rho, \theta)$  then  $F'(x_0)^{-1}$  exists and all subsequent Newton iterates remain in this set and converge to  $x^* = x^*(\varepsilon)$ . Further*

$$(3.23) \quad \|P_N \tilde{x}_{i+1}\| < \frac{3}{4}\|P_N \tilde{x}_i\|, \quad i = 0, 1, \dots, \text{some } K > 0$$

$$(3.24) \quad \|P_X \tilde{x}_{i+1}\| \leq K\|x_i\|^2,$$

and  $\rho(\varepsilon), \theta(\varepsilon)$  may be chosen such that  $\rho(0) = \rho_0, \theta(0) = \theta_0$  where  $\rho_0, \theta_0$  are values for which the conclusions of Theorem 2.7 hold.

*Proof.* Contained in § 4.  $\square$

The above result hence provides an enlarged convergence domain in the neighborhood of certain transverse bifurcations (see § 2). For pitchfork bifurcations, the above result does not apply when the underlying singular problem is irregular [11], and it is expected that a quite different analysis would be required for this situation. The result, however, may be directly extended to include those examples of super or subcritical bifurcation where it is assumed the singular problem satisfies  $F''(x^*)(\cdot, \cdot) \equiv 0$  but

$P_N F'''(x^*)(\phi_0, \phi_0, \phi_0) \neq 0$ . For this case the conclusions of Theorem 3.22 remain valid with  $\frac{3}{4}$  replaced with  $\frac{8}{9}$  in (3.23).

We now turn to the question of acceleration of the Newton iterates in the expanded domain  $W_s(\rho, \theta)$ . In [9], it is pointed out that the acceleration scheme described by Theorem 2.11 will still give superlinear convergence in a sufficiently small ball about a regular point, and numerical results are reported for a nearly singular problem using this approach. However, as we show by example in § 5, if  $x_0 \in W_s(\rho(\varepsilon), \theta(\varepsilon))$  and the scheme (2.12) is employed, the iterates may converge to a root other than  $x^*$ . The following result, however, provides an expanded domain  $W_s(\rho_0, \theta_0)$ , possibly smaller than  $W(\rho(\varepsilon), \theta(\varepsilon))$ , but independent of  $\varepsilon$ , in which the modified iterates of Theorem 2.11 remain. This result also presents a test which assures that acceleration is really an improvement. First, for  $x \in W_s(\rho, \theta)$  we define  $\delta(x)$  by

$$(3.25) \quad \delta(x) = F'(x)^{-1}F(x).$$

**THEOREM 3.26.** *Assume  $\lambda \neq 0$  and  $0 < \varepsilon < \bar{\varepsilon}$ . Let  $\alpha \in (0, \frac{1}{2})$ ,  $\gamma \in (2\alpha, 1)$  and  $C > 0$  be given. Then there exists  $\rho_a > 0$ ,  $\theta_a > 0$  sufficiently small, independent of  $\varepsilon$ , such that if  $x \in W_s(\rho_a, \theta_a)$  and*

$$(3.27) \quad \begin{aligned} (a) \quad & \|\delta(y) - 2\delta(z)\| \leq \|\delta(y)\|^{1+\gamma}, \\ (b) \quad & \|\delta(y)\| \geq \frac{1}{3}\|\delta(x)\|, \end{aligned}$$

where

$$(3.28) \quad \begin{aligned} y &= x - \delta(x), \\ z &= y - \delta(y), \end{aligned}$$

then  $x_a$ , given by

$$(3.29) \quad x_a = z - (2 - C\|\delta(z)\|^\alpha)\delta(z),$$

is in  $W_s(\rho_a, \theta_a)$  and

$$(3.30) \quad \|\tilde{x}_a\| \leq K\|\tilde{x}\|^{1+\alpha} \leq \|\tilde{z} - \delta(z)\|$$

for some  $K > 0$ .

As mentioned previously, the inequalities (3.27) act as a test to determine if acceleration is worthwhile. When the inequalities hold, the conclusion (3.30) indicates that it is an improvement over a regular Newton iterate. We note that for a singular problem the tests (3.27) will always be satisfied (for sufficiently small  $\rho$ ) for any  $\gamma < 1$ , and acceleration will never terminate and superlinear convergence of  $\{x_n\}$  with exponent  $1 + \alpha$ ,  $\alpha \in (0, \frac{1}{2})$ , will be achieved. We point out however, that for this singular case, the convergence is poorer than Theorem 2.11 provides. First, one rather than two intermediate iterates are required, and second  $\alpha \in (0, 1)$  is allowed in the singular result. These changes are the result of the test (3.27), which were designed to measure “singular” behavior of the iterates, and which hence perform this task in less than optimal fashion. The design of a more efficient “singularity” test must therefore be the object of further study. The above result, however, demonstrates two basic facts, (1) acceleration of the Newton iterates in the enlarged convergence domain is possible, and (2) that some test must be employed to terminate iterate modification when it is no longer productive.

Finally, we note this acceleration result also may be extended to the particular super (sub) critical bifurcation problem described previously by substituting 3 in place of 2 in the acceleration step given by (3.29).

**4. Proofs.**

*Proof of Theorem 3.22.* Without loss of generality we handle the case  $\varepsilon > 0, \lambda > 0$ . For  $x \in W_+(\rho, \theta)$  we assume  $\rho, \theta$  are such that (3.13) and (3.16) are satisfied and hence Lemma 3.17 holds. Using  $x_0 = x$  and  $x_1 = y$ , (3.21) and the definitions (2.1) yield

$$(4.1) \quad \tilde{y} = F'(x)^{-1} \left\{ \frac{1}{2} (A(x) + B(x) + C(x) + D(x)) \tilde{x} - \frac{1}{2} (\hat{F} + \varepsilon P_N) \tilde{x} + \beta_3(x) \right\}.$$

Using the expression (2.4) for  $F'(x)^{-1}$  and noting  $\tilde{D}(x)^{-1} D(x) = P_N - \tilde{D}(x)^{-1} \beta_2(x) P_N$  we have

$$(4.2) \quad P_N \tilde{y} = \frac{1}{2} [P_N - \tilde{D}(x)^{-1} (\varepsilon P_N + C(x) + \beta_2(x))] \tilde{x}$$

while

$$(4.3) \quad P_X \tilde{y} = \frac{1}{2} \{ P_X \hat{F}^{-1} A_1(x) \tilde{x} + P_X \hat{F}^{-1} B(x) \tilde{D}(x)^{-1} [(\varepsilon P_N + C(x)) \tilde{x} + \beta_3(x)] + \beta_3(x) \}.$$

Now from (4.2), using the definition of  $\tilde{D}(x)^{-1}$  given by (3.18) we find

$$(4.4) \quad P_N \tilde{y} = \frac{1}{2} \left( \frac{1}{\varepsilon + \bar{\lambda} \mu} \right) \{ \bar{\lambda} \mu P_N - C(x) - P_N \beta_2(x) \} \tilde{x}.$$

From (4.4) we have the upper and lower bounds

$$(4.5) \quad \|P_N \tilde{y}\| \leq \frac{1}{2} \left\{ \frac{\bar{\lambda} \mu}{\varepsilon + \bar{\lambda} \mu} + \frac{\kappa \theta \mu + \gamma_2 \mu^2}{\varepsilon + \bar{\lambda} \mu} \right\} \|P_N \tilde{x}\|,$$

$$(4.6) \quad \|P_N \tilde{y}\| \geq \frac{1}{2} \left\{ \frac{\bar{\lambda} \mu}{\varepsilon + \bar{\lambda} \mu} - \frac{(\kappa \theta \mu + \gamma_2 \mu^2)}{\varepsilon + \bar{\lambda} \mu} \right\} \|P_N \tilde{x}\|.$$

Here the term  $\gamma_2 \mu^2$ , for some constant  $\gamma_2$ , represents the remainder term  $\beta_2(x)$  in (4.2). Now if we further assume  $\rho$  and  $\theta$  so small that

$$(4.7) \quad \kappa \theta \mu + \gamma_2 \mu^2 < \frac{\lambda \mu}{4}$$

then since (3.19) is satisfied we see  $\kappa \theta \mu + \gamma_2 \mu^2 < \bar{\lambda} \mu / 2$  and hence (4.5)-(4.6) becomes

$$(4.8) \quad \frac{1}{4} \frac{\bar{\lambda} \mu}{\varepsilon + \bar{\lambda} \mu} \|P_N \tilde{x}\| \leq \|P_N \tilde{y}\| \leq \frac{3}{4} \frac{\bar{\lambda} \mu}{\varepsilon + \bar{\lambda} \mu} \|P_N \tilde{x}\|.$$

Turning to (4.3), since  $x \in W_+(\rho, \theta)$  there are constants  $a, b$  such that

$$(4.9) \quad \|\hat{F}^{-1} A_1(x) \tilde{x}\| \leq a \theta \mu^2, \quad \|\hat{F}^{-1} B(x) \tilde{x}\| \leq b \mu^2$$

and hence

$$(4.10) \quad \|P_X \tilde{y}\| \leq \frac{1}{2} \left\{ a \theta \mu^2 + \frac{b \mu}{\varepsilon + \bar{\lambda} \mu} (\varepsilon \mu + \kappa \theta \mu^2 + \gamma_3 \mu^3) + \gamma_4 \mu^3 \right\}.$$

Once again the terms  $\gamma_3 \mu^3$  and  $\gamma_4 \mu^3$  represent the two remainder terms in (4.3). Now since  $\bar{\lambda} \mu / (\varepsilon + \bar{\lambda} \mu)$  decreases as  $\bar{\lambda} \mu$  decreases, we see from (4.8) and (3.19) that

$$(4.11) \quad \|P_N \tilde{y}\| \geq \frac{1}{4} \frac{\lambda \mu^2}{(2\varepsilon + \lambda \mu)}.$$

From (4.10-11) and  $\bar{\lambda} \geq \lambda / 2$  we see  $\|P_X \tilde{y}\| \leq \theta \|P_N \tilde{y}\|$  provided

$$(4.12) \quad 2 \left\{ a \theta + \frac{2b}{2\varepsilon + \lambda \mu} (\varepsilon + \kappa \theta \mu + \gamma_3 \mu^2) + \gamma_4 \mu \right\} \leq \frac{\theta \lambda}{2\varepsilon + \lambda \mu}.$$

First we note that since  $(1 - \theta)\mu \leq \rho \leq (1 + \theta)\mu$ , that smallness restrictions on  $\mu, \theta$  are equivalent to smallness restrictions on  $\rho, \theta$ . Up to this point such restrictions were made (i) to provide bounds for remainder terms as in (3.13) or (ii) to provide bounds relating to  $\tilde{D}(x)^{-1}$  as in (3.16) and (4.7). Requirements of both type were independent of  $\varepsilon$  for  $0 \leq \varepsilon \leq \bar{\varepsilon}$  sufficiently small and precisely the same restrictions are made in choosing  $\rho_0, \theta_0$  to satisfy the singular Theorem 2.7. In addition, for  $\varepsilon = 0$ , inequality (4.12) provides the final determination of an allowable  $\rho_0, \theta_0$  for this singular result. For  $0 < \varepsilon \leq \bar{\varepsilon}$  sufficiently small, however, the requirement (4.12) can still be satisfied for  $\mu, \theta$  sufficiently small and one may choose the resulting  $\rho(\varepsilon), \theta(\varepsilon)$  to be monotone increasing as  $\varepsilon$  decreases with  $\rho(0) = \rho_0$  and  $\theta(0) = \theta_0$ .

Finally we see from (4.8) that

$$(4.13) \quad \|P_N \tilde{y}\| < \frac{3}{4} \|P_N \tilde{x}\|$$

and from (4.10) there is a  $K > 0$  such that

$$(4.14) \quad \|P_X \tilde{y}\| \leq K \|\tilde{x}\|^2$$

thus providing the bounds (3.23), (3.24).  $\square$

The proof of Theorem 3.26 rests on an elementary extension of a result from [16]. The perspective of this result is to view the parameter  $\varepsilon$  as a perturbation of the singular problem which may be varied with the current iterate  $x$ . The result answers the question, how fast must the perturbation  $\varepsilon$  shrink with the iterate  $x$  in order that the asymptotic acceleration error estimates remain those of the *singular* problem? We have

LEMMA 4.15. *Let  $\lambda \neq 0$  and  $0 < \varepsilon \leq \bar{\varepsilon}$ . Let  $K_0, C > 0$  and  $\alpha \in (0, 1), \gamma \in (\alpha, 1)$  be given. Then there exist  $\rho_a > 0, \theta_a > 0$ , independent of  $\varepsilon$ , such that if  $x \in W_s(\rho_a, \theta_a)$ ,  $\varepsilon \leq K_0 \|\tilde{x}\|^{1+\gamma}$  and  $x_a$  is given by (3.29), then  $x_a \in W_s(\rho_a, \theta_a)$  and (3.30) holds. Moreover, there exist  $K_1, K_2 > 0$  such that*

$$(4.16) \quad \begin{aligned} \|P_X \tilde{x}_a\| &\leq K_1 \|\tilde{x}\|^2, \\ \|P_N \tilde{x}_a\| &\leq K_2 \|\tilde{x}\|^{1+\alpha}. \end{aligned}$$

The proof of Lemma 4.15 is virtually identical to the  $\varepsilon = 0$  case in [13] and is hence merely sketched. If  $\varepsilon = \beta_{1+\gamma}(x)$  then by (4.3)

$$(4.17) \quad \begin{aligned} P_X \tilde{y} &= O(\|P_N \tilde{x}\| \cdot \|P_X \tilde{x}\|) + O(\varepsilon \|P_N \tilde{x}\|) \\ &= \theta \beta_2(x) + \beta_{2+\gamma}(x) \end{aligned}$$

and by (4.4)

$$(4.18) \quad P_N \tilde{y} = \frac{1}{2} P_N \tilde{x} + O(\|P_X \tilde{x}\|) + O(\varepsilon) = \beta_1(x).$$

Similarly

$$(4.19) \quad P_X \tilde{z} = O(\|P_N \tilde{y}\| \|P_2 \tilde{y}\|) + O(\varepsilon \|P_N \tilde{y}\|) + \beta_3(x) = \beta_{2+\gamma}(x)$$

and

$$(4.20) \quad \begin{aligned} P_N \tilde{z} &= \frac{1}{2} P_N \tilde{y} + O(\|P_X \tilde{y}\|) + O(\varepsilon) + \beta_2(x) \\ &= \frac{1}{2} P_N \tilde{y} + \beta_{1+\gamma}(x) = \frac{1}{4} P_N \tilde{x} + \theta P_N \beta_1(x) + P_N \beta_{1+\gamma}(x). \end{aligned}$$

Hence, if  $w = z - \delta(z)$

$$(4.21) \quad \begin{aligned} \delta(z) &= \tilde{z} - \tilde{w} = \frac{1}{2} P_N \tilde{z} + P_N \beta_{1+\gamma}(x) + \beta_{2+\gamma}(x) \\ &= \frac{1}{8} P_N \tilde{x} + \theta P_N \beta_1(x) + P_N \beta_{1+\gamma}(x) + \beta_{2+\gamma}(x). \end{aligned}$$

Thus

$$(4.22) \quad \begin{aligned} \tilde{x}_a &= C \|\delta(z)\|^\alpha \delta(z) + \tilde{z} - 2\delta(z) \\ &= C \|\delta(z)\|^\alpha \delta(z) + P_N \beta_{1+\gamma}(z) + \beta_{2+\gamma}(x). \end{aligned}$$

Hence, since  $\gamma > \alpha$ , for  $\rho$  sufficiently small, say  $\rho < \rho_a$  (4.16) holds and so  $x_a \in W_s(\rho_a, \theta_a)$ . (Here  $\theta_a$  may be chosen to be the value  $\theta(\bar{\varepsilon})$  given by Theorem 3.21.) Moreover,  $\tilde{w} = \frac{1}{16} P_N \tilde{x} + \theta P_N \beta_1(x) + \beta_{1+\gamma}(x)$  and hence, for  $\rho_a$  sufficiently small,  $\|\tilde{w}\| > \|\tilde{x}_a\|$  and acceleration is an improvement. This completes the proof of the Lemma.

Now in Theorem 3.26  $\varepsilon$  is fixed, but the above lemma may be applied if we can show the appropriate relationship between  $x$  and  $\varepsilon$  is satisfied whenever acceleration is employed. That is, we must show the existence of a constant  $J_0$ , independent of  $\varepsilon$ , such that whenever  $x \in W_s(\rho_a, \theta_a)$  and the tests (3.27) are satisfied  $\varepsilon \leq K_0 \|x\|^{1+\gamma/2}$  (and hence  $\varepsilon < \beta_{1+\alpha}(x)$ ). Using the definitions  $\tilde{w} = \tilde{z} - \delta(z)$ ,  $\tilde{z} = \tilde{y} - \delta(y)$  and the conclusion (4.14), which implies  $P_X \tilde{y}$ ,  $P_X \tilde{z}$ ,  $P_X \tilde{w}$  are all  $\beta_2(x)$ , we have

$$(4.23) \quad \begin{aligned} \delta(z) &= P_N \tilde{z} - P_N \tilde{w} + \beta_2(x), \\ \delta(y) &= P_N \tilde{y} - P_N \tilde{z} + \beta_2(x). \end{aligned}$$

Now from (4.2) with  $x, y$  replaced by  $y, z$  we see

$$(4.24) \quad P_N \tilde{z} = \frac{1}{2} P_N \tilde{y} - \frac{1}{2} \frac{\varepsilon \bar{\lambda} \mu(y)}{\varepsilon + \bar{\lambda} \mu(y)} + \beta_2(x)$$

where we have used  $C(y) \tilde{y} = \beta_3(x)$  from (4.14). In the same fashion

$$(4.25) \quad P_N \tilde{w} = \frac{1}{2} P_N \tilde{z} - \frac{1}{2} \frac{\varepsilon \bar{\lambda} \mu(z)}{\varepsilon + \bar{\lambda} \mu(z)} + \beta_2(x).$$

Hence

$$(4.26) \quad \delta(y) - 2\delta(z) = \frac{1}{2} P_N \tilde{y} + \frac{1}{2} \frac{\varepsilon \bar{\lambda} \mu(y)}{\varepsilon + \bar{\lambda} \mu(y)} - P_N \tilde{z} - \frac{\varepsilon \bar{\lambda} \mu(z)}{\varepsilon + \bar{\lambda} \mu(z)} + \beta_2(x).$$

But from (4.24) we see (4.26) becomes

$$(4.27) \quad \delta(y) - 2\delta(z) = \frac{\varepsilon \bar{\lambda} \mu(y)}{\varepsilon + \bar{\lambda} \mu(y)} - \frac{\varepsilon \bar{\lambda} \mu(z)}{\varepsilon + \bar{\lambda} \mu(z)} + \beta_2(x).$$

From (3.12)  $\bar{\lambda}(y) = \lambda + \beta_2(x)$ ,  $\bar{\lambda}(z) = \lambda + \beta_2(y) = \lambda + \beta_2(x)$ , and from (3.23)  $\mu(z) \leq \frac{3}{4} \mu(y)$ . Hence

$$(4.28) \quad \begin{aligned} \|\delta(y) - 2\delta(z)\| &= \frac{\varepsilon^2 \lambda (\mu(y) - \mu(z))}{(\varepsilon + \lambda \mu(y))(\varepsilon + \lambda \mu(z))} + \beta_2(x) \\ &\leq \frac{\frac{1}{4} \varepsilon^2 \lambda \mu(y)}{(\varepsilon + \lambda \mu(y))^2} + \beta_2(x). \end{aligned}$$

Then the test (3.27a) provides a constant  $k$  such that

$$(4.29) \quad \frac{\varepsilon^2 \mu(y)}{(\varepsilon + \lambda \mu(y))^2} \leq k(\mu(y)^{1+\gamma} + \mu(x)^2).$$

Now  $\delta(x) = \tilde{x} - \tilde{y} = P_N \tilde{x} - P_N \tilde{y} + P_X \tilde{x} - P_X \tilde{y}$ , and  $P_X \tilde{y} = \beta_2(x)$  and so  $\|P_N \tilde{x} - P_N \tilde{y}\| - \|P_X \tilde{x}\| \leq \|\delta(x)\| + \beta_2(x) \leq \|P_N \tilde{x} - P_N \tilde{y}\| + \|P_X \tilde{x}\|$ . From the expressions (4.4)-(4.7)

we see

$$(4.30) \quad \left( \frac{\varepsilon + \bar{\lambda}\mu(x)/4}{\varepsilon + \bar{\lambda}\mu(x)} \right) \|P_N \tilde{x}\| - \|P_X \tilde{x}\| \leq \|\delta(x)\| + \beta_2(x) \leq \left( \frac{\varepsilon + 3\bar{\lambda}\mu(x)/4}{\varepsilon + \bar{\lambda}\mu(x)} \right) \|P_N \tilde{x}\| + \|P_X \tilde{x}\|.$$

Now since  $\|P_X \tilde{x}\| \leq \theta \|P_N \tilde{x}\|$  we have  $c_0\mu(x) \leq \|\delta(x)\| \leq c_1\mu(x)$  for some constants  $c_0, c_1$  and the same is true with  $x$  replaced by  $y$ . Hence the test (3.27b) assures  $\mu(x) \leq c_2\mu(y)$  for some constant  $c_2$  and the relationship (4.29) may be replaced by

$$(4.31) \quad \frac{\varepsilon^2 \mu(y)}{(\varepsilon + \lambda\mu(y))^2} \leq K\mu(y)^{1+\gamma}$$

for some constant  $K$ , which in turn implies

$$(4.32) \quad \varepsilon(1 - K\mu(y)^{\gamma/2}) \leq K\mu(y)^{1+\gamma/2} = \beta_{1+\gamma/2}(y) = \beta_{1+\gamma/2}(x)$$

and hence  $\varepsilon \leq K_0 \|\tilde{x}\|^{1+\gamma/2}$  for some constant  $K_0$  and Lemma 4.15 may be applied. This completes the proof.  $\square$

**5. Examples.** We define  $F: \mathbb{R}^2 \rightarrow \mathbb{R}^2$  by

$$(5.1) \quad F \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} (x-1) + (y-3)^2 \\ \varepsilon(y-3) + \frac{3}{2}(x-1)(y-3) + (y-3)^2 + (y-3)^3 \end{pmatrix}.$$

Here

$$(5.2) \quad P_{N_0} = \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix}, \quad P_{X_0} = \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix}.$$

There are two roots near  $(1, 3)^T$ ,

$$(5.3) \quad \mathbf{x}_+^* = (1, 3)^T, \quad \mathbf{x}_-^* = (1 - \eta^2, 3 + \eta)$$

where  $\eta = 1 - \sqrt{1 + 2\varepsilon}$ . For this situation the region  $W_+(\rho, \theta)$  will correspond to the root,  $\mathbf{x}_+^*$ . Here,  $P_N = P_{N_0}$ ,  $P_X = P_{X_0}$  and

$$(5.4) \quad W_{\pm}(\rho, \theta) = \left\{ \begin{pmatrix} x \\ y \end{pmatrix} \mid 0 < |x-1| + |y-3| \leq \rho, \pm(y-3) > 0, |x-1| \leq \theta|y-3| \right\}.$$

For an initial guess  $x_0 \in W_{\pm}(\rho, \theta)$ , where  $\rho$  and  $\theta$  satisfy the requirements of Theorem 3.22, the Newton iterates will converge to  $\mathbf{x}_{\pm}^*$ . If the acceleration method of [16] is employed, however, the convergence behavior of the Newton iterates is irregular and initial guesses in  $W_{\pm}$  may give rise to iterates converging to either of  $\mathbf{x}_+^*$  or  $\mathbf{x}_-^*$ . Note that the radius of the ball about  $\mathbf{x}_{\pm}^*$  for which quadratic convergence is assured is less than  $2\varepsilon$  for this example since  $\|\mathbf{x}_{\pm}^* - \mathbf{x}^*\| \leq 2\varepsilon$ . In our calculations an initial guess of  $x_0 = .9$  and  $y_0 = 3.3$  was used;  $\varepsilon$  varied from  $10^{-4}$  to  $10^{-7}$ ,  $\alpha$  varied from .05 to .47;  $\gamma$  was taken to be the midpoint of the range of allowable choices,  $(2\alpha + 1)/2$ ;  $C$  varied from .5 to 3.5. The iteration was terminated when  $|x_n - x_{\pm}^*| < 10^{-7}$ .

Table 1 gives, for various values of  $\varepsilon$ , the number of iterates  $n_0$  required to obtain termination for Newton's method, the values of  $\alpha$  and  $C$  that gave the best performance for the modified scheme, the number,  $n_1$ , of iterates required for termination of the modified scheme, the first iterate,  $n_2$ , for which the test was passed, and the number of times,  $n_3$ , the test was passed.



TABLE 1

$\epsilon$	$\alpha$	$c$	$n_0$	$n_1$	$n_2$	$n_3$
$10^{-4}$	.4	.5	12	8	5	1
$10^{-5}$	.3	1.0	15	7	4	2
$10^{-6}$	.35	.5	17	7	4	2
$10^{-7}$	.4	.5	19	8	5	2

In Table 2 we give  $n_1$  for  $\epsilon = 10^{-6}$  and several other values of  $C$  and  $\alpha$  that were considered.

TABLE 2  
( $\epsilon = 10^{-6}$ ).

$C$	$\alpha$	$n_1$	$C$	$\alpha$	$n_1$
.5	.1	13	2.0	.1	16
.5	.4	13	2.0	.35	12
.5	.47	14	2.0	.47	12
1.0	.1	16	3.0	.1	19
1.0	.35	11	3.0	.35	13
1.0	.47	13	3.0	.47	11
1.5	.1	16	3.5	.1	20
1.5	.35	11	3.5	.35	14
1.5	.4	8	3.5	.47	12
1.5	.47	12			

Our second example is a discrete approximation to the Chandrasekhar  $H$ -equation [1]. This satisfies our hypotheses [14], [15], [17].

$$(5.5) \quad H(x) - \left( 1 - \frac{c}{2} \int_0^1 \frac{x}{x+t} H(t) dt \right)^{-1} = F(H) = 0.$$

We approximated the integral by eight point Gaussian quadratures. Equation (5.5) and the approximate equation are singular at  $c = 1$  [17]. When  $c = 1$  the null space of  $F'(H)$  is spanned by  $\phi = xH(x)$  and, for  $f \in L^1[0, 1]$ ,

$$(5.6) \quad P_N f(x) = 2 \int_0^1 H^{-1}(t) f(t) dt \phi(x).$$

For  $c \in (0, 1)$ , let  $L$  denote the integral operator in (5.5)

$$(5.7) \quad Lf(x) = \frac{c}{2} \int_0^1 \frac{x}{x+t} f(t) dt.$$

It is known that  $F'(H)$  is Fredholm of index zero for  $c \in (0, 1)$ , [17], and hence, for  $c$  sufficiently near one, there is a unique eigenvalue  $\epsilon$ , least in absolute value, having algebraic and geometric multiplicity one. We let  $u$  denote an eigenfunction corresponding to the eigenvalue  $\epsilon$ , normalized so that  $\int_0^1 u(x) dx = 1$ , and use (5.5) to obtain, as in [5],

$$(5.8) \quad F'(H)u = u - H^2 Lu = \epsilon u.$$

Now for  $c \in (0, 1)$ , (5.5) has two solutions, only one of which is of physical importance [17]. This solution, which we denote by  $H$ , is characterized by [14], [17],

$$(5.9) \quad \int_0^1 H(x) dx = \frac{2}{c}(1 - \sqrt{1-c}).$$

We set  $u = Hv$  in (5.8), divide by  $H$ , and integrate to obtain

$$(5.10) \quad \begin{aligned} (1 - \varepsilon) \int_0^1 v(x) dx &= \frac{c}{2} \int_0^1 H(x) \int_0^1 \frac{x}{x+t} H(t)v(t) dt dx \\ &= \frac{c}{2} \int_0^1 H(t)v(t) \left[ \int_0^1 H(x) dx - \int_0^1 \frac{t}{x+t} H(x) dx \right] dt \\ &= \int_0^1 H(t)v(t) [(1 - \sqrt{1-c}) + (H^{-1}(t) - 1)] dt \\ &= \int_0^1 H(t)v(t)(H^{-1}(t) - \sqrt{1-c}) dt \\ &= \int_0^1 v(t) dt - \sqrt{1-c} \int_0^1 u(t) dt. \end{aligned}$$

The normalization of  $u$  then implies that

$$(5.11) \quad \varepsilon \int_0^1 v(x) dx = \sqrt{1-c}.$$

Hence  $\varepsilon$  and  $\sqrt{1-c}$  are of the same order.

In order to compute  $\lambda$ , we set  $\varepsilon = 0$ , then, as in [5]

$$(5.12) \quad F''(H)(\phi, \phi) = -2x^2H(x).$$

Hence, by (5.6),

$$(5.13) \quad P_N F''(H)(\phi, \phi) = \left( -4 \int_0^1 x^2 dx \right) \phi = -\frac{4}{3} \phi.$$

Therefore, for  $\varepsilon$  sufficiently small,  $\lambda < 0$ . Hence our initial iterate should lie in  $W_-$ . This is not surprising in view of the fact that the second solution to (5.5) is pointwise greater than  $H$  [17]. We have observed that if the initial iterate is larger than  $H$  then convergence can be to the nonphysical solution.

In Tables 3 and 4 we present data for (5.3) in a format identical to Tables 1 and 2, except that we vary  $\sqrt{1-c}$  rather than  $\varepsilon$ . In each case the norm used was the  $L^1$  norm. The initial guess was chosen as  $H_0(x) \equiv 1$ . This choice was made since  $H(x)$  satisfies  $H(x) > H_0(x)$  for  $x > 0$  and hence from (5.6) we have  $P_N(H_0 - H) = \mu\phi$  with sign ( $\mu$ ) negative, placing  $H_0(x)$  in  $W_-(\rho, \theta)$  for some  $\rho$  and  $\theta$ .

TABLE 3

$\sqrt{1-c}$	$\alpha$	$c$	$n_0$	$n_1$	$n_2$	$n_3$
$10^{-4}$	.45	1.0	15	8	3	2
$10^{-5}$	.45, .46	.5	18	9	3	2
$10^{-6}$	.45, .46,	.5	21	11	3	2
	.47, .35	(1.0 for $\alpha = 35$ )				(3 for $\alpha = .35$ )
$10^{-7}$	.45, .46	1.0	23	10	3	3

TABLE 4  
 $(\sqrt{1-c} = 10^{-6})$ .

$C$	$\alpha$	$n_1$	$C$	$\alpha$	$n_1$
.5	.35	13	1.5	.4	13
.5	.4	12	1.5	.47	16
1.0	.4	16	2.0	.4	13
1.0	.45	15	2.0	.47	12
1.0	.46	15	3.0	.4	17
1.0	.47	15	3.0	.47	15
1.5	.35	14	3.5	.4	18
			3.5	.47	16

From (4.22) we see that the dominant term in  $\tilde{x}_a$  is of order  $C\|\delta(z)\|^{1+\alpha}$ . From this one might expect that best performance would be achieved for very small  $C$  and  $\alpha$  very near  $\frac{1}{2}$ . Our tables, in fact, indicate that the smaller values of  $C$  give more rapid convergence. However, if  $C$  is too small the other terms in (4.22) may become dominant. The effect of this will be that  $x_a$  may well leave  $W_s$  and the iterates converge to a solution not in  $W_s$ . This was observed in both of our examples. When  $C$  was on the order of .1 quite rapid convergence to the nonphysical solution of the  $H$ -equation was observed for several values of  $\alpha$ . Values of  $C$  larger than .5 always gave iterates that remained in  $W_s$ . Values of 2.0 or larger tended to slow down the convergence.

The proper choice of the parameter  $\alpha$  is in general the largest allowable value for which the test, (3.27a), can be passed at an early state in the iteration and passed sufficiently many times. Values of  $\alpha$  that are too large make the test difficult to pass and hence many unmodified iterates might be needed before a modified step is allowed. This problem is especially pronounced in the first example. If  $\alpha$  is too small, convergence may be slowed.

The combination  $C = .5, \alpha = .4$  gave reasonable results in both examples.

Other rules for choosing  $\gamma$  were tried with no qualitative changes in the results.

REFERENCES

[1] S. CHANDRASEKHAR, *Radiative Transfer*, Dover, New York, 1960.  
 [2] M. G. CRANDALL AND P. H. RABINOWITZ, *Bifurcation, perturbation of simple eigenvalues, and linearized stability*, Arch. Rat. Mech. Anal., 52 (1973), pp. 161-180.  
 [3] D. W. DECKER AND C. T. KELLEY, *Newton's method at singular points I*, SIAM J. Numer. Anal., 17 (1980), pp. 66-70.  
 [4] ———, *Newton's method at singular points II*, SIAM J. Numer. Anal., 17 (1980), pp. 465-471.  
 [5] ———, *Convergence acceleration for Newton's method at singular points*, SIAM J. Numer. Anal., 19 (1982), pp. 219-229.  
 [6] D. W. DECKER, H. B. KELLER AND C. T. KELLEY, *Convergence rates for Newton's method at singular points*, SIAM J. Numer. Anal., 20 (1983), pp. 296-314.  
 [7] A. O. GRIEWANK, *Analysis and modification of Newton's method at singularities*, Thesis, Australian National University, Canberra, 1980.  
 [8] ———, *Starlike domains of convergence for Newton's method at singularities*, Numer. Math., 35 (1980), pp. 95-111.  
 [9] ———, *On solving nonlinear equations with simple singularities or nearly singular solutions*, submitted for publication.  
 [10] A. O. GRIEWANK AND M. R. OSBORNE, *Newton's method for singular problems when the dimension of the null space is  $>1$* , SIAM J. Numer. Anal., 18 (1981), pp. 179-189.  
 [11] ———, *Analysis of Newton's method at irregular singular points*, SIAM J. Numer. Anal., 20 (1983), pp. 747-773.

- [12] L. V. KANTOROVICH AND G. P. AKILOV, *Functional Analysis in Normed Spaces*, Pergamon, New York, 1964.
- [13] 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. 359–384.
- [14] C. T. KELLEY, *Solution of the Chandrasekhar H-equation by Newton's method*, *J. Math. Phys.*, 19 (1978), pp. 500–501.
- [15] C. T. KELLEY, *Approximate methods for the solution of the Chandrasekhar H-equation*, *J. Math. Phys.*, 23 (1982), pp. 2097–2100.
- [16] C. T. KELLEY AND R. SURESH, *A new acceleration method for Newton's method at singular points*, *SIAM J. Numer. Anal.*, 20 (1983), pp. 1001–1009.
- [17] T. W. MULLIKIN, *Some probability distributions for neutron transport in a half-space*, *J. Appl. Prob.*, 5 (1968), pp. 357–374.
- [18] L. B. RALL, *Convergence of the Newton process of multiple solutions*, *Numer. Math.*, 9 (1961), pp. 23–37.
- [19] G. W. REDDIEN, *On Newton's method for singular problems*, *SIAM J. Numer. Anal.*, 15 (1978), pp. 993–996.
- [20] ———, *Newton's method and high order singularities*, *Comput. Math. Appl.*, 5 (1980), pp. 79–86.
- [21] R. B. SCHNABEL, *Conic methods for unconstrained minimization problems and tensor methods for nonlinear equations*, Univ. Colorado Technical Report, Dept. Computer Science, Univ. Colorado, Boulder, 1982.
- [22] I. STAKGOLD, *Branching of solutions of nonlinear equations*, *SIAM Rev.*, 13 (1971), pp. 289–332.

## SHAPE PRESERVING PIECEWISE RATIONAL INTERPOLATION\*

R. DELBOURGO† AND J. A. GREGORY‡

**Abstract.** An explicit representation of a  $C^1$  piecewise rational cubic function is developed which can be used to solve the problem of shape preserving interpolation. It is shown that the interpolation method can be applied to convex and/or monotonic sets of data and an error analysis of the interpolant is given. The scheme includes, as a special case, the monotonic rational quadratic interpolant considered by the authors in [1] and [5]. However, the requirement of convexity necessitates the generalization to the rational cubic form employed here.

**Key words.** rational, shape preserving, monotonic, convex, interpolation

**1. Introduction.** The problem of shape preserving interpolation has been considered by a number of authors. Fritsch and Carlson [4] and Fritsch and Butland [3] have discussed the piecewise cubic interpolation of monotonic data. Also, McAllister, Passow and Roulier [6] and Passow and Roulier [9] consider the piecewise polynomial interpolation of monotonic and convex data. In particular, an algorithm for quadratic spline interpolation is given in McAllister and Roulier [7]. An alternative to the use of polynomials, for the interpolation of monotonic data, is the application of piecewise rational quadratic functions, as described by the authors in references [1] and [5].

In this paper we describe a piecewise rational cubic function which can be used to solve the problem of shape preserving interpolation. The rational cubic includes the rational quadratic function as a special case. However, the rational quadratic is not necessarily applicable to the interpolation of convex data and this necessitates the generalization to the rational cubic form employed here.

The paper begins with a definition and error analysis of the rational cubic interpolant. The application of the interpolant to monotonic and/or convex sets of data is then discussed in § 3. It is shown that  $O(h^4)$  error bounds can be expected when exact derivative information is given at the data points. For the case where the derivatives are not known, these have to be estimated and various schemes for this are considered. Finally, in § 4, examples of the rational interpolants applied with various derivative schemes are given.

**2. The rational cubic interpolant.** Let  $(x_i, f_i)$   $i = 1, \dots, n$  be a given set of data points, where  $x_1 < x_2 < \dots < x_n$ . Let

$$(2.1) \quad \begin{aligned} h_i &= x_{i+1} - x_i, \\ \Delta_i &= (f_{i+1} - f_i) / h_i. \end{aligned}$$

A piecewise rational cubic function  $s \in C^1[x_1, x_n]$  is defined as follows. For  $x \in [x_i, x_{i+1}]$  let

$$(2.2) \quad \theta = (x - x_i) / h_i.$$

Then

$$(2.3) \quad s(x) = P_i(\theta) / Q_i(\theta),$$

where

$$(2.4) \quad P_i(\theta) = f_{i+1}\theta^3 + (r_i f_{i+1} - h_i d_{i+1})\theta^2(1 - \theta) + (r_i f_i + h_i d_i)\theta(1 - \theta)^2 + f_i(1 - \theta)^3,$$

\* Received by the editors November 23, 1983, and in revised form June 14, 1984.

† Faculty of Engineering, Science and Mathematics, Middlesex Polytechnic, London, England.

‡ Department of Mathematics and Statistics, Brunel University, Uxbridge, England.

$$(2.5) \quad \begin{aligned} Q_i(\theta) &= \theta^3 + r_i[\theta^2(1-\theta) + \theta(1-\theta)^2] + (1-\theta)^3 \\ &= 1 + (r_i - 3)\theta(1-\theta). \end{aligned}$$

The rational cubic has the following interpolatory properties

$$(2.6) \quad \begin{aligned} s(x_i) &= f_i, & s(x_{i+1}) &= f_{i+1}, \\ s^{(1)}(x_i) &= d_i, & s^{(1)}(x_{i+1}) &= d_{i+1}, \end{aligned}$$

where  $s^{(1)}$  denotes differentiation with respect to  $x$  and the  $d_i$  denote derivative values given at the knots  $x_i$ .

The parameter  $r_i$  is to be chosen such that

$$(2.7) \quad r_i > -1$$

which ensures a strictly positive denominator in the rational cubic. When  $r_i = 3$  the rational cubic clearly reduces to the standard cubic Hermite polynomial. For our purposes  $r_i$  will be chosen to ensure that the interpolant preserves the monotonic or convex shape of the data. This choice requires a knowledge of  $s^{(1)}(x)$  and  $s^{(2)}(x)$  which are given in the relevant sections below.

*Remark.* It should be noted that the interpolant will define a nonlinear operator, since the  $r_i$  will be dependent on the data. However the interpolant to the zero function is zero. Also, the interpolant to the data  $K + f_i, i = 1, \dots, n$ , where  $K$  is a constant, is  $K + s(x)$ , provided the  $r_i$  are independent of such translations. This will be the case for the choices of  $r_i$  in this paper.

An error bound for the rational cubic is given by the following theorem.

**THEOREM 2.1.** *Let  $f \in C^4[x_1, x_n]$  and let  $s$  be the piecewise rational cubic interpolant such that  $s(x_i) = f(x_i)$  and  $s^{(1)}(x_i) = d_i, i = 1, \dots, n$ . Then for  $x \in [x_i, x_{i+1}]$*

$$(2.8) \quad \begin{aligned} |f(x) - s(x)| &\leq \frac{h_i}{4c_i} \max \{ |f_i^{(1)} - d_i|, |f_{i+1}^{(1)} - d_{i+1}| \} \\ &+ \frac{1}{384c_i} \{ h_i^4 \|f^{(4)}\| (1 + |r_i - 3|/4) + 4|r_i - 3| (h_i^3 \|f^{(3)}\| + 3h_i^2 \|f^{(2)}\|) \}, \end{aligned}$$

where

$$(2.9) \quad c_i = \begin{cases} (1 + r_i)/4 & \text{if } -1 < r_i < 3, \\ 1 & \text{if } r_i \geq 3, \end{cases}$$

and  $\| \cdot \|$  denotes the uniform norm on  $[x_i, x_{i+1}]$ .

*Proof.* On  $[x_i, x_{i+1}]$  let  $x(\theta) = x_i + \theta h_i$  and  $F_i(\theta) = f(x(\theta))$ . Then

$$f(x) - s(x) = F_i(\theta) - P_i(\theta)/Q_i(\theta)$$

where  $0 \leq \theta \leq 1$ . Consider

$$(2.10) \quad |F_i(\theta) - P_i(\theta)/Q_i(\theta)| \leq [ |F_i(\theta)Q_i(\theta) - P_i^*(\theta)| + |P_i^*(\theta) - P_i(\theta)| ] / |Q_i(\theta)|,$$

where (cf. 2.4)

$$(2.11) \quad P_i^*(\theta) = f_{i+1}\theta^3 + (r_i f_{i+1} - h_i f_{i+1}^{(1)})\theta^2(1-\theta) + (r_i f_i + h_i f_i^{(1)})\theta(1-\theta)^2 + f_i(1-\theta)^3.$$

Then  $P_i^*(\theta)$  is the cubic Hermite interpolant to  $F_i(\theta)Q_i(\theta)$  on  $0 \leq \theta \leq 1$  and bounding

the Cauchy remainder of the interpolant gives

$$\begin{aligned} |F_i(\theta)Q_i(\theta) - P_i^*(\theta)| &\leq \frac{1}{384} \max_{0 \leq \theta \leq 1} \left| \frac{d^4}{d\theta^4} F_i(\theta)Q_i(\theta) \right| \\ &= \frac{1}{384} \max_{0 \leq \theta \leq 1} |F_i^{(4)}(\theta)Q_i(\theta) + 4F_i^{(3)}(\theta)Q_i^{(1)}(\theta) + 6F_i^{(2)}(\theta)Q_i^{(2)}(\theta)| \end{aligned}$$

since  $Q_i(\theta)$  is quadratic. Now

$$|Q_i(\theta)| \leq 1 + |r_i - 3|/4, \quad |Q_i^{(1)}(\theta)| \leq |r_i - 3|, \quad |Q_i^{(2)}(\theta)| = 2|r_i - 3|$$

and

$$F_i^{(j)}(\theta) \leq h_i^j \|f^{(j)}\|.$$

Hence

$$\begin{aligned} (2.12) \quad |F_i(\theta)Q_i(\theta) - P_i^*(\theta)| &\leq \frac{1}{384} \{h_i^4 \|f^{(4)}\| (1 + |r_i - 3|/4) \\ &\quad + 4h_i^3 \|f^{(3)}\| |r_i - 3| + 12h_i^2 \|f^{(2)}\| |r_i - 3|\}. \end{aligned}$$

Also

$$\begin{aligned} (2.13) \quad |P_i^*(\theta) - P_i(\theta)| &= |\theta(1 - \theta)h_i[\theta(d_{i+1} - f_{i+1}^{(1)}) + (1 - \theta)(f_i^{(1)} - d_i)]|, \\ &\leq \frac{1}{4} h_i \max \{|f_i^{(1)} - d_i|, |f_{i+1}^{(1)} - d_{i+1}|\}. \end{aligned}$$

Finally

$$(2.14) \quad |Q_i(\theta)| = Q_i(\theta) \geq \begin{cases} 1 & \text{if } r_i \geq 3, \\ 1 - (3 - r_i)/4 & \text{if } -1 < r_i < 3. \end{cases}$$

Combining (2.12), (2.13) and (2.14) in inequality (2.10) completes the proof of the theorem.

A direct consequence of Theorem 2.1 which is of relevance in the remaining sections is the following corollary.

**COROLLARY 2.1.** *Let  $x \in [x_i, x_{i+1}]$ .*

(i) *If  $d_i - f_i^{(1)} = O(h_i^2) = d_{i+1} - f_{i+1}^{(1)}$  and  $r_i - 3 = O(h_i)$  then  $|f(x) - s(x)| = O(h_i^3)$ .*

(ii) *If  $d_i - f_i^{(1)} = O(h_i^3) = d_{i+1} - f_{i+1}^{(1)}$  and  $r_i - 3 = O(h_i^2)$  then  $|f(x) - s(x)| = O(h_i^4)$ .*

The above theorem and corollary show that  $r_i$  should ideally be such that  $r_i - 3 = O(h^2)$ . We now consider how  $r_i$  can be chosen to preserve the monotonic or convex shape of the data, whilst maintaining this optimal  $O(h^2)$  requirement.

### 3. Shape preserving interpolation.

**3.1. Monotonic data.** For simplicity of presentation, we assume a monotonic increasing set of data so that

$$(3.1) \quad f_1 \leq f_2 \leq \dots \leq f_m$$

or equivalently

$$(3.2) \quad \Delta_i \geq 0, \quad i = 1, \dots, n-1.$$

(The case of a monotonic decreasing set of data can be treated in a similar manner.) For a monotonic interpolant  $s(x)$ , it is then necessary that the derivative parameters

should be such that

$$(3.3) \quad d_i \geq 0, \quad i = 1, \dots, n.$$

Now  $s(x)$  is monotonic increasing if and only if

$$(3.4) \quad s^{(1)}(x) \geq 0$$

for all  $x \in [x_1, x_n]$ . For  $x \in [x_i, x_{i+1}]$  it can be shown, after some simplification, that

$$(3.5) \quad s^{(1)}(x) = \frac{d_{i+1}\theta^4 + \alpha_i\theta^3(1-\theta) + \beta_i\theta^2(1-\theta)^2 + \gamma_i\theta(1-\theta)^3 + d_i(1-\theta)^4}{[1 + (r_i - 3)\theta(1-\theta)]^2},$$

where

$$(3.6) \quad \begin{aligned} \alpha_i &= 2(r_i\Delta_i - d_i), \\ \beta_i &= (r_i^2 + 3)\Delta_i - r_i(d_i + d_{i+1}), \\ \gamma_i &= 2(r_i\Delta_i - d_{i+1}). \end{aligned}$$

Thus sufficient conditions for monotonicity on  $[x_i, x_{i+1}]$  are

$$(3.7) \quad \alpha_i \geq 0, \quad \beta_i \geq 0, \quad \gamma_i \geq 0$$

where the necessary conditions  $d_i \geq 0$  and  $d_{i+1} \geq 0$  are assumed.

If  $\Delta_i > 0$  (strict inequality) then a sufficient condition for (3.7) is

$$(3.8) \quad r_i \geq (d_i + d_{i+1})/\Delta_i$$

In particular, if

$$(3.9) \quad r_i = 1 + (d_i + d_{i+1})/\Delta_i$$

then the rational cubic defined by (2.3)-(2.5) reduces to the rational quadratic form

$$(3.10) \quad s(x) = \frac{f_{i+1}\theta^2 + \Delta_i^{-1}(f_{i+1}d_i + f_id_{i+1})\theta(1-\theta) + f_i(1-\theta)^2}{\theta^2 + \Delta_i^{-1}(d_i + d_{i+1})\theta(1-\theta) + (1-\theta)^2},$$

for which  $d_i \geq 0$  and  $d_{i+1} \geq 0$  are necessary and sufficient conditions for a monotonic increasing interpolant. It should be noted that if  $\Delta_i = 0$ , then  $d_i = d_{i+1} = 0$  and

$$(3.11) \quad s(x) = f_i = f_{i+1}$$

is a constant on  $[x_i, x_{i+1}]$ .

The rational quadratic form (3.10) has been investigated in detail elsewhere by the authors, see references [1] and [5]. It is worth remarking that (3.9) gives  $r_i - 3 = (d_i + d_{i+1} - 2\Delta_i)/\Delta_i$  and it can then be shown that

$$r_i - 3 = (d_i - f_i^{(1)} + d_{i+1} - f_{i+1}^{(1)})/\Delta_i + O(h_i^2).$$

Thus, Theorem 2.1 and its corollary show that (3.9) is a good choice for  $r_i$ , since the optimal  $O(h^4)$  bound on the interpolation error can be achieved if  $d_i$  and  $d_{i+1}$  are chosen with  $O(h^3)$  accuracy.

**3.2. Convex data.** We assume a strictly convex set of data so that

$$(3.12) \quad \Delta_1 < \Delta_2 < \dots < \Delta_{n-1}.$$

(The case of concave data, where the inequalities are reversed, can be treated in a similar way.) To have a convex interpolant  $s(x)$ , and to avoid the possibility of  $s(x)$



having straight line segments, it is necessary that the derivative parameters should satisfy

$$(3.13) \quad d_1 < \Delta_1 < d_2 < \cdots < \Delta_{i-1} < d_i < \Delta_i < \cdots < d_n$$

Now  $s(x)$  is convex if and only if

$$(3.14) \quad s^{(2)}(x) \geq 0$$

for all  $x \in [x_i, x_{i+1}]$ . After some simplification, it can be shown that for  $x \in [x_i, x_{i+1}]$

$$(3.15) \quad s^{(2)}(x) = \frac{(2/h_i)[\alpha_i\theta^3 + \beta_i\theta^2(1-\theta) + \gamma_i\theta(1-\theta)^2 + \delta_i(1-\theta)^3]}{[1 + (r_i - 3)\theta(1-\theta)]^3}$$

where

$$(3.16) \quad \begin{aligned} \alpha_i &= r_i(d_{i+1} - \Delta_i) - d_{i+1} + d_i, \\ \beta_i &= 3(d_{i+1} - \Delta_i), \\ \gamma_i &= 3(\Delta_i - d_i), \\ \delta_i &= r_i(\Delta_i - d_i) - d_{i+1} + d_i. \end{aligned}$$

Hence, from (3.15), necessary conditions for convexity are

$$(3.17) \quad \alpha_i \geq 0 \quad \text{and} \quad \delta_i \geq 0.$$

These conditions, together with inequalities (3.13), are also sufficient since we have

$$\beta_i > 0 \quad \text{and} \quad \gamma_i > 0$$

in (3.15). Thus, from (3.17), we have the condition that the interpolant is convex if and only if

$$(3.18) \quad r_i \geq \max \left\{ \frac{d_{i+1} - d_i}{d_{i+1} - \Delta_i}, \frac{d_{i+1} - d_i}{\Delta_i - d_i} \right\} = 1 + M_i/m_i,$$

where

$$(3.19) \quad \begin{aligned} M_i &= \max \{d_{i+1} - \Delta_i, \Delta_i - d_i\}, \\ m_i &= \min \{d_{i+1} - \Delta_i, \Delta_i - d_i\}, \end{aligned}$$

and the necessary conditions (3.13) are assumed.

We have found two choices of  $r_i$  which satisfy (3.18) and produce pleasing graphical results. These are

$$(3.20) \quad r_i = 2 + M_i/m_i,$$

$$(3.21) \quad \begin{aligned} r_i &= 3 + (M_i/m_i - 1)^2 / (M_i/m_i) \\ &= 1 + M_i/m_i + m_i/M_i \\ &= 1 + (d_{i+1} - \Delta_i) / (\Delta_i - d_i) + (\Delta_i - d_i) / (d_{i+1} - \Delta_i), \end{aligned}$$

the latter being the smaller value. Their use is justified by Theorem 2.1 and its corollary as follows. Suppose  $d_i - f_i^{(1)} = O(h_i^2)$  and  $d_{i+1} - f_{i+1}^{(1)} = O(h_i^2)$ . Then it can be shown that  $M_i/m_i = 1 + O(h_i)$ . Thus  $r_i - 3 = O(h_i)$  for (3.20) and  $r_i - 3 = O(h_i^2)$  for (3.21). In practice, therefore, we prefer the use of (3.21), since the optimal  $O(h^4)$  bound on the interpolation error can be achieved if  $O(h^3)$  derivative values are given.

*Remark.* In the above we have assumed strictly convex data. Otherwise, if  $\Delta_i = \Delta_{i+1}$  then on  $[x_i, x_{i+1}]$  we must have  $d_i = d_{i+1} = \Delta_i$ . As would be expected, the rational cubic

then reduces to the straight line segment

$$s(x) = (1 - \theta)f_i + \theta f_{i+1},$$

with an equivalent result on  $[x_{i+1}, x_{i+2}]$ .

**3.3. Convex and monotonic data.** We now consider the possibility that the data satisfy both the monotonic increasing condition (3.1) and the strictly convex condition (3.12). The derivative parameters must then satisfy the inequalities

$$(3.22) \quad 0 \leq d_1 < \Delta_1 < d_2 < \dots < \Delta_{i-1} < d_i < \Delta_i < \dots < d_n.$$

Any convex interpolant must then also be monotonic. This result follows since

$$s^{(1)}(x) = \int_{x_1}^x s^{(2)}(x) dx + s^{(1)}(x_1), = \int_{x_1}^x s^{(2)}(x) dx + d_1.$$

Hence  $d_1 \geq 0$  and the convexity condition  $s^{(2)}(x) \geq 0$  imply that  $s^{(1)}(x) \geq 0$  for  $x \in [x_1, x_n]$ . Thus the convex interpolation method of the previous subsection is also suitable for the interpolation of convex and monotonic data. This result is confirmed by the fact that

$$1 + M_i/m_i \geq (d_i + d_{i+1})/\Delta_i$$

for data satisfying (3.22). Thus the convexity condition (3.18) is sufficient to ensure that the monotonicity condition (3.8) is satisfied.

It should be noted that if the data is convex but not strictly convex, then the interpolant can produce straight line (and hence monotonic) segments, as observed in the previous subsection.

**3.4. Approximations for the derivative parameters.** In most applications, the derivative parameters  $d_i$  will not be given and hence must be determined from the data  $(x_i, f_i)$ ,  $i = 1, \dots, n$ . An obvious choice is the  $O(h^2)$  three point difference approximation

$$(3.23) \quad d_i = (h_i \Delta_{i-1} + h_{i-1} \Delta_i) / (h_{i-1} + h_i), \quad i = 2, \dots, n-1$$

with end conditions

$$(3.24) \quad \begin{aligned} d_1 &= (1 + h_1/h_2)\Delta_1 - (h_1/h_2)\Delta_{3,1}, & \Delta_{3,1} &= (f_3 - f_1)/(x_3 - x_1), \\ d_n &= (1 + h_{n-1}/h_{n-2}) - (h_{n-1}/h_{n-2})\Delta_{n,n-2}, & \Delta_{n,n-2} &= (f_n - f_{n-2})/(x_n - x_{n-2}). \end{aligned}$$

These arithmetic mean approximations are suitable for the convex interpolation problem, since they satisfy inequalities (3.13). However, for the interpolation of monotonic increasing data, (3.24) may give negative results, thus violating the necessary condition (3.3). Also (3.23) does not define a continuous functional on the space of monotonic  $C^1$  functions, since we can have  $\lim d_i \neq 0$  as either  $\lim \Delta_{i-1} = 0$  or  $\lim \Delta_i = 0$ .

Alternative  $O(h^2)$  approximations which avoid the above problems are the geometric means

$$(3.25) \quad d_i = \Delta_{i-1}^{h_i/(h_{i-1}+h_i)} \Delta_i^{h_{i-1}/(h_{i-1}+h_i)}, \quad i = 2, \dots, n-1$$

with end conditions

$$(3.26) \quad \begin{aligned} d_1 &= \Delta_1^{(1+h_1/h_2)} \Delta_{3,1}^{-h_1/h_2}, \\ d_n &= \Delta_{n-1}^{(1+h_{n-1}/h_{n-2})} \Delta_{n,n-2}^{-h_{n-1}/h_{n-2}}. \end{aligned}$$

These approximations, which are discussed in detail in Delbourgo and Gregory [2],

are suitable for the interpolation of monotonic data. Furthermore, if the data is monotonic and convex, then the geometric mean approximations are also appropriate, since they satisfy inequalities (3.22). Reference [2] also considers the use of harmonic mean approximations. However, we do not discuss these here.

The above  $O(h^2)$  derivative approximations give  $O(h^3)$  bounds on the interpolation error, see Corollary 2.1. The use of  $O(h^3)$  derivative approximations for monotonic interpolation is discussed in detail in reference [2]. Unfortunately these approximations do not necessarily satisfy the convexity constraints and the existence of  $O(h^3)$  approximations which a priori satisfy such constraints is an open question. Finally it should be noted that the rational quadratic (3.10) can be used to construct a  $C^2$  rational spline which interpolates strictly monotonic data. This is discussed in detail in Delbourgo and Gregory [1] where it is shown that the spline produces  $O(h^3)$  derivative approximations.

**4. Numerical results.** We consider the application of the rational schemes to two sets of data. The first is the monotonic and convex set defined by  $f(x) = 1/x^2$  on  $[-2, -0.2]$ , with the interpolation points at  $x = -2, -1, -0.3$  and  $-0.2$ . This is the example used by McAllister et al. [6]. Since few data points are given, this is a fairly severe test of any scheme, particularly one where the derivatives are estimated from the data. Also, we cannot expect the rational interpolants to reproduce  $1/x^2$ , because of the nonlinear nature of the interpolation method.

Figure 1 shows that application of the rational cubic scheme of § 3.2 to the above data, where  $r_i$  is defined by (3.21). The graphs (i) and (ii) are respectively the interpolants with the arithmetic and geometric  $O(h^2)$  derivative approximations of § 3.4, and graph (iii) is that with the known exact derivatives. As expected from the theory, all graphs are convex but the graph with the arithmetic derivative approximations is not monotonic. It can be seen that the graph with the exact derivative settings gives the best result.

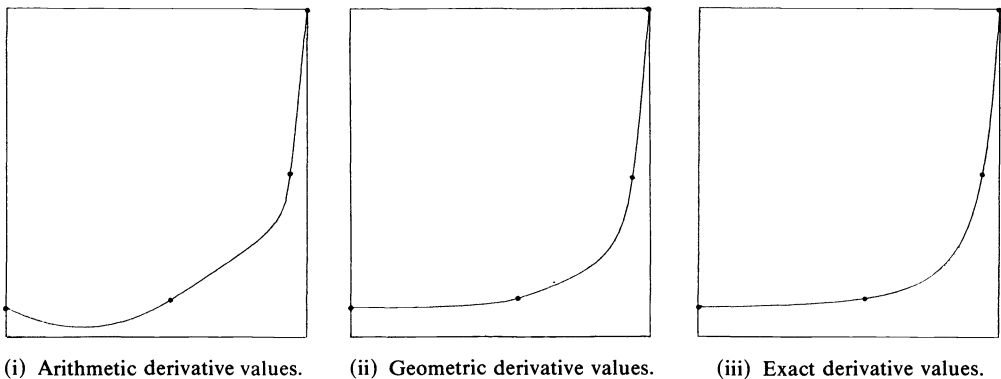


FIG. 1. Convex rational cubics for  $f(x) = 1/x^2$  on  $[-2, -0.2]$ .

Since the data is monotonic, the rational quadratic scheme of § 3.1 is also applicable. Figure 2 shows the application of this scheme with various choices of the derivative parameters. These are (i) the  $O(h^2)$  geometric approximations of § 3.4, (ii) the  $C^2$  spline approximations of Delbourgo and Gregory [1], and (iii) the known exact derivatives. All curves are monotonic but (i) exhibits an inflexion. Curves (ii) and (iii) give good results. It should be noted that exact end conditions have been used for the

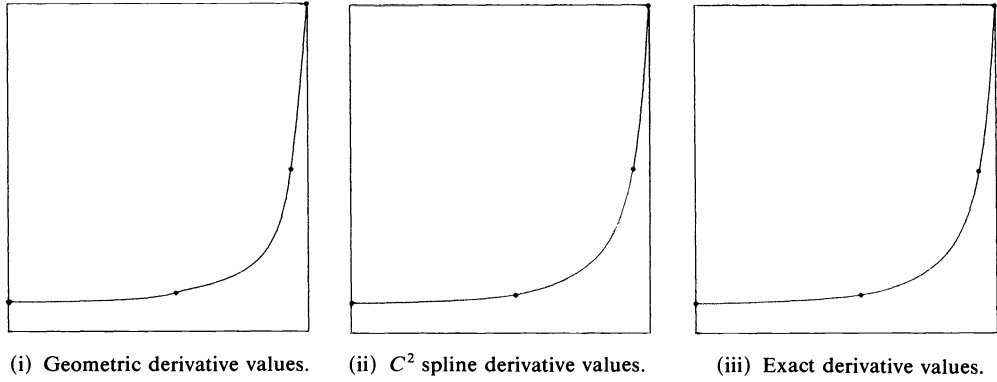


FIG. 2. *Monotonic rational quadratics for  $f(x) = 1/x^2$  on  $[-2, -0.2]$ .*

$C^2$  spline scheme. The alternative use of geometric approximations to the end derivatives gives comparable results for this set of data. The curves illustrate that although the monotonic rational quadratic schemes are not a priori convex, in practice they might be so. An a posteriori test for convexity is the necessary and sufficient condition (3.18).

Our second set of data consists of points uniformly spaced at  $15^\circ$  intervals over a half or quarter circle. The half circle of points is a convex but not monotonic set and the set of points on the quarter circle is convex and monotonic.

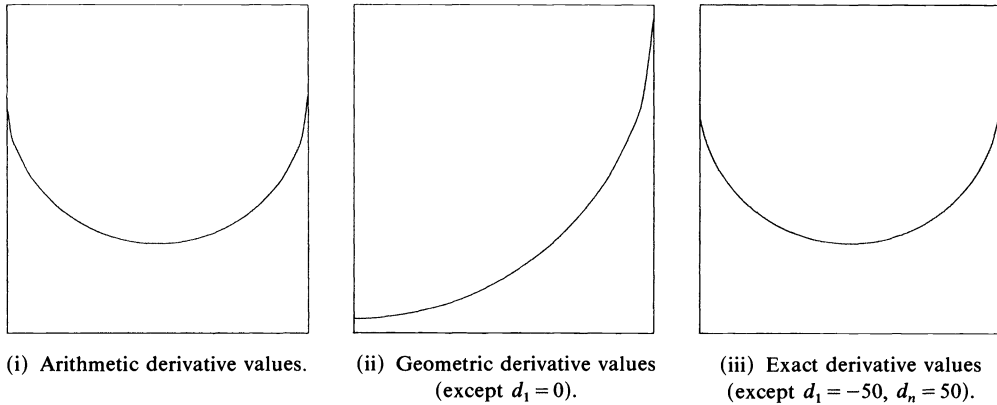
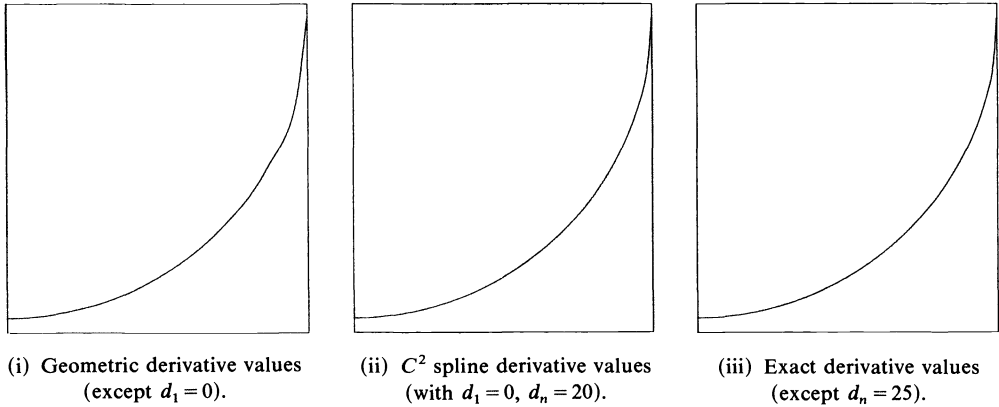


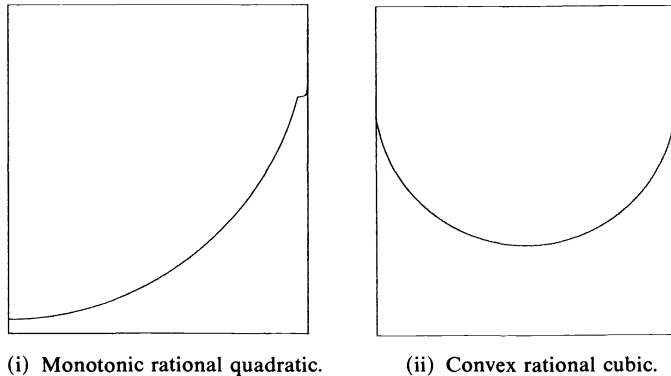
FIG. 3. *Convex rational cubics for circle data.*

The results of applying the convex rational cubic schemes and the monotonic rational quadratic schemes to the circle data are given in Figs. 3 and 4. Figure 3 shows that convexity is assured for all choices of the derivative parameters, the arithmetic settings being suitable for the convex half circle data and the geometric settings being appropriate for the convex and monotonic quarter circle data. The choice of exact derivatives has once more produced a good result, although here the end derivative values  $\pm 50$ , which replace the infinite gradients of the circle data, have been set by trial and error.

It can be seen from Fig. 4 that the monotonic rational quadratic scheme with geometric derivative approximations has a slight inflexion in the curve. The choice of exact derivatives or the  $C^2$  spline approximations have again produced good curves,

FIG. 4. *Monotonic rational quadratics for circle data.*

where at the end, where the quarter circle has infinite gradient, we have set the derivative  $d_n$  by trial and error. Too large a value of  $d_n$  creates an inflexion in the last interval and it is of interest to compare the behaviour of the rational quadratic and rational cubic schemes as the end condition  $d_n$  is made large. Figure 5 illustrates this for the case  $d_n = 1,000$  with exact derivative settings elsewhere. Since the rational quadratic has only to maintain monotonicity, the graph begins to behave in a step function manner. However, the additional convex constraint on the rational cubic eliminates this behaviour and instead produces a straight line almost vertical section at the end.

FIG. 5. *The effect of large end conditions for circle data.*

**5. Conclusion.** A shape preserving piecewise rational cubic scheme has been described which can be used to interpolate convex and/or monotonic data. The method seems to produce visually pleasing  $C^1$  curves and good error bounds can be expected, particularly when exact derivative information is given at the interpolation points.

## REFERENCES

- [1] R. DELBOURGO AND J. A. GREGORY,  $C^2$  rational quadratic spline interpolation to monotonic data, *IMA J. Numer. Anal.*, 3 (1983), pp. 141-152.
- [2] R. DELBOURGO AND J. A. GREGORY, *The determination of derivative parameters for a monotonic rational quadratic interpolant*, TR/07/84, Dept. Mathematics and Statistics, Brunel University, Uxbridge, England.

- [3] F. N. FRITSCH AND J. BUTLAND, *A method for constructing local monotone piecewise cubic interpolants*, SIAM J. Sci. Stat. Comput., 5 (1984), pp. 300-304.
- [4] F. N. FRITSCH AND R. E. CARLSON, *Monotone piecewise cubic interpolation*, SIAM J. Numer. Anal., 17 (1980), pp. 238-246.
- [5] J. A. GREGORY AND R. DELBOURGO, *Piecewise rational quadratic interpolation to monotonic data*, IMA J. Numer. Anal., 2 (1982), pp. 123-130.
- [6] D. F. MCALLISTER, E. PASSOW AND J. A. ROULIER, *Algorithms for computing shape preserving spline interpolations to data*, Math. Comp., 31 (1977), pp. 717-725.
- [7] D. F. MCALLISTER AND J. A. ROULIER, *An algorithm for computing a shape preserving osculatory quadratic spline*, ACM Trans. Math. Software, 7 (1981), pp. 331-347.
- [8] E. PASSOW, *Piecewise monotone spline interpolation*, J. Approx. Theory, 12 (1974), pp. 240-241.
- [9] E. PASSOW AND J. A. ROULIER, *Monotone and convex spline interpolation*, SIAM J. Numer. Anal., 14 (1977), pp. 904-909.

## ON OVERWHELMING NUMERICAL EVIDENCE IN THE SETTLING OF KINNEY'S WAITING-TIME CONJECTURE\*

EUGENE F. SCHUSTER†

**Abstract.** The computer is often used as an exploratory tool in problem solving. This exploratory analysis is very important in gaining insight into problems and theorems and often leads to new conjectures. Heuristic reasoning may then lead from the conjecture to a formal theorem whose validity leads itself to further numerical investigation. When these numerical investigations present overwhelming numerical evidence that the proposed theorem is true, the researcher sets upon a search for a mathematical proof. This note reports experience in this regard in the settling of Kinney's waiting time conjecture on the asymptotic logarithmic form of the expected value of the maximum of independent identically distributed geometrics. The lesson learned is that overwhelming numerical evidence is not enough.

**Key words.** asymptotic mean, maximum of geometrics, weighing numerical evidence, Kinney's conjecture, waiting time

**1. Introduction.** Kinney (1978) considered the problem of tossing  $n$  coins simultaneously. Those coins which come up tails are tossed again. This process is continued until each of the coins comes up heads. Kinney wondered how many (group) tosses would be required on the average. If we let  $Y_n = \max(X_1, \dots, X_n)$ , where  $X_1, \dots, X_n$  are independent identically distributed geometric random variables with probability density function (pdf)

$$p(x) = pq^{x-1}, \quad 0 < p < 1, \quad q = 1 - p, \quad x = 1, 2, 3, \dots,$$

then the solution of Kinney's problem is almost routine and is given by

$$(1.1) \quad E(Y_n) = \sum_{x=1}^n \binom{n}{x} (-1)^{x+1} (1 - q^x)^{-1},$$

where  $q$  is the probability of a tail on a single toss (see Schuster (1975), Kinney (1978)). Schuster (1975) studied  $E(Y_n)$  as the expected length of a ring tossing game and as the waiting time until failure of a certain active redundant system of  $n$  components connected in parallel.

Kinney noted that the formula (1.1) could be used to compute  $E(Y_n)$  for small values of  $n$ . However, for large values of  $n$ , the binomial coefficients  $\binom{n}{x}$  become very large, whereas the factors  $(-1)^{x+1}(1 - q^x)^{-1}$  alternate in sign while rapidly approaching a magnitude of 1. Thus roundoff error will quickly undermine the computing accuracy of (1.1).

For a large value of  $n$ , say  $n = 1,000$ , Kinney was forced to turn to simulation to estimate  $E(Y_n)$ . However, for  $n \leq 50$ , he used (1.1) to compute  $E(Y_n)$  to five decimal places. A graph of  $n$  versus  $E(Y_n)$  for this data led Kinney to conjecture that  $E(Y_n)$  is asymptotically a logarithmic function. In Fig. 1 we display Kinney's evidence for this conjecture for  $q = \frac{1}{2}$  in a Calcomp plot of select data pairs  $(n, E(Y_n))$  for  $n \leq 50$  together with the graph of the simple least squares logarithmic fit to this data given by Kinney as the estimator  $\hat{E}(Y_n) = 0.952575 \log_2 n + 1.58995$ . Notice that, even for small  $n$ , the data pairs  $(n, E(Y_n))$  located at the  $x$ 's fall remarkably close to the graph of the logarithmic curve.

\* Received by the editors September 12, 1983, and in revised form July 19, 1984.

† Department of Mathematical Sciences, University of Texas at El Paso, El Paso, Texas 79968.

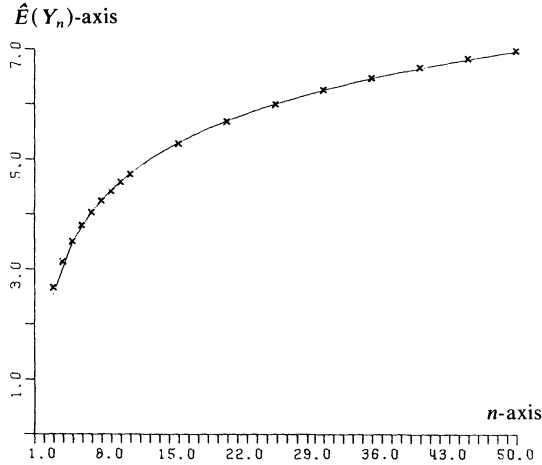


FIG. 1. The  $x$ 's show the proximity of select data pairs  $(n, E(Y_n))$  to the graph of the logarithmic curve  $\hat{E}(Y_n) = 0.952575 \log_2 n + 1.58995$ .

**2. Formalizing Kinney's conjecture.** Let us consider the following formal version of Kinney's conjecture:

$$\lim_{n \rightarrow \infty} \{E(Y_n) - a \log(n) - b\} = 0 \quad \text{for some } a = a(q), \quad b = b(q).$$

We first argue for appropriate choices for  $a$  and  $b$ . The distribution function (cdf) of  $Y_n$  is easily seen to be

$$F_n(x) = \begin{cases} (1 - q^k)^n & \text{for } k \leq x < k + 1, \quad k \in \{0, 1, 2, \dots\}, \\ 0 & \text{otherwise.} \end{cases}$$

Let  $Y_n^c$  be the continuous (exponential based) version of  $Y_n$  having cdf  $F_n^c(x) = (1 - q^x)^n$  for  $x \geq 0$  and 0 otherwise ( $c$  for continuous). Then one can easily establish the distributional relationship  $Y_n = [Y_n^c] + 1$ , where  $[x]$  is the greatest integer function. Now it is not difficult to see that

$$\begin{aligned} E(Y_n^c) &= \int_0^\infty x d(1 - q^x)^n = \int_0^\infty \{1 - (1 - q^x)^n\} dx \\ &= \sum_{x=1}^n \binom{n}{x} (-1)^{x+1} \frac{1}{x \log(1/q)} = \frac{S_n}{\log(1/q)}, \end{aligned}$$

with  $S_n = \sum_{x=1}^n (1/x)$ . Thus

$$\lim_{n \rightarrow \infty} \{E(Y_n^c) - a^c \log(n) - b^c\} = 0,$$

where  $a^c = 1/\log(1/q)$ ,  $b^c = \gamma/\log(1/q)$ , and  $\gamma = 0.577215 \dots$  is Euler's constant. Since  $[Y_n^c] \leq Y_n^c + 1$ , the following weak version of Kinney's conjecture is immediate:

LEMMA 1.  $\lim_{n \rightarrow \infty} \{E(Y_n)/(a^c \log(n) + b^c)\} = 1$ .

Note, however, that since Lemma 1 is true, we would have  $\lim_{n \rightarrow \infty} \{E(Y_n)/(a^c \log(n) + b)\} = 1$  for any choice of  $b$  and we could not immediately deduce the strong asymptotic order of  $E(Y_n)$  as conjectured. However, we can conclude that  $a = a^c = 1/\log(1/q)$ . Now to find  $b^c$ .

Since  $Y_n = [Y_n^c] + 1$  and  $Y_n^c \leq Y_n \leq Y_n^c + 1$ , a first naive guess might put  $E(Y_n)$  asymptotically at  $E(Y_n^c) + \frac{1}{2}$ . This guess is further reinforced by taking  $U$  to be a



uniform over  $(0, 1)$  independent of  $Y_n$  and then considering  $Z_n = [Y_n^c] + U$  as an approximation of  $Y_n^c$ . In this case the cdf of  $Z_n, F_{Z_n}$ , is a simple continuous piecewise linear approximation to  $F_n^c$  that equals  $F_n^c$  at the integers. On any interval  $[k, k + 1]$ , both distribution functions would flatten out entirely and the difference between  $F_{Z_n}$  and  $F_n^c$  could be made arbitrarily small as  $n$  tends to infinity. This approximation again leads to

$$\begin{aligned} E(Y_n) &= E[Y_n^c] + 1 = E([Y_n^c] + U) + 1 - E(U) \\ &= E([Y_n^c] + U) + \frac{1}{2} \cong E(Y_n^c) + \frac{1}{2} = S_n \log \frac{1}{q} + \frac{1}{2}. \end{aligned}$$

Thus our version of Kinney's conjecture becomes

$$(2.1) \quad \lim_{n \rightarrow \infty} \{E(Y_n) - a \log(n) - b\} = 0,$$

$$\text{with } a = a^c = \frac{1}{\log(1/q)} \text{ and } b = b^c + \frac{1}{2} = \frac{\gamma}{\log(1/q)} + \frac{1}{2}.$$

Numerical evidence for this conjecture can be obtained by *abandoning the finite summation formula (1.1) in favor of the infinite series (2.2)* derived from the expansion for the mean of nonnegative integer-valued discrete random variables in terms of the tail probabilities  $P(Y_n > x)$ :

$$(2.2) \quad E(Y_n) = \sum_{x=0}^{\infty} P(Y_n > x) = \sum_{x=0}^{\infty} \{1 - (1 - q^x)^n\}.$$

Alternately, one can obtain this formula directly by substituting the geometric series expansion for  $1/(1 - q^x)$  in (1.1) and then interchanging the order of summation. Now  $1 - (1 - q^x)^n = q^x \{1 + (1 - q^x) + \dots + (1 - q^x)^{n-1}\} \leq nq^x$ . Thus if we approximate  $E(Y_n)$  by the  $k$ th partial sum of the series in (2.2), our approximation error is less than  $nq^{k+1}/(1 - q)$ . Note that *the infinite series in (2.2) is better than the finite sum in (1.1)* in the sense that it allows one to numerically solve Kinney's problem of computing  $E(Y_n)$ . In Table 1 we present the numerical evidence for the conjecture (2.1) by comparing the 50th partial sum of the series in (2.2) for  $q = \frac{1}{2}$  with the approximation of  $E(Y_n)$  given by  $\hat{E}(Y_n) = a \log(n) + b + a/2n$ , where  $a$  and  $b$  are given in (2.1). This particular form of the estimate is used since the asymptotic result in (2.1) was obtained by replacing  $S_n = \sum_{x=1}^n (1/x)$  by  $\log(n) + \gamma + 1/2n$  (the error in this approximation is then  $O(1/n^2)$ ).

TABLE 1

$n$	Computed $E(Y_n)^*$	Approximate $E(Y_n)$	Error
10	4.72555932363	4.72680894779	-0.00124962416
20	5.69043836083	5.69074157177	-0.00030321094
30	6.26355131465	6.26368161382	-0.00013029917
40	6.67263307715	6.67270788376	-0.00007480661
50	6.99097790342	6.99102924105	-0.00005133763
100	7.98380153516	7.98381576584	-0.00001423068
200	8.98020421978	8.98020902824	-0.00000480846
400	9.97840328257	9.97840565944	-0.00000237687
800	10.97750224455	10.97750397504	-0.00000173049

\* Truncation error  $\leq 10^{-12}$ .

After reviewing Table 1 we can set about finding a mathematical proof of the conjecture (2.1) with overwhelming numerical evidence that the conjecture is true.

**3. Settling Kinney's conjecture.** We first note that for each positive integer  $n$  we can write  $-\log_q(n) = r_n + d_n$ , where  $r_n$  is a nonnegative integer and  $0 \leq d_n < 1$ . Our first lemma indicates that the  $d_n$ 's are dense in  $[0, 1]$ .

LEMMA 2. *For each  $d$  in  $[0, 1]$ , there exists a subsequence  $\{d_{n_i}\}$  of  $\{d_n\}$  with  $\lim_{i \rightarrow \infty} d_{n_i} = d$ .*

*Proof.* Suppose not. Then there exists some  $d$  in  $[0, 1]$  and an  $\varepsilon > 0$  such that  $[d - \varepsilon, d + \varepsilon]$  does not contain any member of the sequence  $\{d_n\}$ . Let  $[\alpha, \beta] = [d - \varepsilon, d + \varepsilon] \cap [0, 1]$ . Then it follows that the interval  $[\alpha^*, \beta^*] = [(1/q)^{r+\alpha}, (1/q)^{r+\beta}]$  does not contain an integer for any integer  $r > 0$ . But this is a contradiction since the length of  $[\alpha^*, \beta^*]$  tends to infinity as  $r$  tends to infinity.  $\square$

Our biggest stumbling block in settling Kinney's conjecture was in proving the following Lemma 3. We were unable to find a proof of this lemma until we put on our "statistician's hat" and considered the  $d$  of the lemma to be a parameter of some family of distributions. A mathematical proof is outlined in § 4.

LEMMA 3. *The function  $H(d) = \int_{-\infty}^{\infty} (y + d - [y + d]) d \exp(-q^y)$  is not constant on  $[0, 1]$ .*

*Proof.* Take  $h(d) = H(d) - H(0)$ . Since  $G(y) = \exp(-q^y)$  is a distribution function, we see that

$$\begin{aligned} h(d) &= \int_{-\infty}^{\infty} \{y + d - [y + d] - H(0)\} d \exp(-q^y) \\ &= \int_{-\infty}^{\infty} \{y - [y] - H(0)\} dG(y; d), \end{aligned}$$

where  $G(y; d) = \exp(-q^{y-d})$ . Let  $G'(y; d) = g(y; d)$ . Then the family of probability density functions  $\{g(y; d), d \in [0, 1]\}$  is an exponential family and this exponential family is complete (see Lehman (1959)). But this says that  $H(d)$  is not identically zero on  $[0, 1]$  since, by the completeness property, the only function  $g$  with  $\int_{-\infty}^{\infty} g(y) dG(y; d) = 0$  for all  $d \in [0, 1]$  must be the zero function, except possibly on a null set  $N$  with  $G(y; d)$  probability zero for every  $d$  in  $[0, 1]$ . Clearly  $g(y) = y - [y] - H(0)$  is not equivalent to the zero function, and so  $h(d) = H(d) - H(0)$  is not identically zero on  $[0, 1]$ , i.e.,  $H(d)$  is not constant on  $[0, 1]$ .  $\square$

Using Lemmas 2 and 3 we can prove:

LEMMA 4.  *$\lim_{n \rightarrow \infty} E(Y_n^c - [Y_n^c])$  does not exist.*

*Proof.* Define  $G_n(y) = (1 - q^y/n)^n$  if  $y \geq \log_q n$  and equals 0 otherwise and  $G(y) = \exp(-q^y)$ ,  $-\infty < y < \infty$ . Choose  $d$  in  $[0, 1]$ . Using the notation of Lemma 2, we can find a subsequence of the positive integers, say  $\{n_i\}$ , with  $\{d_{n_i}\}$  converging to  $d$ . Noting that  $x + r - [x + r] = x - [x]$  for any integer  $r$ , one can use Chung (1974, exercise 10, p. 100) to see that

$$\begin{aligned} \lim_{i \rightarrow \infty} E(Y_{n_i}^c - [Y_{n_i}^c]) &= \lim_{i \rightarrow \infty} \int_0^{\infty} (y - [y]) d(1 - q^y)^{n_i} \\ &= \lim_{i \rightarrow \infty} \int_{\log_q n_i}^{\infty} (y - \log_q n_i - [y - \log_q n_i]) d \left(1 - \frac{q^y}{n_i}\right)^{n_i} \\ &= \lim_{i \rightarrow \infty} \int_{-\infty}^{\infty} (y + d_{n_i} - [y + d_{n_i}]) dG_{n_i}(y) \end{aligned}$$

$$\begin{aligned}
 &= \lim_{i \rightarrow \infty} \int_{-\infty}^{\infty} (y - [y]) dG_{n_i}(y - d_{n_i}) \\
 &= \int_{-\infty}^{\infty} (y - [y]) dG(y - d) = \int_{-\infty}^{\infty} (y + d - [y + d]) dG(y) = H(d).
 \end{aligned}$$

The desired conclusion easily follows from Lemma 3 since we can find subsequences which converge to different limits.  $\square$

Suppose the strong form of Kinney's conjecture is true and that  $\lim_{n \rightarrow \infty} \{E(Y_n) - a \log n - b\} = 0$  for some  $a = a(q)$ ,  $b = b(q)$ . Then since  $Y_n = [Y_n^c] + 1$  and  $E(Y_n^c) = S_n / \log(1/q)$ , we can conclude that

$$\lim_{n \rightarrow \infty} \left\{ E(Y_n) - a \log n - b - E(Y_n^c) + \frac{S_n}{\log(q)} \right\} = \lim_{n \rightarrow \infty} \{E(Y_n - Y_n^c) - a^* \log n - b^*\} = 0$$

for some  $a^*$  and  $b^*$  depending on  $q$ . Since  $E|Y_n - Y_n^c| \leq 1$ ,  $a^*$  must be zero. This means that  $\lim_{n \rightarrow \infty} E([Y_n^c] + 1 - Y_n^c) = \lim_{n \rightarrow \infty} E(Y_n - Y_n^c)$  must exist, in contradiction to Lemma 4. Consequently,

*The strong form of Kinney's conjecture is false.*

Although Kinney's conjecture is false, the following theorem indicates that both the  $\liminf$  and  $\limsup$  of  $E(Y_n)$  are asymptotically of logarithmic form:

**THEOREM**

$$\lim_{n \rightarrow \infty} \{E(Y_n) - a \log(n) - b + H(a \log(n))\} = 0,$$

where  $a = -1/\log(q)$ ,  $b = a \cdot \gamma - 1$ ,  $\gamma$  is Euler's constant, and  $H(d) = \int_{-\infty}^{\infty} (y + d - [y + d]) d \exp(-q^y)$ .

*Proof.* Using the representation  $Y_n = [Y_n^c] + 1$  of § 2 and the notation of Lemma 4, we first define  $H_n(d) = \int_{-\infty}^{\infty} (y + d - [y + d]) dG_n(y)$  and  $c_n = a \log(n)$ . Proceeding as in the proof of Lemma 4, we see that

$$\begin{aligned}
 \lim_{n \rightarrow \infty} \{E(Y_n - E(Y_n^c) + H_n(c_n))\} &= \lim_{n \rightarrow \infty} \{E([Y_n^c] + 1 - Y_n^c + H_n(c_n))\} \\
 &= 1 + \lim_{n \rightarrow \infty} \{H_n(c_n) - H_n(c_n)\} = 1.
 \end{aligned}$$

In § 2 we noted that  $E(Y_n^c) = -S_n / \log(q)$ , where  $S_n = \sum_{x=1}^n (1/x)$ . Hence  $\lim_{n \rightarrow \infty} \{E(Y_n^c) - a \log(n) - a\gamma\} = 0$ . Thus it suffices to show that  $\lim_{n \rightarrow \infty} h_n = 0$ , where  $h_n = |H_n(c_n) - H(c_n)|$ . Since  $|y - [y]| \leq 1$ , it follows that  $\lim_{n \rightarrow \infty} \sup h_n = \alpha$  exists and is finite. Let  $\{n_i\}$  be a subsequence of integers with  $\lim_{i \rightarrow \infty} h_{n_i} = \alpha$ . As in § 3, we write  $c_{n_i} = r_{n_i} + d_{n_i}$ , where  $r_{n_i}$  is an integer and  $0 \leq d_{n_i} < 1$ . Let  $\{d_{n_{i_k}}\}$  be a subsequence of  $\{d_{n_i}\}$  converging to a point  $d$  in  $[0, 1]$ . Using Chung (1974, exercise 10, p. 100), we can proceed as in the proof of Lemma 4 to see that  $\alpha = 0$ , and the proof is easily completed.

*Remark 1.* For the original coin tossing game of Kinney,  $q = \frac{1}{2}$ . In this case,

$$\lim_{i \rightarrow \infty} \{E(Y_{2^i}) - a \log(2^i) - b\} = 1 - H(0),$$

where  $a = -1/\log(q)$  and  $b = a\gamma$ . In fact, for any positive integer  $n$ , the subsequence  $\{E(Y_{n2^i})\}$  is asymptotically of logarithmic form with

$$\lim_{i \rightarrow \infty} \{E(Y_{n2^i}) - a \log(n2^i) - b\} = 1 - H(\log_2 n).$$

*Remark 2.* According to extreme value theory, one cannot find sequences  $\{a_n\}$ ,  $\{b_n\}$  so that  $(Y_n - a_n)/b_n$  has a nondegenerate limiting distribution. However, it is not

difficult to see that if  $\{Y_{n_i}\}$  is any subsequence with  $a_{n_i} = -\log_q(n_i)$  a positive integer for every  $i$ , then  $Y_{n_i} + \log_q(n_i)$  will have a limiting distribution with limiting cdf  $F(x) = \exp\{-q^{\lceil x \rceil}\}$ . In Kinney's coin tossing game,  $Y_{2^i} - i$  would have limiting distribution  $F(x) = \exp\{-\frac{1}{2}^{\lceil x \rceil}\}$ .

**4. Why does the conjecture come so "close?"** In Remark 1, we see that many subsequences of  $E(Y_n)$  are indeed asymptotically of logarithmic form. Moreover, we have shown that the general asymptotic form of  $E(Y_n)$  differs from a logarithmic function  $a \log(n) + b$  only by  $H(-\log_q(n))$ . In the following we see that the function  $H$  is remarkably close to the constant  $\frac{1}{2}$ , at least when  $q = \frac{1}{2}$ .

Let  $G$  be a distribution function with characteristic function  $\Phi$ , and let  $h$  be a bounded measurable 1-periodic function with (complex) Fourier coefficients  $\{h_n\}$ . Then  $H(x) = \int_{-\infty}^{\infty} h(x+y) dG(y)$  is also 1-periodic with Fourier coefficients  $\{H_n = h_n \Phi(2\pi n)\}$ . As a corollary, if  $\Phi(2\pi n)$  never vanishes, then  $H$  can be constant if and only if  $h_n = 0$  for  $n \neq 0$ , i.e.  $h$  is constant. Since the fractional function  $x - [x]$  is nonconstant 1-periodic, this leads to a nice mathematical proof of Lemma 3. Now, for the 1-periodic function  $H$  of the theorem in § 3, one can show that  $H_0 = \frac{1}{2}$ , and for  $n \neq 0$ ,  $H_n = -\Gamma(2\pi i n / \log q) / \log q$ , which goes to zero extremely rapidly. Using the well-known identity  $|\Gamma(ix)| = \sqrt{\pi} / (x \sinh(\pi x))$  for  $x$  real, we see that even for  $|n| = 1$  the modulus of  $H_n$  is about  $7.9 \times 10^{-7}$  when  $q = \frac{1}{2}$ . For  $|n| = 2, 3$  the moduli are about  $3.6 \times 10^{-13}$  and  $1.9 \times 10^{-19}$ , respectively.

*Remark 3.* Although we will continue to use the computer as an exploratory tool in problem solving, we have learned the hard way that *overwhelming numerical evidence* is not enough. The evidence in the present case did play an important but perhaps a usual role. It fueled our stubborn persistence until we discovered the "truth."

#### REFERENCES

- [1] KAI LAI CHUNG (1974), *A Course in Probability Theory*, 2nd. ed. Academic Press, New York and London.
- [2] JOHN KINNEY (1978), *Tossing coins until all are heads*, Math. Magazine, 51, 3, pp. 184-186.
- [3] E. L. LEHMANN (1959), *Testing Statistical Hypotheses*, John Wiley, New York.
- [4] EUGENE F. SCHUSTER (1975), *An integer programming handicap system in a "Write Ring Tossing Game,"* Math. Magazine, 48, 3, pp. 134-142.

## ON MINIMIZING A SET OF TESTS\*

BERNARD M. E. MORET<sup>†‡</sup> AND HENRY D. SHAPIRO<sup>†</sup>

**Abstract.** Minimizing the size or cost of a set of tests without losing any discrimination power is a common problem in fault testing and diagnosis, pattern recognition, and biological identification. This problem, referred to as the minimum test set problem, is known to be NP-hard, so that determining an optimal solution is not always computationally feasible. Accordingly, researchers have proposed a number of heuristics for building approximate solutions, without, however, providing an analysis of their performance. In this paper, we take an in-depth look at the main heuristics and at the optimal solution methods, both from a theoretical and an experimental standpoint. We characterize the worst-case behavior of the heuristics and discuss their use in bounding. We then present the results of extensive experimentation with randomly generated problems. While the exponential explosion suggested by the problem's NP-hardness is apparent, our results suggest that real world testing problems of large sizes can be solved quickly at the expense of large storage requirements.

**Key words.** experimental results, greedy heuristics, minimum test set, NP-hard, single branch enumeration, worst-case behavior

**1. Introduction.** Identification problems arise in almost all fields of scientific research. We are concerned here with a special type of deterministic identification, where an unknown (system state, animal species, location of a fault) must be classified in one of a given set of categories, based on the outcome of a set of tests. Each category is characterized by a vector of test outcomes, and an unknown object is classified in that category if its vector of test outcomes matches the category's characteristic vector. The collection of all categories together with their characteristic vectors is known as a *diagnostic table*. A diagnostic table with  $m$  categories and  $n$  tests can be represented as an  $m \times n$  matrix, where the  $(i, j)$  entry is the result of test  $T_j$  applied to an object in category  $C_i$ . Such formulation is common in testing and fault analysis [2], [4], [11], biology [15], [22], [23] and pattern recognition [5], [13].

Given a diagnostic table, it is often the case that some tests are redundant. In such a case, it is of interest to find the smallest suitable subset in order to minimize the cost of identification. The *minimum test set* (also known as the test of minimum length) is the smallest subset of tests which discriminates between all categories distinguished by the full set of tests. Knowledge of the minimal test set can reduce costs in applications where a rapid identification is needed, that is, in situations where all the tests will be applied in parallel. Cost reduction will also occur in applications where the capital costs (procurement of the test equipment) far exceed the running costs, regardless of whether the actual testing is done in a parallel or sequential manner. (This is the minimization of the *acquisition cost* in decision trees [12].) Applications of the second type are to be found in most fields of human endeavor, including some that do not explicitly include testing: servicing equipment under poor access conditions (military equipment in the field, oil rigs at sea), where the cost of delivering service personnel and apparatus must be minimized; remote sensing missions, where the cost of the apparatus must be minimized subject to performing all the required tasks; and fault diagnosis and design for testability, where, for instance, the number of checkpoints added to a circuit must be minimized subject to retaining a prescribed level of testability.

Unfortunately, the minimization problem is known to be NP-hard [6]. Accordingly, researchers have developed a number of heuristics for building suboptimal test collec-

\* Received by the editors August 5, 1983, and in final form June 11, 1984.

<sup>†</sup> Department of Computer Science, University of New Mexico, Albuquerque, New Mexico 87131.

<sup>‡</sup> The work of this author was supported by the Office of Naval Research, under grant 0014-78-C-0311.

tions by using variants of a greedy algorithm where, at each step, the locally optimal test is added to the partial solution. However, no analysis of those methods is offered in the literature.

In this paper, we take an in-depth look at existing heuristics and how they can be applied to develop optimal solutions. We show that existing selection heuristics can exceed the optimal by a factor of at most  $\log n$  and provide a generic example where this factor is asymptotically reached. We then present and discuss the results of extensive experiments with both artificial (randomly generated) and real-world problems. While the exponential explosion suggested by the problem's NP-hardness is quite apparent in the artificial examples, our results suggest that real-world problems of large sizes can be solved in reasonable time.

**2. An analysis of proposed heuristics.** Almost all proposed heuristics belong to the class of *greedy* algorithms, in that they perform local, step-by-step optimization, using a suitable selection criterion. Very few analytical results are available about the minimum test set problem in general and the behavior of the proposed heuristics in particular. A number of Russian researchers [8], [9], [21] have studied the expected size of the minimum test set for randomly constructed tables; the analyses of the main two heuristics discussed in [12] in the context of identification trees do not extend to the minimum test set problem. In the following, we briefly define the four main heuristics proposed in the literature and offer a partial analysis of their worst-case behavior.

**2.1. Definitions.** When a pair of categories is distinguished by only one test (that is, the categories' characteristic vectors differ in exactly one component), that test is called *essential* and must be included in any complete set of tests. Thus, in a step-by-step method, preincluding all essential tests is an optimal policy; all proposed methods [2], [3], [18], [20] make use of this policy.

When all essential tests have been included, one can either attempt to extend the notion of essentiality or resort to a measure of a test's local optimality. The first approach has been used by researchers in microbiology [16], [18], [20]: since a test is essential when it is the only test to separate a pair of categories, a test is "nearly essential" if it is one of only two (or a few) tests to separate a pair of categories. This extension restricts the choice of the next test to one of those that separate that pair of categories which is separated by the least number of tests—the least-separated pair. An algorithm using this criterion will thus focus on category pairs; ties between tests and/or between equally poorly separated pairs are broken by the use of a "second-level" heuristic—one of the measures of local optimality described below. We shall call this the *least-separated pair* criterion.

The second approach attempts to measure how well a new test will complement those already chosen; in such an approach, all as yet unclassified tests are considered for inclusion. An obvious choice is to count how many as yet unclassified pairs the new test will distinguish and choose a test which maximizes this count: we shall call this the *separation* criterion. This heuristic has been extensively used in fault analysis [3], [4] and microbiology [16], [20]. The contribution of a test can also be measured in terms of entropy (or, equivalently, of information), in which case the initial state—a single homogeneous group—corresponds to an entropy of 0 and the final state— $m$  distinct groups of one category each—to an entropy of  $\log_2 m$ ; it can also be measured in terms of permutations, where the initial state corresponds to a value of 1—for there is only one way to assign a label to the single initial set—and the final state corresponds to a value of  $m!$ , the number of ways in which  $m$  distinct items can be labelled. The information theory approach is found early in the literature and used extensively for both the minimum test set and the related minimum identification tree problems [4],

[12], [16]. Formally, the entropy of collection  $C$  of  $k$  clusters, of sizes  $s_1, \dots, s_k$ , comprising  $m$  elements in all, is defined as

$$H(C) = \log_2 m - \frac{1}{m} \sum_{i=1}^k s_i \cdot \log_2 s_i.$$

Applying a test to a collection of clusters yields a new collection, with larger (or equal) entropy; the difference is the amount of information contributed by that test. The test which brings about the largest increase will be selected. We shall call this approach the *information* criterion. The combinatorial approach, described in [17] and used in [14], considers how many possible distinct partitions of the size used could exist; the logarithm of this quantity is used as a measure, called *repartment* [17]. However, the repartment of a partition of  $m$  objects is equal to  $m$  times the entropy of the partition (within an additive factor of  $\log_2 m$ ), so that the two approaches are essentially equivalent.

Thus, we have four main heuristics: least-separated pairs with separation used to choose among the candidate tests (at the “second level”); least-separated pairs with information used at the second level; separation alone; and information alone. The first two restrict the choice of tests before applying a local measure of optimality, while the last two apply such a measure to all remaining tests.

**2.2. Analysis.** Examples are easily constructed that show that no heuristic is uniformly better than the other three. The four heuristics are rather similar: the effect of restricting the choice to those tests separating the least-separated pair does affect the order in which the tests are selected—which is of no consequence with regard to the final subset selected; it also affects the composition of the final subset, since a different order of selection may modify the local measures of optimality of the remaining tests. Typically, we found that these indirect effects are minor. Moreover, the measures of local optimality are all convex functions, the minima and maxima of which all occur at the same points.

It is easily shown that, in the worst-case, no heuristics will yield a solution with a cost that is at most a constant away from the optimal. Indeed, staying within a *fixed constant* around the optimal is itself an NP-hard problem (our proof uses a technique that has become standard in the field: see [6, pp. 138–139]). We show that this problem is NP-hard by showing that, if a heuristic existed that produced a solution that had at most  $k$  more tests than the optimal, then we could use it to construct the optimal solution (and thus solve a NP-hard problem). The idea is to scale our problem up, solve it with the heuristic, and then scale it down, so that the error margin will shrink to zero by truncation. Specifically, given a problem, we “multiply” it by some factor  $\alpha$ , by making  $\alpha$  copies of each object (regard each copy as a coordinate in a  $\alpha$ -tuple) and thus  $\alpha$  copies of each test (one copy for each coordinate); since the various coordinates may not be separable with the existing tests, we also add  $\log \alpha$  well-splitting tests for that purpose. Notice that the optimal solution for this problem has at most  $\alpha$  times plus  $\log \alpha$  the number of tests of the optimal solution for the original problem. The heuristic solution will fall within  $k$  of the optimal for the larger problem; now pick as solution for the original problem the smallest of the test sets obtained by retaining from the heuristic solution only those tests that apply to the same coordinate (and discarding any of the  $\log \alpha$  tests, which give a uniform result for any one coordinate). That set has no more than  $1/\alpha$  the number of tests of the heuristic solution; thus it has at most  $(\log \alpha + k)/\alpha$  more tests than the optimal solution; but all quantities must be integer, so that for an appropriate choice of  $\alpha$ , independent of the problem, we have in fact obtained the optimal solution.

Furthermore, another standard technique can be used to show that staying within an *arbitrarily small ratio* of the optimal is also NP-hard (i.e., no fully polynomial time approximation scheme [6] exists for the problem): it is an immediate consequence of the corollary on page 141 of [6] and of the fact that our problem is *strongly* NP-hard. Thus the best possible algorithm is one that produces solutions, the size of which stays within a *fixed* ratio of the optimal (of course, we have no guarantee that such a heuristic exists).

None of the four heuristics discussed above comes close to this bound; in fact, all four can produce solutions, the size of which exceeds the optimal by a ratio of  $\log n$ , for  $n$  objects. This ratio is also pessimal: it describes the worst-case behavior of the heuristics. The proof that  $\log n$  is the worst-case ratio and an example that shows that such a ratio is attainable, can both be derived by considering the closely related set covering problem [1], [6], [10]. A set covering problem is given by a family of sets, the goal being to find the smallest number of sets in that family that cover the family, i.e., such that their union is equal to the union of all sets in the family. We shall use  $s$  to denote the number of sets in the family and  $e$  to denote the total number of elements in the union. A minimum test set problem with  $n$  tests and  $m$  objects can be considered as a set covering problem with  $e = m \cdot (m - 1)/2$  elements (all distinct pairs of objects) and a family of  $s = n$  sets (each set is composed of all pairs distinguished by the corresponding test). Distinguishing all objects is then equivalent to covering all pairs. Conversely, given a set covering problem with  $e$  elements and  $s$  sets, we can construct a corresponding minimum test set problem with  $m = 2e$  objects (two objects for each element; one could think of a male and a female representative) and  $n = s + \lceil \log e \rceil$  tests. The  $\lceil \log e \rceil$  tests are even splitting tests that distinguish each male-female pair from all other pairs, but do not make any distinction between the male and the female representatives of a set element; they are all essential (since no other tests will differentiate the males of the pairs). The  $s$  tests correspond to the  $s$  sets: they return true only for objects that are female representatives of elements of the corresponding set; selecting some of these tests to complement the  $\lceil \log e \rceil$  tests is then equivalent to picking a set cover. Note that each transformation creates a bigger problem; in particular, the two transformations are not inverses of each other.

Johnson [7] has shown that the selection of covering sets by the greedy heuristic (pick that set which covers the largest number of not-yet-covered elements) must produce solutions that do not exceed the optimal by more than a ratio of  $\log e$ . Since the greedy heuristic based on separation picks that test which separates the largest number of not-yet-separated objects, we immediately get the desired upper bound on the worst-case behavior of the separation heuristic: a ratio of  $\log(m \cdot (m - 1)/2)$  or approximately  $2 \log m$ . Moreover, Johnson also provided a set covering problem where the  $\log e$  factor is reached. The example has  $3 \cdot 2^k$  elements and  $k + 4$  sets. The  $k + 4$  sets are in two groups: 3 disjoint sets of  $2^k$  elements each, which form the optimal cover; and  $k + 1$  disjoint sets of sizes  $3 \cdot 2^{k-1}, 3 \cdot 2^{k-2}, \dots, 3 \cdot 2, 3$ , and  $3$ , which will be picked by the greedy heuristic. The situation is illustrated below.

TABLE 1  
 $3 \cdot 2^k$  elements in all.

				...			1 set of $2^k$
				...			1 set of $2^k$
				...			1 set of $2^k$
3	3	6	12	...	$3 \cdot 2^{k-2}$	$3 \cdot 2^{k-1}$	

$k + 1$  sets of respective size



The ratio of the greedy solution to the optimal solution is  $(k+1)/3$ , or about  $\frac{1}{3} \log e$ . While a direct translation (by the construction described above) of this example yields a problem where our four heuristics behave well (staying within a ratio of 2), the problem can be “multiplied” in a fashion similar to that used above for our NP-hardness reduction so as to produce a worst-case example with a logarithmic ratio. Specifically, we duplicate it  $k^2$  times, then use the direct translation, giving rise to a collection of  $m = k^2 \cdot 3 \cdot 2^{k+1}$  objects. Thus the optimal cover has  $3 \cdot k^2$  sets, giving rise to a minimum test set of  $3 \cdot k^2 + \lceil \log(m/2) \rceil$  tests, while the greedy heuristic derives a cover of  $k^2 \cdot (k+1)$  sets, corresponding to a test set of  $k^2 \cdot (k+1) + \lceil \log(m/2) \rceil$  tests. Hence for large  $k$ , the ratio becomes  $k/3$ , which is approximately  $\frac{1}{3} \log m$ . (Duplication by a factor of  $k$  would be sufficient and lead to more rapid convergence, but the asymptotic ratio would only be  $\frac{1}{4} \log m$ .)

The example and the proof can be modified to account for the other three heuristics as well as for similar heuristics with finite look-ahead. Thus all heuristics based on local measures of a test’s power of discrimination have a poor worst-case behavior: they can yield solutions that are arbitrarily far from optimal. However, the worst-case construction is rather elaborate and produces large ratios only for very large problems. (Our worst-case construction has a logarithmic asymptotic ratio with a constant factor of  $\frac{1}{3}$ ; other constructions may converge to a ratio with a larger constant, since our proof merely shows that the largest constant factor is 2. Also, other constructions may converge faster than ours. Nevertheless, observe that, in our construction, we need over 10,000 objects in order to produce a ratio larger than 2; with  $k = 10$  and 614,000 objects, the ratio is still only 3.5.) In consequence, we would expect the heuristics to behave rather well on problems of practical size (less than a thousand objects).

**3. Bounding methods.** Exhaustive search algorithms that find the optimal solution—such as branch-and-bound or depth-first search—require both upper and lower bounds on the size of the optimal solution in order to run efficiently. The bounds are used in pruning, i.e., in eliminating fruitless directions of search (pruning occurs whenever the local lower bound reaches or exceeds the global upper bound); they can also be used in guiding the selection of that portion of the search space to explore next. A global upper bound is trivially provided by the size of the best solution found so far; from our previous discussion, we know that this bound can be arbitrarily loose, but would expect it to be fairly tight in most cases.

Any measure of a test’s discriminating power that gauges distances on the way from the initial to the final partition can be used to derive lower bounds. At any step, we compute the distance from the partition determined by our partial solution to the final partition. We also determine the local contribution of each available test—i.e., those not yet chosen nor eliminated. We then sort the available tests’ contributions in decreasing order and, assuming no interaction between tests, find by repeated summing how many tests are needed to complete the partial solution. Since the contribution of a test does not increase as the partial solution is expanded, this gives us a safe lower bound.

The separation-derived function gives us a lower bound on the number of additional tests needed to distinguish the remaining unseparated pairs, assuming that no two tests separate the same pair. The information-derived function finds a lower bound on the number of additional tests needed to increase the entropy to the final partition’s value, assuming no cross-information<sup>1</sup> between tests. We note that, whenever there is

<sup>1</sup> There is cross-information between two tests if the amount of information which they contribute together is less than the sum of the amounts which they contribute individually.

cross-information between two tests, there will be some pairs that they both distinguish, but that the converse is false. For instance, if  $m$  is a power of 2 and  $\log_2 m$  of the tests perfectly complement each other, then those tests have no cross-information, yet any two of them distinguish  $m^2/8$  common pairs. Thus we expect the information bound to be better than the separation bound.

A simple example will illustrate our point. Consider a problem with  $m$  (an even number) categories, where all tests effect an even split,  $m/2$  vs.  $m/2$ . The tightest possible lower bound on the size of the optimal solution is  $\lceil \log_2 m \rceil$ , an achievable size. Now, each test separates  $(m/2) \cdot (m/2) = m^2/4$  pairs and brings an increase in entropy of 1 bit, so that the separation-derived bound is

$$\left\lceil \frac{m \cdot (m-1)/2}{m^2/4} \right\rceil = \left\lceil \frac{2 \cdot (m-1)}{m} \right\rceil = 2,$$

while the information-derived bound is

$$\left\lceil \frac{\log_2 m}{1} \right\rceil = \lceil \log_2 m \rceil.$$

The information-derived bound is as tight as possible; the separation-derived bound is off by an unbounded factor.

In fact, the information-derived bound can also be arbitrarily smaller than the size of the optimal solution, although the factor cannot grow as large as for the separation-derived bound. Clearly, the worst possible behavior for the information-derived bound is to indicate the need for a logarithmic number of tests (close to the theoretical minimum) while the problem in fact requires a linear number of tests (close to the theoretical maximum). Such a behavior can be observed in a square diagnostic table where the 1 entries are disposed so as to make the table into a lower triangular matrix: the tests in the middle columns are well-splitting but closely correlated, thereby misleading both bounding methods. Thus, for  $m$  object categories, the information-derived bound can be off by a factor of  $O(m/\log_2 m)$ , while the separation-derived bound can be off by a factor of  $O(m)$ .

However, information-derived bounds are probably much preferable to separation-derived ones, since, the better the tests are, the poorer the separation-derived lower bounds become. In fact, since well-splitting tests are far more likely than others in randomly chosen test collections, the example discussed above is essentially the average case for random problems. Thus the separation-derived bounds will be practically useless in problems that have a large number of tests relative to their number of categories (a "wide" diagnostic table). Even for "square" or "tall" diagnostic tables, those bounds will be effective only if the tests are rather poor.

#### 4. Experimental results and discussion.

**4.1. Goals and methodology.** The goals of experimentation were three-fold: to verify our deductions about the selection criteria and bounding functions; to determine how much work was expended on finding the optimal solution (as opposed to verifying its optimality); and to obtain an estimate of the size of the largest problems that could be solved in a reasonable amount of time by these techniques.

Faced with a problem of subset search, the algorithm designer usually has a choice of four techniques: dynamic programming, cutting plane techniques, branch-and-bound, and depth-first search. The minimum test set problem has no apparent formulation in the framework of dynamic programming. Cutting plane algorithms require that the minimum test set problem be formulated as an integer linear programming prob-

lem—by transforming it into a set covering problem as described earlier. While cutting plane techniques have proved very effective for the set covering problem (see [1]), the quadratic growth in problem size resulting from the transformation results in excessively large systems of equations. The last two methods are more attractive for our purposes since they both perform an explicit search of the state space as guided by selection criteria and bounding functions. An estimate of the amount of storage needed for the intermediate solutions in a straightforward implementation of branch-and-bound techniques shows that memory requirements are too large to allow the solution of problems of useful size.

Therefore we chose the depth-first search technique (also known as single branch enumeration). It has the advantage of requiring only the storage of a single path in the search space (whereas branch-and-bound techniques may require an exponential number). Also, since the first solution produced by the depth-first search algorithm is the greedy solution, the chosen algorithm has the added advantage of allowing immediate comparisons between selection criteria. Finally, the two methods based on the least-separated pair should work very well in most cases, as the restriction on the choice of candidate tests should drastically diminish the size of the search space. We wrote four Pascal programs, each implementing one of the four selection methods described earlier with its matching bounding function. The global upper bound is provided by the size of the best solution found so far; our discussion lets us hope that this bound is fairly tight for problems of reasonable sizes. The lower bounds are derived from the local contributions of each remaining test and the distance to the solution: by assuming that the tests do not interact, a lower bound on the number of tests needed to complete the partial solution can be found by repeated summing of the tests' contributions.

All four programs preinclude essential tests whenever such are to be found. Although only those tests that are essential with the initial partition must appear in any solution, the backtracking process may give rise to "locally essential" tests, since the process of removing a test from consideration in the subtree may make other tests essential in that region of the state space. As a result, an efficient implementation requires a data structure oriented towards object pairs, keeping track of which available tests distinguish each unseparated pair, so that the search for locally essential tests can be performed by incremental readjustments of the data structure.

More specifically, the data structure consists of a collection of small records, referenced by two list structures. By traversing one type of list, all the tests which separate a given pair of objects are found; traversing the other type of list gives access to all the pairs of objects which a given test distinguishes. There is one list of the first type for each object pair, with direct access to the head of each list provided by an array organization; similarly, there is one list of the second type for each test, again organized as an array. A third array maintains a sorted index to the array of object pairs, used for the least-separated pairs heuristics and for identifying essential tests: its  $i$ th entry is a list of all currently unseparated pairs that can be distinguished by exactly  $i$  of the available tests. This sorted array is maintained by incremental sorting (bubbling changed entries up or down), which is much more efficient than sorting the entire array at each step in the search. When the information heuristic is used, a list of clusters (each cluster being in turn represented as a list) is also maintained, which shows the current breakdown of objects. (In addition, the data structure reflects the availability of tests and the coverage of object pairs; in two of our programs, this was done by means of flags, while the other two kept the data structure tailored to the subproblem at hand.) With this data structure, locally essential tests are identified

instantly; similarly, the selection heuristics can operate in time linear in the number of tests (separation-based) or linear in the size of the input (information-based). This data structure can grow very large (the array of pairs has  $m \cdot (m-1)/2$  lists for  $m$  objects and each list can have up to  $n$  entries) and its memory requirements—rather than the time needed to maintain it—turned out to be the main limiting factor in real world examples. Updating the data structure itself can require time proportional to its size, although it is usually much faster; in particular, the worst-case behavior occurs only near the root of the search tree, that is to say, rarely. The trade-off seems favorable, as a search for locally essential tests would, in absence of the data structure, require time proportional to  $n \cdot m \cdot (m-1)/2$ .

All four programs were run on randomly generated problems and those two programs based on the separation measure were also run on real world examples excerpted from the microbiology literature (in which test sets are regularly published). Nearly all real world examples included variable outcomes (i.e., undefined or inconsistent test values) and a few had multiple-valued (as opposed to binary) tests. All artificial examples had binary tests only, with all entries defined. When the number of tests was small for the number of objects, the randomly generated tables often did not distinguish between all objects; in such a case, a solution was defined as a subset of tests that effected as much discrimination as the full set of tests. The random generator was set so that the two possible outcomes would be exactly balanced over the whole table. (Such problems appear to be harder than those where one outcome is favored, presumably because the even balance introduces a bias in favor of well-splitting tests,<sup>2</sup> which offer more potential for hidden interaction—thereby favoring looser bounds—and also lie in a region where the selection functions have little sensitivity—intuitively, the larger the difference in separation or information there is between the greedy choice and the other available tests, the more likely it is that the greedy choice is in fact globally optimal. We also ran a number of experiments with various skews; all proved noticeably easier to solve than their evenly balanced counterparts.)

The data collected included the size of the greedy solution and of the optimal solution, the initial lower bounds, the number of backtracks needed to reach the solution, the total number of backtracks used, and the total CPU time required. A backtracking step is defined as a reduction in the size of the current partial solution, from size  $i+1$  to size  $i$ ; the backtracking results in making available again all those tests that were children of the test included at level  $i$ , in the exclusion of the test at level  $i$ , and in the inclusion of the test's right sibling (if any), where the siblings are ordered left to right by the chosen selection criterion. (If no right sibling exists, or if it can be bounded—in which case all further right siblings are bounded as well, backtracking will be repeated.)

We decided to use the number of backtracks as a measure of work because this number depends only on the problem and the algorithm, therefore avoiding hardware or software dependencies. However, there clearly is a trade-off between the chosen heuristics with their backtracking behavior and the work needed to maintain a data structure that supports the heuristics. Of course, poor bounding or poor selection can easily lead to an exponential increase in the amount of work, while reasonably complex bounding or selection methods only add another linear (in the size of the input) factor. Our two bounding criteria provide a good example. In the case of the information-based heuristics, the work is concentrated in the selection and bounding steps: each available

---

<sup>2</sup> In view of our previous discussion about bounding, this bias may in fact have favored information-based bounding over separation-based bounding.

test must be evaluated, which requires that the breakdown induced by the test be computed. The time needed to construct the new partition is proportional to the number of objects; thus the bounding and selection step requires time proportional to the product of the number of objects and the number of available tests. In the case of the separation-based heuristics (which we argued should prove weaker for bounding), only simple look-ups of data already in the data structure are required and both selection and bounding can be done in time proportional to the number of available tests. Thus experimentation with both methods should yield valuable insight in the trade-off between the work required at each step and the total amount of work needed.

**4.2. Artificial examples.** In order to study the influence of the number of categories and that of the number of tests on the behavior of the algorithms, four series of experiments were run. In two of the series, the number of categories was kept constant while the number of tests was varied; in one series, the process was reversed; and in the last series, all problems were square with increasing sizes. The sizes varied from 6 to 64, with varying resolution. Twenty-five examples were generated for each size in each series and their results averaged; in all, nearly 2500 examples were run.

The results are presented in graphical form in Figs. 1-7. In all graphs, data points marked with a triangle correspond to results obtained with the information-derived bounding and selection functions; those marked with a square correspond to results obtained with the separation-derived bounding and selection functions; those marked with a cross ( $\times$ ) correspond to results obtained with the least-separated pair heuristics; those marked with a plus sign (+) correspond to results obtained with the single level heuristics; and those marked with a diamond indicate the average size of the solution. Each of the first four figures displays the data collected in one series of experiments in the form of four graphs: the top two show the average total number of backtracks required as well as the average size of the solution, while the bottom two show the percentage of work that was devoted to finding the optimal solution (as opposed to verifying it). The two graphs on the left display the data obtained with the least-separated pair heuristics and those on the right are concerned with the other two heuristics. Fig. 5 illustrates the performance of the greedy methods (it displays the average and largest values of the ratio of the size of the greedy solution to that of the optimal solution) while Fig. 6 illustrates the performance of the bounding methods (it shows the average and largest values of the ratio of the initial lower bound to the size of the optimal solution). Finally, Fig. 7 illustrates the value of the trade-off between local and global work discussed above, showing the ratio of the total CPU time required by the separation-based programs to that required by the more complex information-based programs. Curves were passed through the points in order to make the graphs more legible. (Those curves should not be taken as an accurate depiction of the heuristics' behavior: the data are intrinsically discrete.)

In the first series, all problems had 16 categories, while the number of tests varied from 8 to 64 in steps of 2. Since 16 is a power of 2, it is a transition point for the size of the best possible test set: although problems of that size could admit a solution of 4 tests, such a solution would be hard to find, since it must be composed of 4 perfect tests with no cross-information. For such a solution to exist, a rather large choice of tests must be provided. Thus a solution of 5 tests (if it exists) will be optimal in most cases. Figure 1 shows the experimental results. We note that the average size of a solution stayed above 5 until the number of tests grew large, then decreased slowly; optimal solutions of size 5 were quickly found, presumably because they exist in large numbers. While the information-based bounding did very well (and thus stopped the

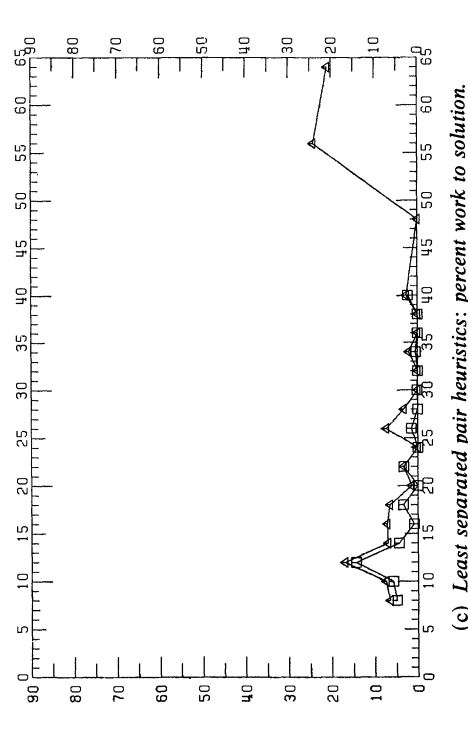
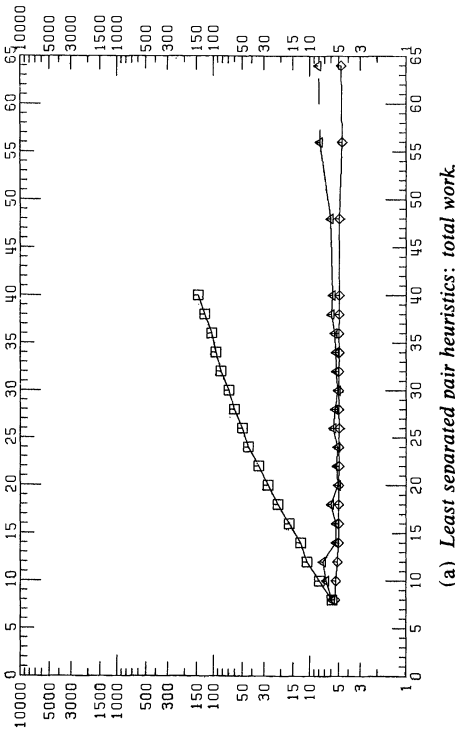
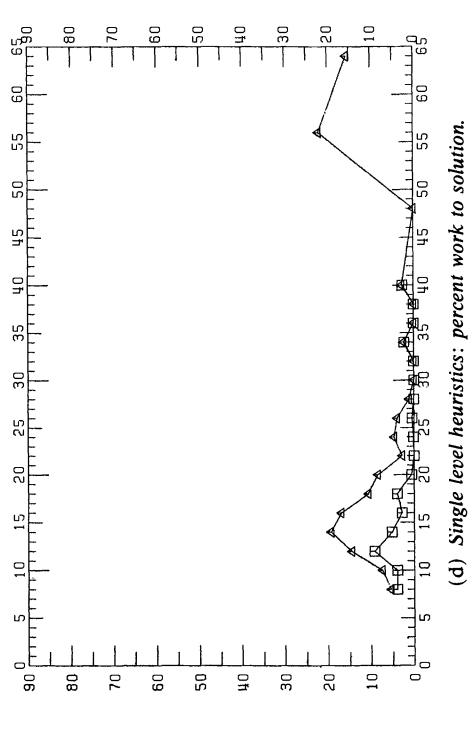
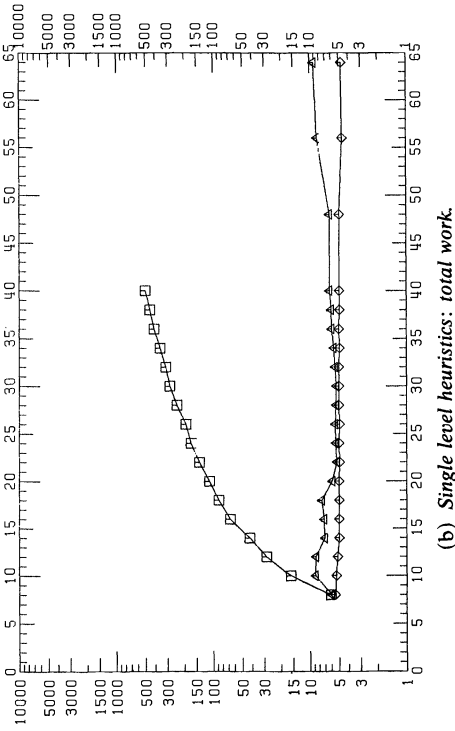
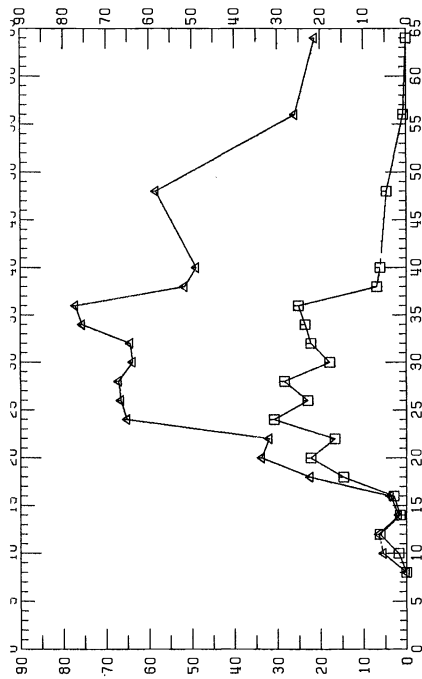
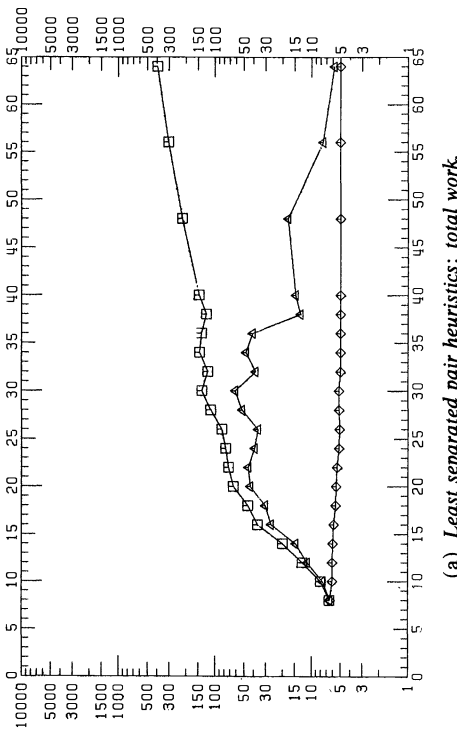
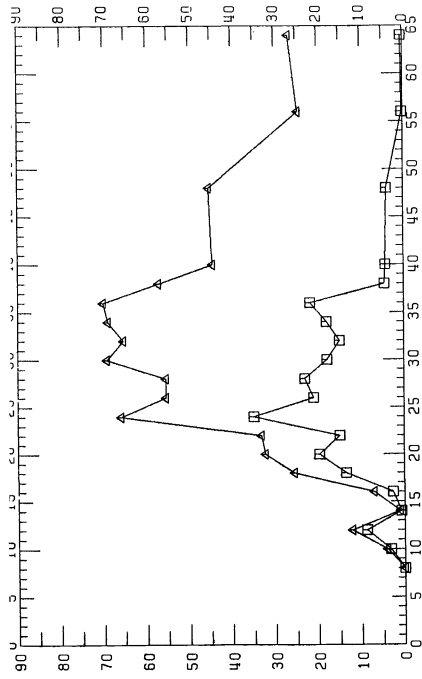
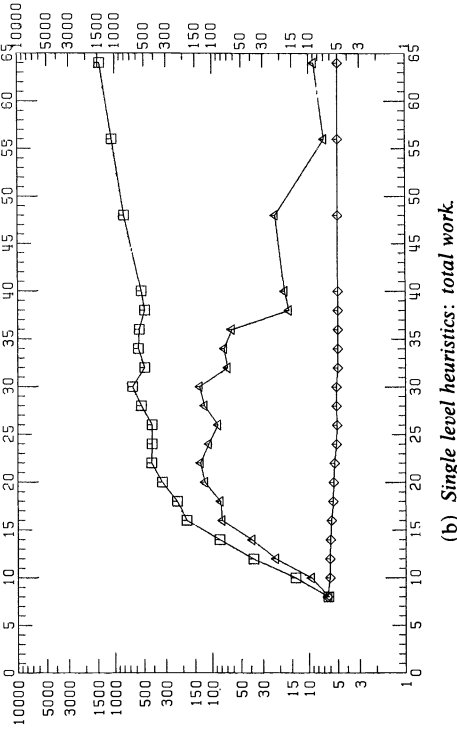


FIG. 1. Results for the problems with 16 objects.



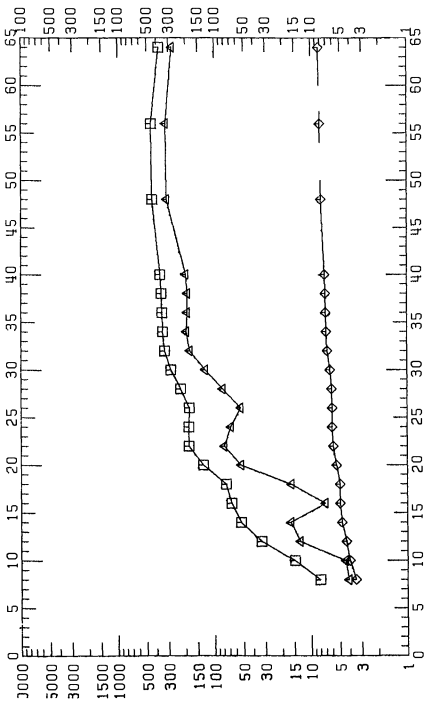
(a) Least separated pair heuristics: total work.

(b) Single level heuristics: total work.

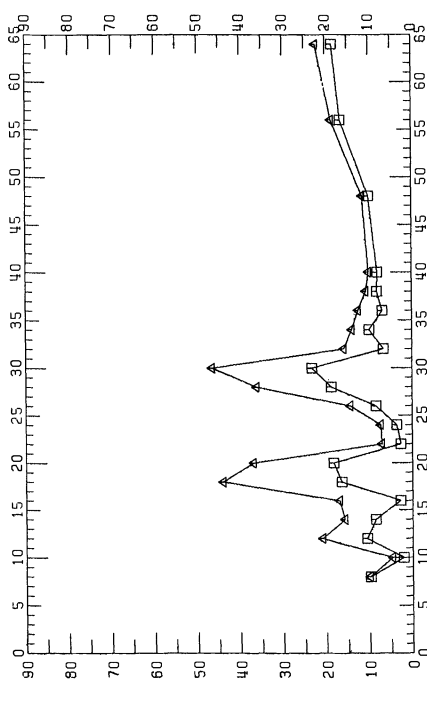
(c) Least separated pair heuristics: percent work to solution.

(d) Single level heuristics: percent work to solution.

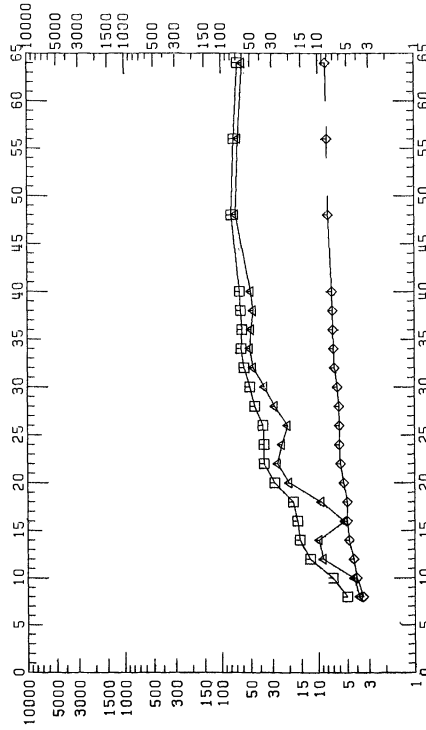
FIG. 2. Results for the problems with 22 objects.



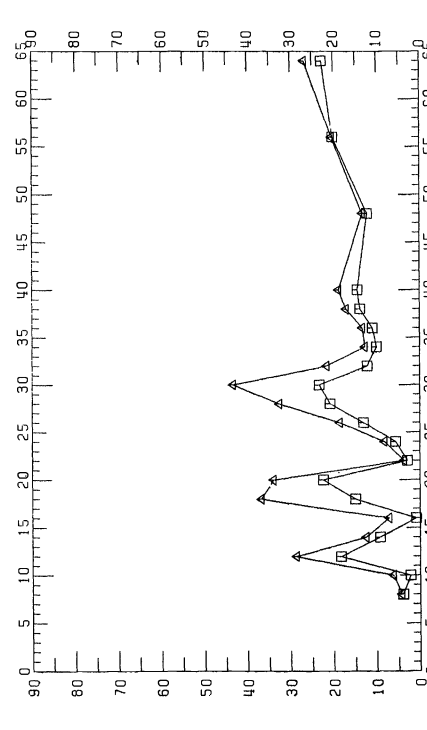
(a) Least separated pair heuristics: total work.



(b) Single level heuristics: total work.



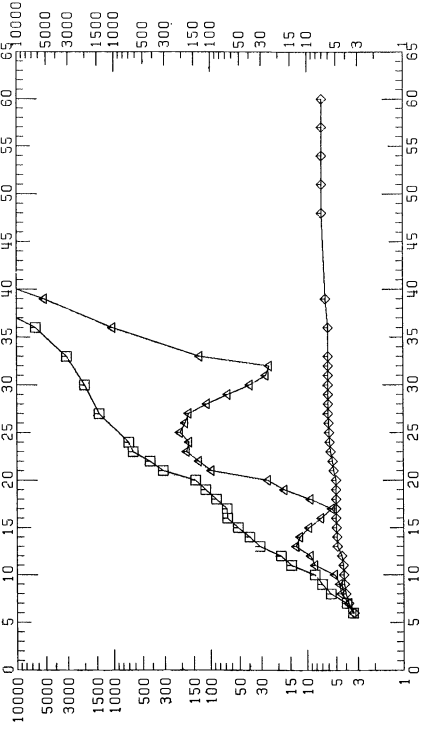
(c) Least separated pair heuristics: percent work to solution.



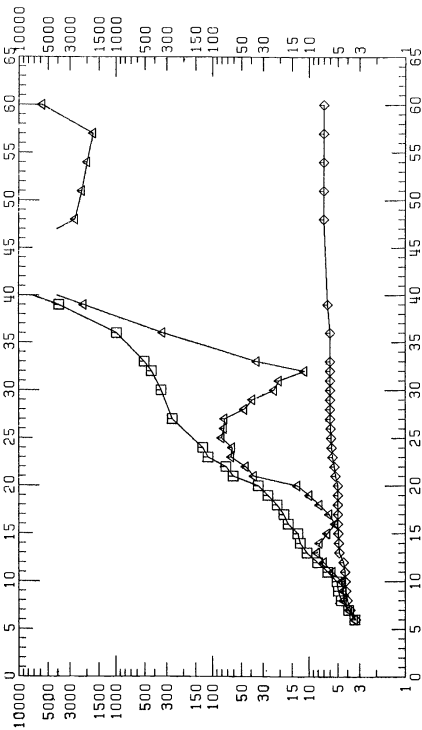
(d) Single level heuristics: percent work to solution.

FIG. 3. Results for the problems with 16 tests.

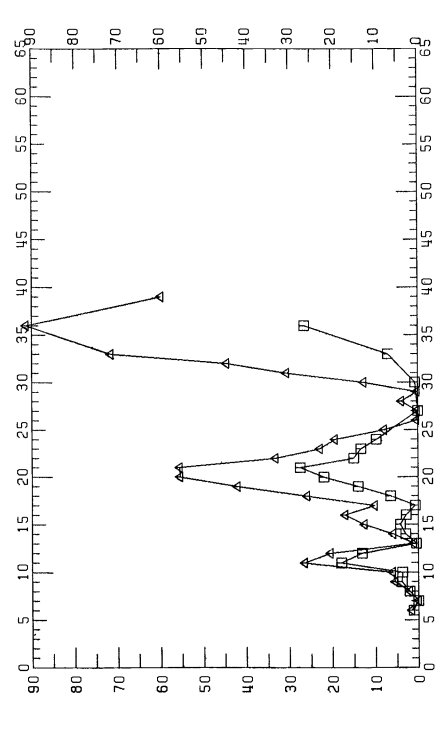




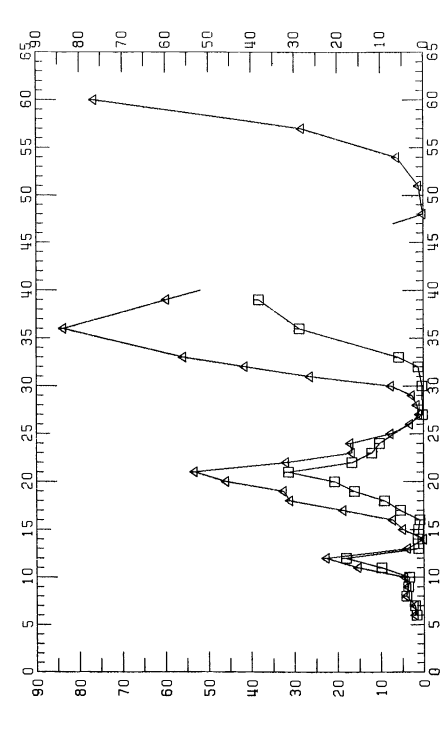
(a) Least separated pair heuristics: total work.



(b) Single level heuristics: total work.

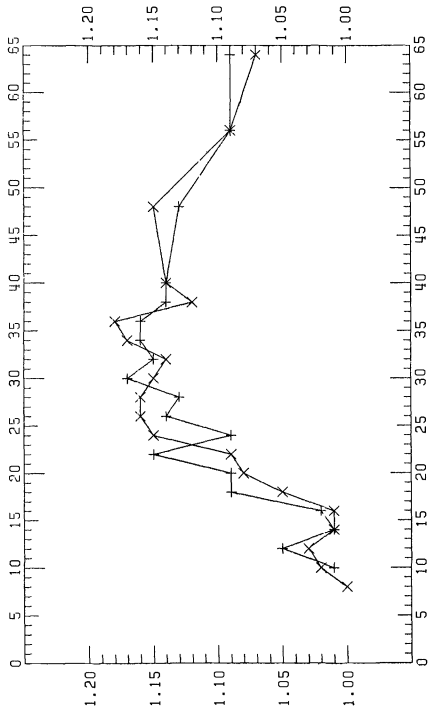


(c) Least separated pair heuristics: percent work to solution.

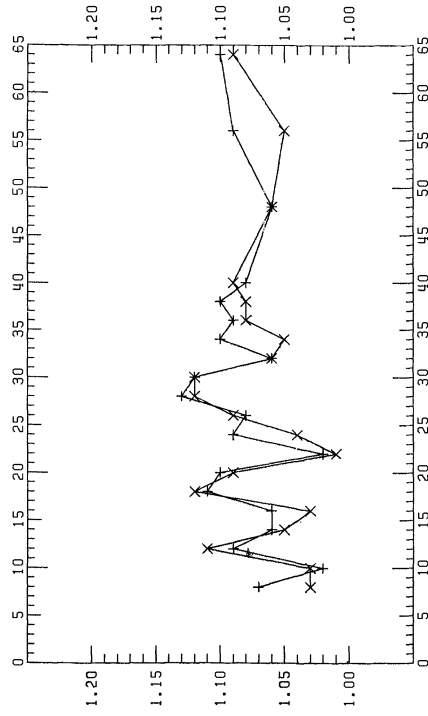


(d) Single level heuristics: percent work to solution.

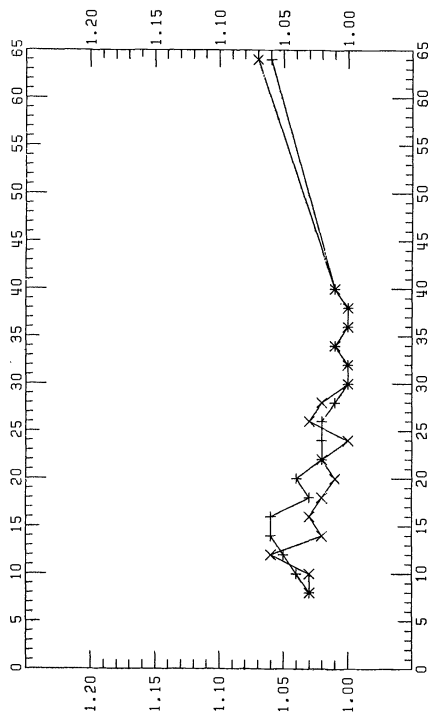
FIG. 4. Results for the square problems.



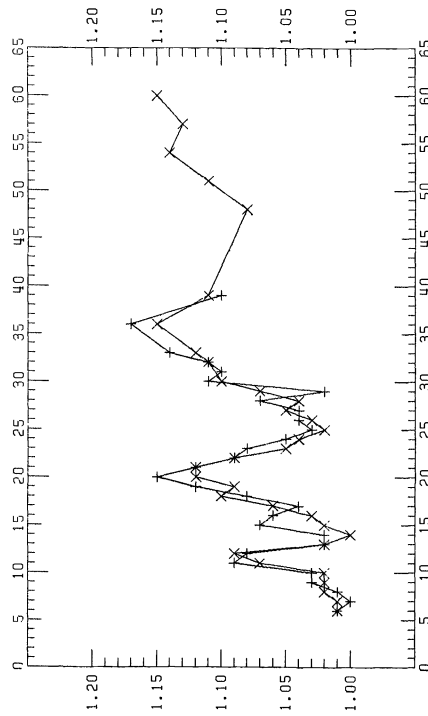
(a) 16 objects, varying number of tests.



(b) 22 objects, varying number of tests.

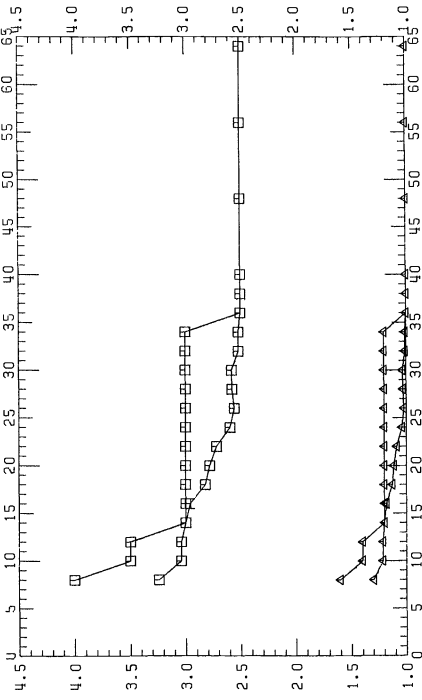


(c) Square problems.

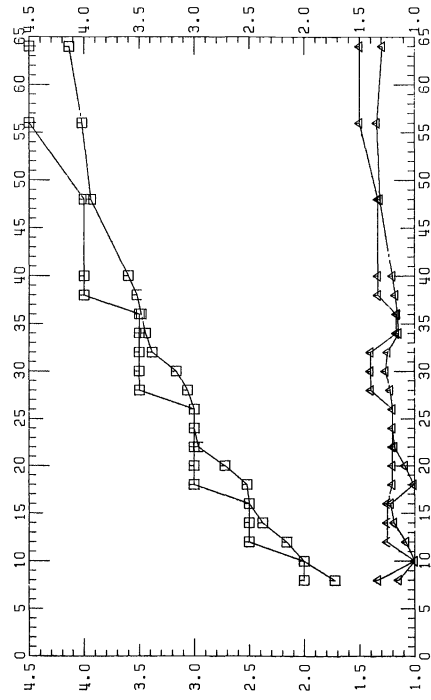


(d) 16 tests, varying number of objects.

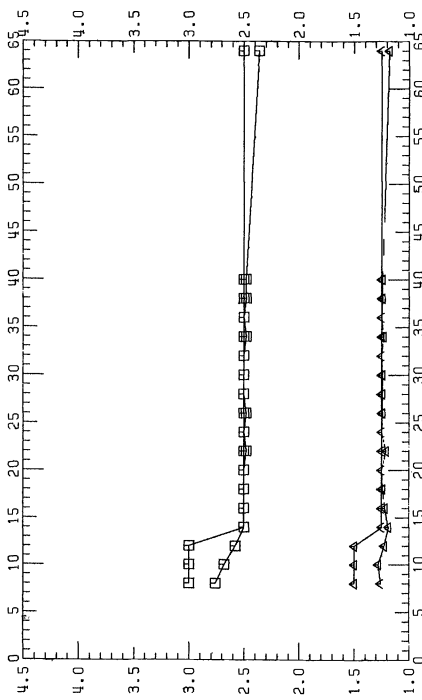
FIG. 5. The ratio of the size of the greedy solution to that of the optimal solution.



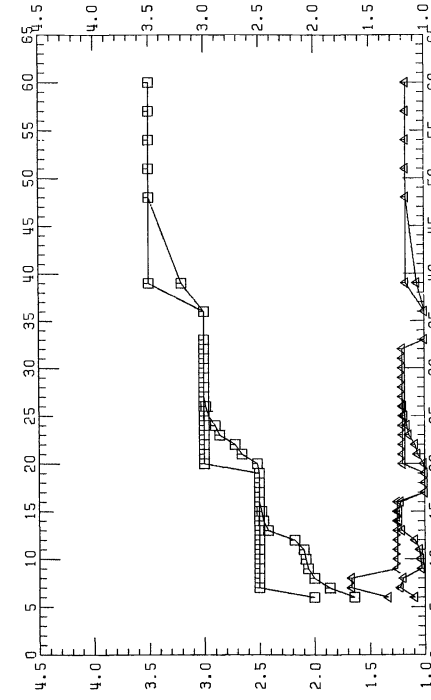
(a) 16 objects, varying number of tests.



(b) 22 objects, varying number of tests.

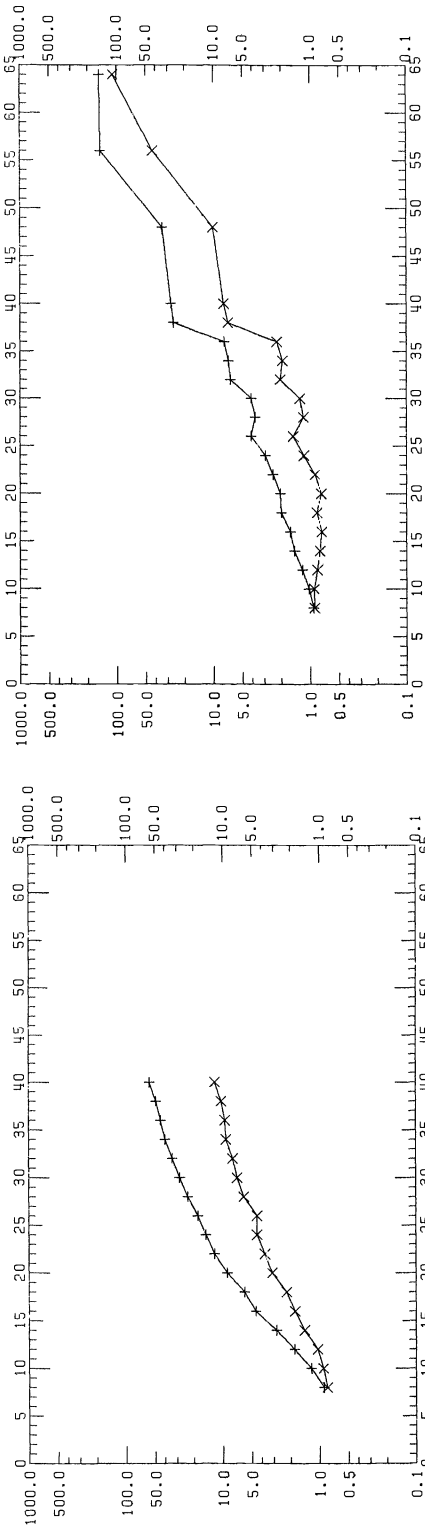


(c) Square problems.

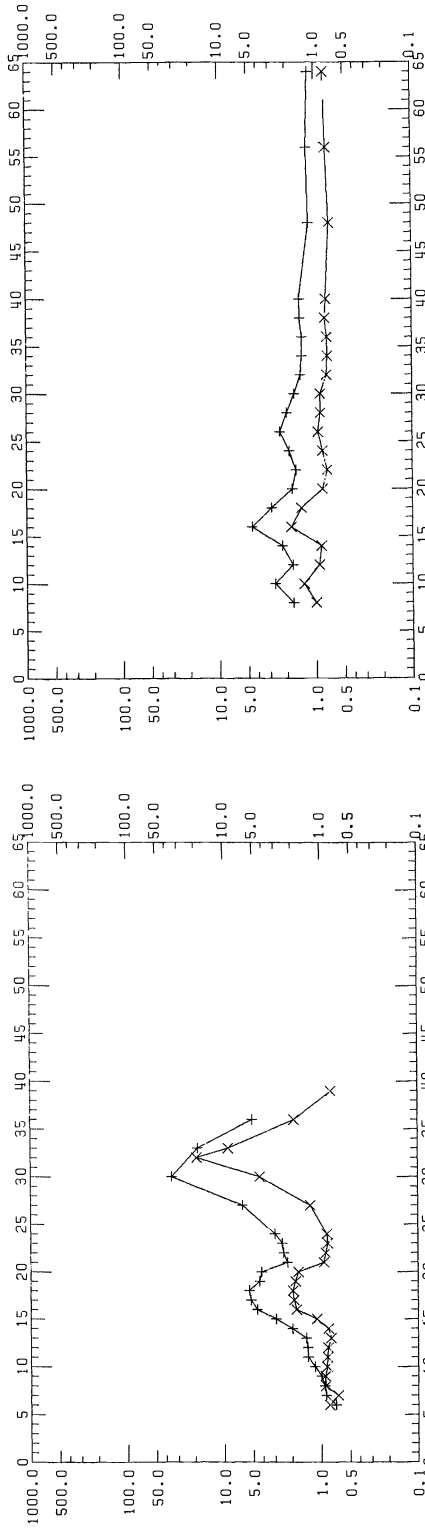


(d) 16 tests, varying number of objects.

FIG. 6. The ratio of the size of the optimal solution to the initial lower bound.



(b) 22 objects, varying number of tests.



(c) Square problems.

(d) 16 tests, varying number of objects.

(c) Square problems.

(d) 16 tests, varying number of objects.

FIG. 7. Ratios of execution times: separation-guided programs to information-guided programs.

search shortly after the solution was found), the separation-based one did very poorly—because, when the choice is large, many of the tests are well-splitting. Since information bounds were tight, the reduced branching factor associated with the least-separated pair heuristics did not play a significant role, while that role is clearly exemplified by the two programs using the separation bounds.

In the second series, all problems had 22 categories; other parameters were as in the first series. With 22 categories, the best possible test set has size 5; such a solution is not as difficult to realize as in the first series, since it is well above  $\log_2 22 \cong 4.46$ . On the other hand, the bounding can be decisive only if a solution of 5 tests is reached; the test interactions that make a 6-test set optimal will not be reflected strongly enough in the bounds. Of course, solutions of size 6 abound, so that all heuristics will find one almost instantly, while solutions of 5 tests will be considerably more difficult to find until the choice of tests becomes sufficiently large. Experimental results are shown in Fig. 2. We note that, as expected, all heuristics perform similarly when the choice of tests is small; a solution of 6 tests is found almost immediately, but bounding is ineffective. As the choice grows, all four heuristics find a solution of 5 tests early; the information-bounded programs then stop shortly, while the separation-bounded ones go on and explore nearly the full tree. The role of the reduced branching factor of the least-separated pair heuristics is as in the first series, minor for the information-bounded programs and major for the other two.

In the third series, the number of tests was kept at 16, while the number of categories was varied from 8 to 64 in steps of 2. As the number of categories increases, so does the size of the theoretically minimal solution; the size of the best realizable solution increases even faster, since the choice of tests becomes relatively small. With a large number of objects, the tests interact significantly, thereby impairing the effectiveness of our bounding methods. On the other hand, the probability that a pair is separated by only one or two tests significantly increases. Finally the tests, with so many entries in each column, are well differentiated (the probability of any two tests being identical—or even having the same number of ones and zeros—is very low). The results are shown in Fig. 3. Since the average number of tests separating a pair is low, the least-separated pair heuristics did much better than the other two in keeping the amount of search low. No significant difference can be observed between information-based bounding and separation-based bounding, presumably because the large amount of interaction between tests renders bounding ineffective. For small to medium numbers of categories, the situation is more complex. We conjecture that the observed behavior is a version of the trade-off between bounding and finding solutions that the fourth series of experiments brought to light.

The fourth series had square problems, with a size increasing from 6 to 60. (Only the best of the four heuristics was used for the larger sizes; indeed,  $40 \times 40$  appeared to be the practical limit for the separation-bounded heuristics.) Since, in a square problem, the choice of tests is relatively restricted, we would expect a behavior similar to that exhibited in the first half of Fig. 3. Experimental results are shown in Fig. 4. Least separated pair heuristics hold a slight edge over the other two and the information-guided heuristics vary in performance between much better than the separation-guided—when the solutions become harder to find and this provide for better bounding—and almost as poor—when the solutions become easier to find and thus cause much looser bounds. The graphs dramatically illustrate the trade-off between tight bounding and ease in finding solutions: the harder a solution is to find, the easier it will be to prune the remaining branches, yet, if the solution is too hard to find, most of the tree will be explored just looking for it.

The data collected about the size of the greedy solutions confirmed that all four heuristics are good selection criteria. No greedy solution ever exceeded the optimal by more than 50%; on the average, greedy solutions were only 6% to 7% larger than optimal. As expected from our discussion of the separation and information measures, the two performed equally well (the average ratios were always within one percent of each other, which is not a statistically significant difference over 25 experiments); the two methods relying on the least-separated pair heuristic showed a slight advantage, presumably due to the narrower focus they impart on selection. Fig. 5 presents the results (the average ratios of the size of the greedy solution to the size of the optimal solution) in the form of four graphs (one for each series of experiments).

We chose to illustrate the behavior of the bounding methods by collecting statistics on the ratio of the size of the optimal solution to the initial lower bound (as derived by using separation or information measures). The average and worst-case values of this ratio are plotted in Fig. 6 in four graphs (one for each series of experiments). As expected from our discussion, the lower bounds based on information are consistently better than those based on separation. In particular, while the separation-derived bounds worsen with increasing number of objects, no such trend is apparent for the information-derived bounds.

While Figs. 1–4 and Fig. 6 demonstrate that information-based bounding is superior to separation-based bounding, they do not allow us to decide whether the better bounding was worth the added complexity. Fig. 7 provides the necessary information, showing the ratio of the total CPU times required by the two types of heuristics for the four series of problems. (Ratios are used because they minimize the influence of programming style—an important feature since the two programs using least-separated pair heuristics were written by one author and the other two by the other author; in this context, it is reassuring to note that the two curves in each graph are always consistent.) In all cases, despite their simplicity (and consequent speed of execution), the separation-based programs took much longer than the information-based ones for all but the smallest problems, allowing us to conclude that the trade-off is clearly in favor of more local work. As might be expected, the single level heuristics, relying exclusively on bounding, benefitted more from the added local work than the least-separated pair heuristics, which can count on reduced branching factors.

Overall, the experimental results confirmed our evaluation of the selection criteria and the bounding functions. All four selection criteria appear equal. Least separated pair heuristics are vastly superior to the other two when efficient bounding is not possible (as when the number of categories grows large with respect to the number of tests), due to their small branching factor. Information-bounded heuristics are much better than the other two when the optimal solution is found early and efficient bounding can be done (as when the number of tests grows large with respect to the number of categories). In all cases, the most efficient program used the least-separated pair heuristic with information-based bounding. The largest solvable problems had sizes of around 40 by 40, although a single parameter could be increased well beyond that.

**4.3. Real world examples.** In view of the results obtained with artificial examples, a certain optimism is justified as regards real world problems. Such problems tend to have many essential tests; moreover, they are often composed of a small number of well-splitting tests and a large number of rather poor (possibly simple) tests. With such a structure, selection criteria should perform well, as several microbiology researchers have found [16], [19]. Moreover, many pairs will be separated by only a few tests, so that the least-separated pair heuristics should keep the branching factor quite low.

We used the separation criterion only, as the information criterion is not easily adapted to problems with variable test outcomes. (Being based on clusters, it requires that the size of all clusters established by the inclusion of tests be recorded. Since variable outcomes require that some objects be placed in more than one cluster, the composition of each cluster must also be recorded, in order to eliminate common subsets. All of this adds up to excessive bookkeeping and enormous storage requirements.) Table 2 presents a synopsis of the results on eight examples from the microbiology literature; in that table, LSP stands for the least-separated pair heuristic with separation bounding while SEP stands for separation as a single level heuristic. The data presented include the size of the problem, the size of the optimal solution and that of the greedy solutions found by each heuristic, the number of essential tests, the total number of backtracks used by each heuristic in obtaining the optimal solution, and the percentage of work used to discover—as opposed to verify—the optimal solution (again, work is taken to be the number of backtracks used to obtain the optimal solution *after* having obtained the greedy solution, so that the work turns out to be zero if the greedy solution proved optimal). Several remarks are in order. First, many of the tests incorporated in an optimal solution were essential, showing how important it is for an algorithm to include essential tests whenever possible. Secondly, the optimal solution was almost always found immediately, confirming the power of the greedy heuristics in real world examples. Third, some of the problems run were four times larger than the largest artificial examples attempted, yet ran almost a hundred times faster. Finally, the program using the least-separated pair heuristic with separation bounding never used more than a minute of CPU time running on a VAX11/780 computer. (For comparison, the greedy heuristic used in [14] on the *Pseudomonas* example took 2.8 minutes on an IBM360/50, while our program took 1.9 seconds to guarantee an optimal solution for the same problem—an enormous difference that we attribute mostly to our careful choice of data structures.)

TABLE 2

Isolates	Problem size		Solution			Number Ess. tests	Backtracks		% Work to sol.	
	Categories	Tests	Opt.	LSP	SEP		LSP	SEP	LSP	SEP
Actinomadura <sup>1</sup>	11	32	5	5	7	1	7	20	0.0	67.2
Cyanobacteria <sup>2</sup>	106	19	16	16	16	15	16	16	0.0	0.0
Enterobacteria <sup>3</sup>	7	20	7	7	7	4	7	7	0.0	0.0
<i>Pseudomonas</i> [14]	27	21	8	9	8	2	16	21	21.7	0.0
SMA12 kit <sup>3</sup>	142	12	12	12	12	12	12	10	0.0	0.0
Streptococci <sup>3</sup>	36	32	25	25	25	25	25	25	0.0	0.0
Streptococci [18]	50	122	36	36	36	31	43	43	0.0	0.0
Yeasts <sup>4</sup>	98	56	16	16	17	5	144	1556	0.0	0.2

<sup>1</sup> M. Goodfellow et al., *Numerical taxonomy of Actinomadura and related Actinomycetes*, J. Gen. Microbiol., 112 (1979), pp. 95-111.

<sup>2</sup> R. Rippka et al., *Generic assignments, strain histories and properties of pure cultures of cyanobacteria*, J. Gen. Microbiol., 111 (1979), pp. 1-61.

<sup>3</sup> E. W. Rypka, Private communication, Lovelace Medical Center, Albuquerque, NM, 1981.

<sup>4</sup> J. M. Belin, *Identification of yeasts and yeast-like fungi I: taxonomy and characteristics of new species described since 1973*, Can. J. Microbiol., 27 (1981), pp. 1235-1251.

Overall our results confirmed our optimism about real-world problems and justify an even more positive attitude: with a judicious trade-off between time and space, it will be possible to solve even larger examples without major modifications. If some better bounding method can be developed—and preliminary research indicates that

this is within reach, using linear programming with merged constraints—then the time traded off will easily be regained. In fact, this indicates that a hybrid algorithm, partaking of both depth-first search and branch-and-bound techniques, may be best.

**5. Conclusion.** We have reviewed the methods proposed in the literature for dealing with the minimum test set problem. We have evaluated the proposed selection heuristics and characterized their worst-case behavior. We have presented the results of extensive experimentation with four backtracking algorithms. Our results confirm that existing selection heuristics are quite satisfactory; they also indicate that the best backtracking method involves a heuristic which uses the information criterion for selecting tests and deriving bounds and relies on the least-separated pair heuristic to keep branching factors low. Experimentation with real world problems showed the importance of preinclusion of essential tests; it also gave grounds for optimism since, despite the known NP-hardness of the general problem, an inferior version of our programs solved large problems in a very short time.

Much work remains to be done. Better bounding methods must be sought, which apply even when variable outcomes are present. An extension of the information criterion is the obvious first step. Beyond that, the use of the linear programming subproblem for deriving bounds (as used for the set covering problem in [10]) appears promising; although the size of the linear programming problem grows faster than that of the original problem, one can diminish it by merging some conditions, thereby relaxing the constraints (indeed, merging all conditions into one gives us the separation criterion). In addition, the integer linear programming approach with cutting plane methods is worth investigating, despite the size of its formulation. Most importantly, ways of incorporating measured amounts of redundancy into the solution must be sought: the minimum test set is, by definition, a fragile entity. While redundancy can easily be incorporated through simple methods (such as insisting that each pair be separated by at least two tests, whenever possible), the more complete risk model of pattern recognition [5] provides a suitable framework for more sophisticated methods.

**Acknowledgments.** The authors wish to thank the referees for many helpful comments; in particular, we are heavily indebted to the second referee for correcting our intuition on the worst-case behavior of the heuristics and suggesting the use of Johnson's results on set covering.

#### REFERENCES

- [1] E. BALAS AND A. HO, *Set covering algorithms using cutting planes, heuristics, and subgradient optimization: a computational study*, Math. Programming, 12 (1980), pp. 37-60.
- [2] L. W. BEARNSON AND C. C. CARROLL, *On the design of minimum length fault tests for combinational circuits*, IEEE Trans. Comp., TC-20 (1971), pp. 1353-1356.
- [3] H. Y. CHANG, *An algorithm for selecting an optimum set of diagnostic tests*, IEEE Trans. Electr. Comp. EC-14 (1965), pp. 706-711.
- [4] H. Y. CHANG, E. MANNING AND G. METZE, *Fault Diagnosis of Digital Systems*, John Wiley, New York, 1970.
- [5] P. A. DEVIJVER AND J. KITTLER, *Pattern Recognition: A Statistical Approach*, Prentice-Hall, Englewood Cliffs, NJ, 1982.
- [6] M. R. GAREY AND D. S. JOHNSON, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, W. H. Freeman, San Francisco, 1979.
- [7] D. S. JOHNSON, *Approximation algorithms for combinatorial problems*, J. Comp. System Sci., 9 (1974), pp. 256-278.
- [8] A. D. KORSHUNOV, *The length of minimum tests for rectangular tables I*, Cybernetics, 6 (1970), pp. 723-733.
- [9] ———, *The length of minimum tests for rectangular tables II*, Cybernetics, 7 (1971), pp. 1-14.



- [10] C. E. LEMKE, H. M. SALKIN AND K. SPIELBERG, *Set covering by single-branch enumeration with linear programming subproblems*, *Oper. Res.*, 19 (1971), pp. 998-1022.
- [11] P. N. MARINOS, *Derivation of minimal complete sets of test-input sequences using Boolean differences*, *IEEE Trans. Comp.*, TC-20 (1971), pp. 25-32.
- [12] B. M. E. MORET, *Decision trees and diagrams*, *Comp. Surveys*, 14 (1982), pp. 593-623.
- [13] P. M. NARENDRA AND K. FUKUNAGA, *A branch-and-bound algorithm for feature subset selection*, *IEEE Trans. Comp.*, TC-26 (1977), pp. 917-922.
- [14] S. I. NIEMELA, J. W. HOPKINS AND C. QUADLING, *Selecting an economical binary test battery for a set of microbial cultures*, *Can. J. Microbiol.*, 14 (1968), pp. 271-279.
- [15] R. W. PAYNE AND D. A. PREECE, *Identification keys and diagnostic tables: a review*, *J. Royal Statist. Soc. A*, 143 (1980), pp. 253-292.
- [16] R. W. PAYNE, *Selection criteria for the construction of efficient diagnostic keys*, *J. Statist. Planning & Inf.*, 5 (1981), pp. 27-36.
- [17] A. RESCIGNO AND G. A. MACCACCARO, *The information content of biological classifications*, in *Information Theory: Fourth London Symposium*, C. Cherry, ed., Butterworths, London, 1961, pp. 437-445.
- [18] E. W. RYPKA, W. E. CLAPPER, I. G. BROWN AND R. BABB, *A model for the identification of bacteria*, *J. Gen. Microbiol.*, 46 (1967), pp. 407-424.
- [19] E. W. RYPKA, *Truth table classification and identification*, *Space Life Sciences*, 3 (1971), pp. 135-156.
- [20] E. W. RYPKA, L. VOLKMAN AND E. KINTER, *Construction and use of an optimized identification scheme*, *Laboratory Medicine*, 9 (1978), pp. 32-41.
- [21] V. A. SLEPYAN, *Minimal test length for a certain class of tables*, *Discretnyi Analiz*, 23 (1973), pp. 59-71. (In Russian.)
- [22] W. R. WILLCOX AND S. P. LAPAGE, *Automatic construction of diagnostic tables*, *Comput. J.*, 15 (1972), pp. 263-267.
- [23] W. R. WILLCOX, S. P. LAPAGE AND B. HOLMES, *A review of numerical methods in bacterial identification*, *Antonie van Leeuwenhoek*, 46 (1980), pp. 233-299.

## COMPUTING ENVIRONMENTS FOR DATA ANALYSIS I. INTRODUCTION\*

JOHN ALAN McDONALD† AND JAN PEDERSEN†

**Abstract.** This is the first in a series of papers on aspects of modern computing environments that are relevant to statistical data analysis. We argue that a network of graphics workstations is far superior to conventional batch or time-sharing computers as an environment for interactive data analysis. The first paper in the series provides a general introduction and motivation for more detailed considerations of hardware and software in subsequent parts.

**Key words.** data analysis, interactive graphics, workstations, programming environments

**1. Introduction.** Statistics has long been thought of as *applied mathematics*. Certain parts of it, especially data analysis, could easily be viewed as *applied computation*. In this context, different research issues assume importance, in particular, the design and implementation of *computing environments for data analysis*.

Analyzing data requires computing, whether with paper-and-pencil or with a Cray super computer. The *computing environment* determines what sorts of statistical methods are practical. More importantly, the statistician's unconscious assumptions, or *mental model* of the computing environment determines the kinds of new statistical methods that are likely to be invented.

Most current research in statistical computing is based on a *batch processing* model of computing environments that was appropriate twenty years ago. With a few exceptions, statisticians have not addressed the implications of current and future developments in scientific computing environments.

Although no single number can be a complete summary, it is probably fair to say—for purposes of discussion—that the performance per dollar of computing equipment has increased by a factor of a thousand over the past twenty years. The large quantitative change implies a qualitative change in how computers can be used. For comparison, the magnitude of the difference is larger than that between walking and flying—if one could fly for the same price as one could walk.

This is the first part of a series of papers that discuss *local networks of graphics workstations* as environments for statistical computing. In this, the first part, we provide general background and motivation. Future parts (see this issue, pp. 1013–1021 for part II) will describe relevant aspects of computing hardware and software, in particular present and future scientific programming environments.

### 2. Trends in scientific computing environments.

**2.1. Pencil and paper.** Most statistical methods originated in the pencil-and-paper computing environment of 50 years ago (perhaps we should say the pencil-paper-statistical-table-hand-calculator environment) and were designed with its particular constraints and limitations in mind. Although some statisticians were prodigiously good at arithmetic, there was an obvious practical limit on the volume of computation feasible for a given analysis. This implied statisticians were restricted to fairly small

---

\* Received by the editors June 5, 1984. This research was supported by the National Science Foundation under a Mathematical Sciences Postdoctoral Research Fellowship, the Office of Naval Research under contract N00014-83-K-0472, and grant N00014-83-G-0121, and U.S. Army Research Office under contract DAAG29-82-K-0056.

† Department of Statistics and Computation Research Group, Stanford Linear Accelerator Center, Stanford University, Stanford, California 94305.

data sets and simple methods requiring relatively little computing, by today's standards. The result was an emphasis on idealized models for data whose primary motivation was the availability of closed form solutions or convenient asymptotic approximations.

Balanced against these shortcomings was the flexibility of the pencil-and-paper environment. Since the statistician was immediately involved in every stage, the analysis could be easily adapted to take advantage of contextual information and common sense. In addition, the time between the conception of a qualitatively new method and its implementation was essentially zero.

**2.2. Batch processing.** Twenty years ago, when computing machinery was a scarce resource, great effort went into making efficient use of machine time at the expense of the time of those who used it. A university computer center would typically have a single *mainframe* computer (e.g. an IBM 360), which could easily be in the million dollar range. To use the computer, users submitted programs, usually in the form of decks of punched cards, to the computer's job queue. A batch operating system controlled the execution of jobs, typically processing one job at a time, from start to finish. The results appeared, hours or days later, on reams of awkwardly sized line-printer paper. We will refer to this type of computing environment as *batch processing*.

Although batch computing environments are, fortunately, nearly extinct (at least for scientific computing), it is important to keep the batch processing model (punch card to batch processor to line printer) in mind. Not only are the standard statistical packages cast in this form, for example SAS, BMDP, and SPSS; much of state-of-the-art statistical computing has this model as an implicit underlying assumption.

The batch processing model limits the sort of software that can be developed. The two most confining aspects of batch processing are the long turn-around time for the simplest computations and the even longer time to make minor changes to a program. The result is a style of data analysis based on the concept of a *statistical package*, a program, or collection of programs, implementing a small set (10 to 50) of fixed operations that can be applied to sets of data.

Because the time to modify a program is so long, each operation is essentially fixed, with perhaps a few options that can be set before execution. The set of provided operations in a package is usually small and new operations are difficult or impossible to add. There is, therefore, a tendency to force data sets of great individuality into models that are not really appropriate, just because they happen to be in the available packages. Prior knowledge, both about complex internal structure and about the external context of the data, must often be ignored. The lack of flexibility is pervasive and imposes severe limits on the creativity in solving the new problems posed by each new data set.

Because the execution turn-around time is so long, each operation is designed as a more-or-less complete analysis. Intermediate choices in an analysis must be automated; they cannot be based on interpretations of intermediate results. These automated decisions are often a poor substitute for straightforward interactive methods. As a result, it is not uncommon for packaged routines to attempt to anticipate every possible contingency, generating volumes of output of which only some small part is actually relevant.

An argument can be made that data analysis has regressed by entering the computer age. Although the computing constraint has been largely eliminated, statisticians have lost the flexibility inherent in the pencil-and-paper mode of analysis. John Tukey drives this point home in the postscript to *EDA*, a book written almost entirely from the pencil-and-paper point of view.

This book focusses on paper and pen(cil)—graph paper and tracing paper when you can get it, backs of envelopes if necessary—multicolor pen if you can get it, routine ballpoint or pencil if you cannot. Some would say that this is a step backward, but there is a simple reason why they cannot be right: much of what we have learned to do to data can be done by hand—with or without the aid of a hand-held calculator—*LONG BEFORE* one can find a computing system—to say nothing of getting the data entered. Even when every household has access to a computing system, it is unlikely that “just what we would like to work with” will be easily enough available. Now—and in the directly foreseeable future—there will be a place for hand calculation (Tukey (1977, p. 663)).

The challenge before us is to recover the flexibility that has been lost, but also to augment it with the considerable computing power that should be available at our finger tips.

**2.3. Time-sharing.** As computer prices dropped there was less concern for efficient use of machine time and more concern for the efficient use of people time. This led to the development of *time-sharing operating systems* and a trend towards smaller computers (which were often only smaller in the sense that they were owned and used by a smaller number of people).

A time-shared mainframe communicates with its users through many remote terminals and provides a few centralized peripheral devices (e.g. a line printer, disk drive, tape drive, etc.). The mainframe runs a multi-user operating system and users rent time, sharing the central processor.

Some statistical packages, e.g. Minitab, take advantage of time shared computing to provide some degree of interaction, but most statistical computing has changed only to the extent that it is easier to submit batch jobs.

*Statistical programming languages*, like S (Becker and Chambers (1984a), (1984b)) and ISP (Donoho, Huber, Ramos, and Thoma (1982)), make better use of the possibilities of interactive computing. A *statistical language* provides tools for combining primitive functions and data structures to create new, higher-level functions and data structures. A statistical language is therefore extensible, in a way that a statistical package is not. That is, new operations can be defined and easily integrated into the language. In a package the procedures stand alone and communication between procedures is awkward or impossible, while in a language the operations are designed to work together.

**2.4. Networks of graphics workstations.** Networks of graphics workstations are the logical next step in providing a friendly and more powerful computer environment.

We begin with a few definitions:

A *workstation* is a complete computer that is used by a single person. The workstations discussed in this paper may be thought of as roughly equal—in speed of computation, amount of memory, and so forth—to a VAX (somewhere in the range covered by the 11/730 to 11/790). In other words, they do not have the limitations that one might associate with present day personal or home computers.

A *graphics workstation* includes, in addition, a high resolution *bitmap* display and *graphical input devices* (e.g. mouse, trackerball, joystick) as integral parts (Beatty (1983), Foley and Van Dam (1982), Newman and Sproull (1979)). This combination permits a natural, graphical language for communication between user and computer, which, in turn, leads to the design of computing environments that are more powerful, more sophisticated, and easier to use.

Examples are workstations built by Apollo, Sun, Chromatics, Symbolics, Ridge, and Masscomp, which are discussed in more detail in part II of this paper. Briefly, some of the properties of a graphics workstation appropriate for use in statistics are:

- *Hardware.*
  - A central processor equivalent to a Vax (500,000 to several million instructions per second).
  - Fast floating point equivalent to a Vax (500,000 to several million floating point operations per second).
  - Several megabytes of physical memory (RAM).
  - virtual memory (a large address space).
  - Tens to hundreds of megabytes of disk storage.
  - Bitmap graphics display(s), at a resolution of approximately  $1,000 \times 1,000$ , in black-and-white and/or color.
  - Auxiliary processors, including array processors for rapid numerical computation and graphics processors for fast picture drawing.
  - Graphical input devices such as a mouse.
  - Both hardware and software to support communication over a high speed (several megabits per second) local network (usually some flavor of Ethernet).
- *Programming environment.* A user controls the action of a computer through a set of (software) tools—the *programming environment*. We are intentionally blurring the distinction between direct, interactive (interpreted) commands and the more complex deferred instructions produced by writing, compiling, linking, and loading a program. The basic alternatives in graphics workstations are a conventional operating system, such as Unix (Deitel (1983), Kernighan and Mashey (1981), UNIX (1978)), or an integrated programming environment, such as Interlisp-D (Sheil (1983), Teitleman and Masinter (1981)) or Smalltalk (Goldberg (1983), (1984), Goldberg and Robson (1984), Krasner (1983)). Examples of facilities provided by a programming environment are (Deutsch and Taft (1980)):
  - Programming languages.
  - A file system.
  - An (intelligent) display editor.
  - Interactive debugging tools.
  - A windowing system for effective use of the bitmap display.
  - Support for graphical interaction.

A *network* is a combination of hardware and software that permits independent workstations and their users to communicate—in other words, transfer data—rapidly and share resources (such as tape drives, printers, etc.) (Green (1982), Tannenbaum (1981a), (1981b)). We are most concerned here with *local networks* (connecting computers within a building or a campus), in contrast to *wide area networks* (connecting machines across the country or around the world).

In addition to graphics workstations, the network might include some of the following:

- Small personal computers, such as the Apple Macintosh.
- Several-user midi-computers, such as a VAX 11/750.
- Many-user mainframes, such as DEC 20.
- Special purpose number crunching engines, such as a Floating Point Systems array processor.
- Super computers, such as a Cray.
- Printers.
- Fileservers—computers with large amounts (at least several gigabytes) of disk storage used to store files and databases shared by some or all the machines in the network.

- Tape drives.
- Gateways to other local networks.
- Gateways to wide-area networks, such as Arpanet.

The distinction between personal computers, graphics workstations, and midicomputers may disappear as graphics workstations become more powerful than midicomputers and no more expensive than personal computers.

It is important to emphasize that this series of papers describes computing environments that will not be fully realized for five to ten years. Although graphics workstations are commercially available, they range in price from \$15,000 to \$150,000, which is too expensive for there to be one on every desk. Also, even if the hardware is available, software for statistical applications has yet to be written.

Another serious problem is the lack of standardization of network protocols and internal software that is necessary to allow communication and portability of software between machines of different manufacturers.

There are good reasons to expect the situation to change rapidly. For example, the recently announced Apple Macintosh has many of the desirable features of graphics workstations. However, it has a list price of only \$2,500 (and is available in many universities for about \$1,000).

### 3. Why workstations?

**3.1. More for the money.** If, over the past twenty years, the price of computer hardware has dropped by a factor of 1,000, then we should be able to get 1,000 times more computing for our dollar. There are two basic alternatives for taking advantage of the price decline: make the central mainframe 1,000 times as big or make 1,000 copies of the central mainframe. Each has advantages for different purposes.

For example, a single *super* computer can handle very large problems—in ways that 1,000 smaller ones cannot.

On the other hand, the change in price is not as simple as a uniform 1,000 fold drop in the cost of all aspects of computers. Nonlinearities in price/performance give smaller computers great advantages for many uses.

The primary reason for the drop in the price of computer hardware has been the development of cheap large-scale integrated circuits, which, in particular, has led to cheaper processors and memory. A single-user workstation is less complex and can take greater advantage of VLSI technology than a multi-user machine, resulting in more instruction cycles per dollar. Furthermore, the drop in memory prices has made the combination of workstation and high quality, bitmap graphics displays affordable.

Single user machines also tend to have less software overhead; the effort normally committed to managing the complexity of a multi-user system can instead be invested in enriching the programming environment for the single user.

**3.2. Graphical interaction.** A basic premise of research in computing environments is that using a computer should be a natural activity, in the same sense that driving a car is natural.

An important advantage of workstations is the possibility of a natural, graphical language for control of the computer. Bitmap displays and graphical input devices provide a “high bandwidth channel” (rich and fast information transmission both ways) for communication between person and machine.

Graphical interaction permits the design of sophisticated programming environments that are both easier for the novice and more powerful for the expert. We will discuss programming environments in more detail in a future part of the paper.

Acceptable graphical interaction is difficult to achieve on anything but a single user graphics workstation because it requires

- computing power on demand,
- high speed data transfer between cpu and display

to have guaranteed fast response. On a multi-user machine, the response time will vary as the operating system varies the size of the time-slice given to each user. Multi-user machines typically communicate with graphics terminals at rates of about 10,000 bits per second; required communication speeds may be one or two orders of magnitude higher.

**3.3. Local control.** We expect to see a graphics workstation on the desk of every statistician who wants one—in five to ten years. At present, the type of workstation we are considering is too expensive (\$15,000 to \$150,000) for that, but it is reasonable to expect a Statistics department to be able to buy one or more. There are a number of advantages that come from having a computer that belongs to a small group of people with similar interests and serves only one person at a time.

A single user computer has its total computing power available on demand. Among other benefits, this permits guaranteed response time for interaction. On a large mainframe the central processor's attention is divided among several users in a round-robin fashion so that in any given second no one user receives more than a fraction of a second of processing time, and that fraction depends on the numbers of other users currently active on the system.

The owner(s) of a workstation decides how to use its resources, not a large bureaucracy. This makes it possible to provide essentially free computer time for computer intensive methods; once the machine is paid for the owner(s) need not worry about incremental charges for cpu time.

Since a workstation is dedicated to a single user, it can be tailored to a specific application, e.g. data analysis, rather than maintaining compromises unavoidable in a general purpose computing environment.

**3.4. New statistical methods.** We have mentioned above two measures of performance of computing environments: execution turn-around time and program modification time. The quantitative change in program modification time (from hours or days in batch environments to seconds in sophisticated, integrated programming environments) has great implications which we will explore more fully in future parts of this paper.

In this section, we give some examples of how quantitative change in execution turn-around time leads to qualitatively new statistical methods.

Consider the length of time it takes to draw a scatterplot, with a line fit to the data by least squares. With pencil-and-paper, it might take days for moderate sized (500 observations) data sets. In a batch processing environment, it will take hours. In a timesharing environment, it will take minutes. On a graphics workstation, it need only take 1/10 of a second.

The fact that a new plot can be drawn in fractions of a second makes possible two new methods for data analysis that have been explored with prototype data analysis systems:

- Prim-9 (Fisher, Friedman, and Tukey (1974)).
- Prim-H (Donoho, Huber, and Thoma (1981), Donoho, Huber, Ramos, and Thoma (1982)).
- Orion I (Friedman and Stuetzle (1982c), Friedman, McDonald, and Stuetzle (1982), McDonald (1982a), (1982b), (1982c)).

**3.4.1. Three-dimensional scatterplots.** There are a variety of techniques for drawing three-dimensional scatterplots (Nicholson and Littlefield (1982), Littlefield and Nicholson (1982)). The Prim and Orion systems use the human ability to perceive shape from motion. Essentially, a two-dimensional orthogonal projection of the three-dimensional point cloud is displayed on the graphics screen. Rotating the point cloud in real-time allows the user to view from any direction and gives a convincing and accurate perception of the shape of a three-dimensional scatterplot from the apparent parallax in the motion of the points. This is demonstrated in the Prim-9 and Orion films (Prim-9 (1974), McDonald (1982a), (1982b), (1982c)).

Real-time motion is a demanding task. As is in the cinema, the apparently moving picture is actually composed of a sequence of static frames. Conventional movies are shown at 24 frames per second; television at 30 frames per second. Three-dimensional scatterplots can be successfully displayed somewhat more slowly—down to perhaps 3 frames per second. However, shape is perceived with more comfort at rates closer to 24 or 30 frames per second.

The rate at which motion graphics can be displayed is determined by:

- the time it takes to compute a new frame,
- the time it takes to erase the old frame and draw the new one.

To display three-dimensional scatterplots—or other type of motion graphics—at 30 frames per second, all the computations associated with each frame must be completed in roughly 1/60 of a second, to allow time for drawing and erasing the frames. Rotation about an arbitrary axis with perspective (Foley and Van Dam (1982), Newman and Sproull (1979)) requires at least nine multiplications, nine additions, and two divisions—20 arithmetic operations per point. Therefore, for smooth rotation of a scatterplot with 1,000 points, the computer may need to do 1,200,000 arithmetic operations per second. Orthogonal rotation about either the vertical or horizontal axis takes 3 multiplies and 2 adds per point; at a minimal rate of 5 frames per second, 1,000 points demands about 50,000 arithmetic operations per second.

Moving scatterplots are usually displayed by drawing and erasing a small symbol for each point. Such symbols may consist of from 1 to 100 pixels (picture elements). For five pixel symbols, erasing and re-drawing a frame with 1,000 points in 1/60 second requires a graphics device that can do 600,000 pixel writes per second. Changing a pixel typically means sending 32 bits of address and 8 bits of data—a total of 5 bytes—from the computer to the graphics device. Therefore smooth rotation may require the computer to be able to communicate with the graphics device at 3 megabytes per second.

This type of real-time motion is impractical on a time-shared computer, since one cannot rely on a sufficiently large timeslice to do the necessary computing. In contrast, a workstation is dedicated to a single user, so the timing constraints are more easily satisfied. Also, real-time motion is inhibited in time-shared systems by slow data transfer between a central processor and a remote graphics terminal.

**3.4.2. Interactive model fitting.** Suppose we want to fit a model with several parameters to a set of data. Also suppose we can draw a picture that shows how well the model fits for any given values of the parameters. Then it may be useful to be able to vary some or all of the parameters interactively, with a graphical input device (such as mouse, touchpad, trackerball, joystick).

A simple example is linear regression with power transformations. That is, we approximate observed data  $\{x_i, y_i\}$ , with the model:

$$y = a + b \cdot x^{\theta}.$$



For fixed values of  $\theta$ ,  $\hat{a}(\theta)$  and  $\hat{b}(\theta)$  are fit by least squares. On a graphics workstation, we can use a dial, or some equivalent *graphical input device* to control the value of  $\theta$ . We would display a scatterplot of  $y_i$  versus  $z_i = x_i^\theta$  with the least squares line:  $y = \hat{a}(\theta) + \hat{b}(\theta) \cdot z$ . The picture must change smoothly as we move the dial that controls  $\theta$ . To do this the graphics system must be able to compute the transformation,  $z_i = x_i^\theta$ , the least squares fit, and erase and redraw the picture about 30 times a second.

A still more demanding example is Interactive Projection Pursuit Regression, which is described by Friedman and Stuetzle (1981b), (1982a), (1982b) and McDonald (1982a) and demonstrated in the film by McDonald (1982c).

**4. Stay tuned till next week.** This has been an introduction to a series of papers on research in computing environments for data analysis. It is intended to provide an overview and motivation for what follows.

There are two important aspects to computing environments. The more fundamental and limiting of the two, *Hardware*, will be reviewed in detail in part II (this issue, pp. 1013–1021). The other aspect, *Programming Environments*, is perhaps of more immediate interest as an area of research for statisticians. It will be the basis for parts III through V of this paper:

- Issues in the Design of Programming Environments for Data Analysis.
- Conventional Programming Environments: Unix.
- Integrated Programming Environments: Lisp and Flavors.

#### REFERENCES

- D. R. BARSTOW, H. E. SHROBE AND E. SANDEWALL, eds. (1984), *Interactive Programming Environments*, McGraw-Hill, New York, 1984.
- J. C. BEATTY (1983), *Raster graphics and color*, American Statistician, 37, pp. 60–75.
- R. A. BECKER AND J. M. CHAMBERS (1984a), *Design of the S system for data analysis*, Comm. ACM, 27, pp. 486–495.
- (1984b), *S: An Interactive Environment for Data Analysis and Graphics*, Wadsworth, Belmont, CA.
- H. M. DEITEL (1983), *An Introduction to Operating Systems*, Addison-Wesley, Reading, MA.
- P. DEUTSCH AND E. TAFT, eds. (1980), *Requirements for an experimental programming environment*, Xerox PARC Report CSL-80-10, Palo Alto, CA.
- D. DONOHO, P. J. HUBER AND H. THOMA (1981), *The use of kinematic displays to represent high dimensional data*, Computer Science and Statistics: Proc. 13th Symposium on the Interface, W. F. Eddy, ed., Springer-Verlag, New York.
- D. DONOHO, P. J. HUBER, E. RAMOS AND H. THOMA (1982), *Kinematic display of multivariate data*, Proc. Third Annual Conference and Exposition of the National Computer Graphics Association, Inc., Vol. I.
- M. A. FISHERKELLER, J. H. FRIEDMAN AND J. W. TUKEY (1974), *Prim-9, An interactive multidimensional data display and analysis system*, Proc. Pacific 75, ACM Regional Conference.
- J. D. FOLEY AND A. VAN DAM (1982), *Fundamentals of Interactive Computer Graphics*, Addison-Wesley, Reading, MA.
- J. H. FRIEDMAN, J. A. McDONALD AND W. STUETZLE (1982), *An introduction to real time graphical techniques for analyzing multivariate data*, Proc. Third Annual Conference and Exposition of the National Computer Graphics Association, Inc., Vol. I.
- J. H. FRIEDMAN AND W. STUETZLE (1981b), *Projection pursuit regression*, J. Amer. Statist. Assoc., 76, pp. 817–823.
- (1982a), *Smoothing of scatterplots*. Dept. Statistics Tech. Rept. Orion 2, Stanford Univ., Stanford, CA.
- (1982b), *Projection pursuit methods for data analysis*, in Modern Data Analysis, Launer, R. L. and Siegel, A. F., eds., Academic Press, New York, 1982.
- (1982c), *Hardware for kinematic statistical graphics*, Computer Science and Statistics: Proc. of the 15th Symposium on the Interface.
- A. GOLDBERG (1983), *The influence of an object-oriented language on the programming environment*, reprinted in D. R. Barstow, H. E. Shrobe and E. Sandewall, eds. (1984).
- (1984), *Smalltalk-80: The Interactive Programming Environment*, Addison-Wesley, Reading, MA.

- A. GOLDBERG AND D. ROBSON (1983), *Smalltalk-80, The Language and Its Implementation*, Addison-Wesley, Reading, MA.
- P. E. GREEN, ed. (1982), *Computer Network Architectures and Protocols*, Plenum Press, New York.
- B. W. KERNIGHAN AND J. R. MASHEY (1981), *The UNIX programming environment*, Computer, 14, pp. 25-34.
- B. W. KERNIGHAN AND D. M. RITCHIE (1978), *The C Programming Language*, Prentice-Hall, Englewood Cliffs, NJ.
- G. KRASNER, ed. (1983), *Smalltalk-80, Bits of History, Words of Advice*, Addison-Wesley, Reading, MA.
- R. J. LITTLEFIELD AND W. L. NICHOLSON (1982), *Use of color and motion for the display of higher dimensional data*, Proc. 1982 DOE Statistical Symposium.
- J. A. McDONALD (1982a), *Interactive graphics for data analysis*, Ph.D. thesis, Dept. Statistics, Stanford Univ., available as Dept. of Statistics Tech. Rept. Orion 11, 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.
- W. M. NEWMAN AND R. F. SPROULL (1979), *Principles of Interactive Computer Graphics*, 2nd ed., McGraw-Hill, New York.
- W. L. NICHOLSON AND R. J. LITTLEFIELD (1982), *The use of color and motion to display higher dimensional data*, Proc. Third Annual Conference and Exposition of the National Computer Graphics Association, Inc., Vol. I.
- PRIM-9 (1974), *Prim-9*, a 30 minute, 16 mm sound film. It is available for loan from: Jerome H. Friedman, Computation Research Group, Bin # 88, SLAC, P.O. Box 4349, Stanford, CA 94305.
- B. A. SHEIL (1983), *Power tools for programmers*, reprinted in D. R. Barstow, H. E. Shrobe and E. Sandewall, eds. (1984).
- A. S. TANNENBAUM (1981a), *Computer Networks*, Prentice-Hall, Englewood Cliffs, NJ.
- (1981b), *Network protocols*, ACM Comput. Surveys, 13, pp. 453-489.
- W. TEITLTMAN AND L. MASINTER (1981), *The Interlisp programming environment*, reprinted in D. R. Barstow, H. E. Shrobe and E. Sandewall, eds. (1984).
- J. W. TUKEY (1977), *Exploratory Data Analysis*, Addison-Wesley, Reading, MA.
- UNIX (1978), special issue on Unix, Bell System Tech. J., 57.

## COMPUTING ENVIRONMENTS FOR DATA ANALYSIS II. HARDWARE\*

JOHN ALAN McDONALD† AND JAN PEDERSEN†

**Abstract.** This is the second in a series of papers on aspects of modern computing environments that are relevant to statistical data analysis. We argue that a network of graphics workstations is far superior to conventional batch or time-sharing computers as an environment for interactive data analysis. The second paper in the series discusses the hardware of graphics workstations in detail.

**Key words.** data analysis, interactive graphics, workstations, programming environments

**1. Introduction.** The purpose of this paper is to list important features to consider when evaluating and comparing graphics workstations to be used for data analysis research.

The workstations described here cannot be used for data analysis “off the shelf”; they are at present useful primarily for research. We expect a minimum of 5–10 years of development before one will be able to buy both a workstation and statistical software to use on it.

The computer graphics market changes rapidly; new, better, cheaper machines are announced every few months. In the course of this paper, we will mention features of six specific machines—chosen somewhat arbitrarily out of the hundreds that are available—as concrete examples. Four of these, Apollo, Chromatics, Iris, and Sun, are representatives of a large class of graphics workstations based on versions of the Motorola 68000 microprocessor. This type of workstation typically costs \$40,000 to \$60,000 in configurations appropriate for statistical applications. The fifth machine, the Ridge 32, has a *reduced-instruction-set* central processor that is a bit more expensive and considerably more powerful than a 68000. The architecture of the sixth machine, the Symbolics 3600, is specially designed for efficient execution of programs written in Lisp; it is both more powerful and more expensive (both by a factor of 2–3) than the 68000 based machines.

We emphasize that this is by no means an exhaustive list; these are machines which have been, are being, or will be used for research in data analysis systems. Apollo is used by the Statistics Department at Harvard University; Chromatics by the Statistics Departments at Stanford and Berkeley University; the Iris will be used by the Statistics Department at Stanford; the Ridge 32 will be used by Statistics Departments at Stanford, Berkeley, and the University of Washington; Sun hardware was the basis of the Orion I workstation at Stanford; the Symbolics 3600 is used by the Statistics Departments at Stanford and the University of Washington, and by groups at Bell Laboratories.

A reference for microcomputer architecture is Kraft and Toy (1979). Foley and Van Dam (1982) discuss the architecture of microcomputer graphics systems.

### 2. The central processor.

**2.1. Single-chip microprocessors.** One or two dozen companies make graphics workstations based on the Motorola 68000 microprocessor (Motorola (1982)).

---

\* Received by the editors June 5, 1984. This research was supported by the National Science Foundation under a Mathematical Sciences Postdoctoral Research Fellowship, the Office of Naval Research under contract N00014-83-K-0472 and grant N00014-83-G-0121, and the U.S. Army Research Office under contract DAAG29-82-K-0056.

† Department of Statistics and Computation Research Group, Stanford Linear Accelerator Center, Stanford University, Stanford, California 94305.

Examples are Apollo, Chromatics, Iris, and Sun. In the future, other 32 bit microprocessors will be important competitors (for example, the National Semiconductor NS16032 and NS32032, the Hewlett-Packard HP-9000, Bell Laboratories Bellmac 32, Zilog Z80,000), but there are few available workstations based on these processors (as of April 1984).

The 68000 is sometimes described as a 16/32 bit machine; internally it is a 32-bit machine; it communicates with the outside world with 16-bit data words and 23-bit address words. Motorola has recently released a version called the 68010, which supports virtual memory (the plain 68000 does not). Iris and Sun, and possibly also the Apollo will use the 68010; the Chromatics uses the 68000. A fully 32-bit version of the 68000 (the 68020) is promised for the future.

**2.2. Bit-sliced processors.** *Bit-sliced processors* are an alternative to one-chip microprocessors like the 68000. Bit-sliced processors are made by combining chips which perform standard operations on a *slice* of a machine word. Slices are commonly four bits wide; so, for example, eight 4-bit adders would be combined to make a 32-bit adder, which would be combined with other 32-bit parts to make a complete processor. Bit-sliced processors usually execute a *micro-coded* instruction set. That is, each assembly language instruction calls a small, fast program written in a simpler *microcode* language, which is executed directly by the bit-slice hardware.

Bit-slice components are often used for special purpose processors; the flexible hardware and the micro-coded instruction set make it possible to design and construct a new processor without the capital investment needed to produce a new microprocessor. Bit-slice processors are usually faster (and more expensive) than general-purpose, single-chip microprocessors because they are designed to be particularly efficient at special tasks and also because they typically use faster transistor technology.

**2.2.1. Reduced instruction set machines.** The *Ridge 32* is a graphics workstation with a bit-sliced central processor that is roughly equivalent to a VAX 11/780, or 2-3 68000's, for a price that is no more than that of typical 68000 based workstations. The low price and high power of the Ridge is due, in part, to the fact that it is a *reduced instruction set computer (RISC)*. This means that it has a relatively simple instruction set, which is executed directly by the hardware, rather than going through *micro-code translation* first.

**2.2.2. Lisp machines.** The *Symbolics 3600* (Symbolics (1983), Bawden et al. (1979)) is another graphics workstation with a bit-sliced central processor. The 3600 is a LISP machine, that is, its architecture is designed for fast execution of programs written in LISP. This has, obviously, implications for the 3600's software properties, which will be discussed below. The 3600 is equivalent to 1 or 2 VAX 11/780's, or 2-5 68000's and it is about 3 times the price of a typical 68000-based workstation.

Other Lisp machines are:

Lisp Machine, Inc.'s *Lambda* and Texas Instruments' *Explorer* which are very similar to the 3600.

Xerox has its *D- or 1100-series*, which includes the *Dolphin*, the *Dandelion*, and the *Dorado* (Dorado (1981)) machines.

**2.3. Clock speed and wait states.** The *clock speed* of a 68000 is its basic cycle time and, therefore, determines how fast it executes instructions. The clock speed is a property of the computer in which the 68000 chip is placed. It is also a property of the chip, in the sense that not all chips will run successfully at high speeds. Typical speeds range from 4 to 20 megahertz. A 16 mhz machine is, not surprisingly, about 4

times as fast as a 4 mhz one. Every 16 mhz machine is not exactly 4 times as fast as every 4 mhz one because the speed of memory and the *bus* that connects the 68000 to the memory can make a considerable difference to the effective speed of the processor. Bus and memory speed are reflected in the *number of wait states* in a given machine, which are processor cycles lost while the processor waits for the bus. As a rough guide to the power of 68000 machines, the Sun workstation contains a 10 mhz 68010 with no wait states and is roughly comparable to a VAX 11/750 (for nonfloating computations).

### 3. Auxiliary processors and memory.

**3.1. Bus architecture.** The *bus* is used by the central processor to communicate with memory and other I/O devices, in particular, the graphics display. A common standard bus used in 68000 based machines is the MULTIBUS (Boberg (1980)), used in Sun and Iris. It is an advantage to have a machine with a standard bus architecture, because it is then easier to add peripherals, such as floating point boards, array processors, printers, or additional input devices.

Some machines (Sun and Iris, for example) are designed so that the RAM is either on the same board as the 68000 or is connected with a separate, private bus. This allows the 68000 to access memory without using the main (MULTIBUS) bus, avoiding delays that result when the bus must be shared with auxiliary processors, such as a floating point board or array processor.

Another reason that some workstations add a second, private bus is that the MULTIBUS is not fast enough for a 68000 run at high clock speeds. In the future, workstations will be provided with faster 32-bit buses, to keep up with 32-bit processors run at high clock speeds.

Many workstations increase their computational power by adding auxiliary processors which may be more efficient at specialized tasks and execute in parallel with the central processor.

**3.2. Scalar floating point.** The 68000, like most other microprocessors, does not include floating point operations in its instruction set. A machine with only a 68000 therefore executes floating point operations as small programs, slowly. Efficient software floating point operations take 50–100 microseconds on a 10 mhz 68000 with no wait states. The same operations require closer to one microsecond on a VAX. One solution is to add an auxiliary floating point processor to a 68000-based machine. A floating point board is included in the Apollo and is available (from Sky Computers, Inc., for example) for the Chromatics, Iris, and Sun.

The Ridge 32 and the 3600 bit-sliced central processors do not have floating point operations as primitives. However, software floating point is efficient enough to give about half the speed of a VAX 11/750 for the Symbolics and speed about equal to the 11/750 on the Ridge. Both Ridge and Symbolics plan to add hardware floating point processors in the future.

**3.3. Vector floating point.** Many graphical and statistical computations can be efficiently done in parallel. So it is useful to have a workstation with an *array processor*. The Iris workstation includes an array processor based on a chip called the *Geometry Engine* (Clark (1982)), which is specially designed for graphical computations and should be able to perform about 10 million floating operations per second—the equivalent of 10–20 VAX 11/780's. General purpose array processors (from Sky Computers, Inc., for example) can be added to Chromatics, Sun, and Iris.

The other machines have nonstandard bus architectures, which makes it difficult to add array processors that can communicate quickly with the cpu.

**3.4. I/O processors.** Some workstations contain special processors for input and output, which free the central processor to do other work. I/O processors may, for example, control the keyboard, handle data transfers between disk and memory (DMA's), buffer communications with a network, or speed communication with the graphics device (see below).

**3.5. Random access memory.** Graphics workstations are usually provided with from  $\frac{1}{2}$  to 2 megabytes of RAM (*random access memory*). The speed of the memory places a constraint on the effective speed of computation. Slow memory will require the processor to have some number of wait states (lost cycles, mentioned above).

**3.6. Memory caches.** Some workstations gain additional processing speed by including *high speed caches* (Clark, Lampson and Pier (1981), Smith (1983)), *instruction pre-fetch caches* and/or *data stack caches*. A cache is a limited amount of high speed memory; by keeping the next instructions to be executed and/or currently relevant data in a cache a computer can increase its effective rate of computation. The Apollo has an instruction cache; the 3600 has both instruction and data caches.

**3.7. Virtual memory.** *Virtual memory* allows a computer to run programs and manipulate data sets too large to fit in its physical memory. It is a property of the computer's operating system as well as its hardware. Some processors cannot run virtual memory operating systems (because they cannot recover from page faults). The original 68000 cannot run a virtual memory operating system. The more recent 68010 can. The Chromatics uses the 68000 and does not have virtual memory; Iris and Sun use the 68010. Older versions of Apollo use two 68000's to support virtual memory in a somewhat clumsy way; future Apollos should use the 68010. The Ridge and 3600 cpus support virtual memory.

**3.8. Disk storage.** Graphics workstations usually include a disk for long term data storage; most manufacturers offer a range of disk sizes, typically from 10 to 500 megabytes. Workstations use *Winchester* disk technology, which permits disks to be physically much smaller and less expensive than traditional disk technology. 50 megabytes is adequate for statistical applications; 500 megabytes is not an unreasonable size. A 500 megabyte disk will cost two or three times a 50 megabyte one—from \$15,000 to \$30,000.

#### 4. Graphics device.

**4.1. Bitmap graphics.** All of the graphics workstations that we are considering here use a *bitmap graphics display* which is also called a *raster graphics display* or a *frame buffer*. For a fuller discussion of bitmap graphics devices see: a survey paper aimed at statisticians by Beatty (1983) and texts by Foley and Van Dam (1982) and Newman and Sproull (1979).

**4.1.1. Resolution.** In a bitmap graphics device, the picture is made up of an array (a raster) of dots, called *pixels*. The *resolution* of the device is the number of pixels on the screen. Common resolutions are either approximately  $512 \times 512$  or  $1,024 \times 1,024$ . For many kinds of statistical graphics  $512 \times 512$  is enough. A  $1,024 \times 1,024$  bitmap is needed for realistic images of solid objects, scatterplots in which small symbols (glyphs) are used to code additional variables, or if the screen is used to display several plots at once.

**4.1.2. Depth.** The color of each pixel is determined by the part of the *refresh memory* associated with it. The number of bits of refresh memory associated with each

pixel is the *depth* of the bitmap. This is also referred to as the *number of bitplanes* in the graphics device. Typical depths are 1, 4, 8, 16, 24, or 32, with 1 and 8 being the most common.

The depth of the bitmap determines the number of different colors that can be displayed simultaneously. A device with a single bitplane can display only two colors (e.g. black and white) depending on whether the single bit of refresh memory corresponding to a pixel is 0 or 1. A device with eight planes can display 256 colors at once.

Obviously, the more planes a bitmap has the better. Adding planes increases the price of a system; for a  $1,024 \times 1,024$  bitmap, each additional eight planes requires an additional megabyte of refresh memory. The refresh memory must be fast enough to permit its entire contents to be read 30 times per second (60 times a second for noninterlaced monitors). It is therefore more expensive than ordinary RAM; refresh memory prices are \$7,000–\$20,000 per megabyte as compared to \$2,000–\$7,000 per megabyte of RAM.

For most statistical applications, eight planes is enough. Bitmaps with more depth are useful for drawing realistic solid objects and for image processing applications. Extra bitplanes are also useful for *double buffering*, drawing multiple independent images, and *color map animation*.

**4.2. Refresh cycle.** A bitmap graphics device maintains a picture on the screen by going through the *refresh cycle* fast enough to avoid flicker (typically either 30 or 60 times per second).

In a bitmap device, the electron beam scans over the screen in a regular fashion, modulating the intensity of the beam as it goes to determine the brightness and color of each pixel. A raster line is usually drawn as the beam scans from left to right. With the intensity set to zero, the beam then moves back quickly, from right to left, to the start of the next raster line (the *horizontal blanking or retrace* part of the refresh cycle). When the beam reaches the bottom of the screen, it quickly moves back to the top, with the intensity at zero (*vertical retrace*). Some devices scan every other raster line in one pass from top to bottom, getting the missed lines in a second pass (see section on *interlacing* below).

When the electron beam strikes a point on the screen, the phosphors emit light with a brightness that depends on the intensity of the beam. In color systems, there are small dots of red, green, and blue phosphors, whose emissions mix to produce pixels that are perceived as having arbitrary colors. The brightness of the phosphors decays rapidly after the electron beam leaves. To produce a stable picture, the electron beam must return before the decay in brightness is noticeable.

The picture on the screen is determined by the contents of the refresh memory. With a  $1,024 \times 1,024 \times 8$  bitmap and a 30 hz monitor, the refresh memory must be read at about 32 megabytes per second (to allow time for horizontal and vertical retrace). The 8 bits per pixel is decoded into, typically, 8 bits each of red, green, and blue intensity by a color map (next section). *Digital-to-analog converters* translate the color intensities to voltages that modulate the intensity of the electron beam.

**4.3. Color maps.** *Color maps* (also called *color look up tables*) are used to make bitmap displays more flexible.

A primitive bitmap display with, say, eight planes might assign a fixed color to each of the possible pixel values (0–255). However, it is often convenient if the color associated with each pixel value can be changed. This is done using a color map. Typical color maps are *8-bits-in 24-bits-out*; they can be thought of as three arrays, one each of red, green, and blue; each array has 256 entries, indexed from 0 to 255;

the pixel value is used as an index into these arrays to determine the relative intensities (from 0–255) of red, green, and blue.

**4.3.1. True color.** *True color* is an alternative to a color map. True color may be used, for example, on a system with 24 bitplanes. A 24-bit-in 24-bit-out look up table is impractical, because it would require  $3 * 2^{24}$  bytes of very high speed memory. Instead, the 24 bit (3 byte) pixel values are used to hold 8 bits of intensity for red, green, and blue. True color is available for the Iris and the 3600.

**4.3.2. Double buffering.** Graphics systems often use extra (more than 8) bitplanes for *double buffering*. A 16 plane frame buffer can hold two independent 8 plane pictures; in some machines (Chromatics), it is possible to quickly change which 8 planes are displayed on the screen. Double buffering is useful in motion graphics; a new picture is drawn into the 8 planes that are invisible, the buffers are swapped so that the new picture is visible and the old picture is invisible, and the invisible 8 planes are erased. This eliminates distracting *beating* or *aliasing* effects that arise when the drawing–erasing cycle is visible and interferes with the refresh cycle.

**4.3.3. Synchronization of refresh and color map changes.** A slightly subtle consideration is whether changes to the color map are synchronized with the refresh cycle. The color map should only be changed when the screen is in the blanked part of the refresh cycle. Otherwise there will be contention between the refresh and the cpu for access to the color map, which can cause disturbing effects on the display.

**4.4. Graphics speed.** Many statistical applications require motion graphics, so the time required to modify the bitmap—the time required to erase an old picture and draw a new one—is important. The time required to change a picture should simply be the time required to change a single pixel times the number of pixels that have to be changed. However, the time to change a pixel can vary greatly depending on how it is changed. Common modes for writing to a bitmap are: single pixel change, vector drawing, rectangular area fill, polygonal area fill, and an operation on rectangular blocks of pixels called *RasterOp* or *BitBlit* (Bechtolsheim and Baskett, 1980). Typical speeds are 0.1 to 2 million pixels per second in single pixel write, 0.1 to 16 million pixels per second in vector drawing, and up to 200 hundred million pixels per second in area fill (a special area fill mode on the 3600). For comparison, to redraw an entire screen in real-time requires drawing 1 million pixels 30 times a second, or 30 million pixels per second. To draw a rotating three-dimensional scatterplot containing 1000 points, in which each point is represented by a symbol composed of, say, 5 vectors 10 pixels long, would require a vector drawing speed of at least 3 million pixels per second.

**4.4.1. Graphics processors.** The wide range in drawing speeds is in part due to the fact that some devices have special graphics processors to speed up some drawing tasks.

For example, to draw a vector, it must be *rasterized*, that is, some processor must decide which pixels have to be written to connect the vector's two endpoints. On the Sun, for example, this computation is done by the 68000, which makes vector drawing slow. The Chromatics, in contrast, contains a special vector processor, which rasterizes the vector and modifies the necessary pixels, freeing the 68000 for other computation.

The Iris is an extreme example of a machine with auxiliary graphics processing. The Iris includes the VLSI *Geometry System* (Clark (1982)) a powerful, general purpose graphics processor which will generate linear vectors, quadratic and cubic curves, all conic sections, rotate, scale, clip, do the perspective computation, etc.



**4.4.2. Communication between central processor and bitmap.** The speed of picture drawing is limited by the speed and intimacy of communication between the cpu and the bitmap. On some systems (Chromatics and 3600, for example) the refresh memory is on the cpu bus and is in the cpu's address space; in other words, the pixels in the bitmap can be accessed by the cpu as though they were bytes of ordinary RAM. This makes communication between the cpu and the bitmap fast and flexible. In a less desirable alternative, the cpu talks to the bitmap by sending graphics commands to a special interface (this was the case on Orion I), which is relatively slow and awkward.

**4.5. Monitor.** A 19-inch color monitor is standard on graphics workstations. A monitor may run 30 hz *interlaced*, which means that all the even scan lines are drawn in 1/60th of a second and the odd scan lines are drawn in the next 1/60th of a second, so that the entire screen is refreshed 30 times a second. A better but more expensive alternative is a 60 hz *noninterlaced* monitor, which draws all the scan lines on the screen, in one pass, 60 times a second.

Another consideration in the choice of a monitor is the *persistence* of the phosphors. Each point on the screen is hit by the electron beam 30 or 60 times a second. The brightness of a point decays exponentially after the electron beam leaves it. The phosphors must be persistent enough so that the screen does not flicker noticeably in the 1/30 of a second between refreshes. On the other hand, if the phosphors are too persistent, moving pictures (rotating scatterplots) will produce distracting streaks.

60 hz noninterlaced monitors are less likely to have objectionable *flicker*, making shorter persistence phosphors practical. With 30 hz interlaced monitors, flicker may not be a problem, depending on room lighting and a variety of other factors. In some machines (Chromatics), the refresh is synchronized with the 60 hz cycle in the wall outlet's alternating current. Thus the monitor is synchronized with the 60 hz oscillation in the brightness of fluorescent room lighting, which helps minimize flicker.

**4.6. Miscellaneous features.** Graphics devices come with a variety of additional features which are not critical to statistical applications, but are often useful nonetheless. Among these are:

- Zoom—magnify the bitmap so that a fraction ( $\frac{1}{4}$ ,  $\frac{1}{9}$ ,  $\frac{1}{16}$  etc.) of it fills the screen.
- Pan—translate the bitmap after zooming, so that a different portion is visible.
- Blink planes—are used to make pixels blink; this is often done by alternately masking and not masking the blink bit, which causes the pixel value to alternate between two address in the color map.
- Overlay planes—are used to write text nondestructively over the bitmap.
- Hardware cursor—draws and moves a small pointing symbol or *cursor* nondestructively over the bitmap.

**5. Input devices.** In the preceding sections we have discussed the display, the workstation's output device. One advantage of graphical workstations is use of *graphical input devices*, which can make using a computer a natural activity, in the same way that driving a car is natural.

Foley and Van Dam (1982) define five *logical input devices*: the *locator*, which indicates position and/or orientation, the *pick*, which selects an object displayed on the screen, the *valuator*, which chooses a real number, the *button(s)*, which select from a finite set of alternatives, and the *keyboard*, which inputs a character string.

Graphics workstations are usually provided with keyboards and some number of other graphical input devices, such as: joystick (Chromatics), mouse (Iris, Sun, 3600), and touchpad (Apollo). Each of these physical input devices, including the keyboard,

can be used to implement any of the five logical input devices, but the implementation will be much more natural in some cases than others.

**6. Networking.** Because graphics workstations are single-user, stand-alone computers, it is important for them to be able to communicate quickly and easily with other computers—other workstations or mainframes. Computers communicate better if they are linked in a *network* (Green (1982), Tannenbaum (1981a), (1981b)). A network can be used simply to transfer files between machines, or it can be used in a more sophisticated fashion to distribute computing tasks among machines in the network.

A common standard is Xerox's Ethernet, which is supported by Iris, Sun, and the 3600. Apollo supports its own, idiosyncratic network. The Chromatics has no networking support.

An example of more sophisticated use of a network is the Sun's option for diskless workstations. These diskless stations use the Ethernet to simulate a virtual disk which is physically part of a large disk maintained by a special *fileserver* station. A network of diskless stations provides more computing power for less initial expense and less maintenance, at the cost of somewhat slower disk access and potential bottlenecks when many stations need to use the central disk at the same time.

**7. Benchmarks.** Many factors, software as well as hardware, determine the effective speed of computation. To get a more valid comparison of machines, it is useful to run one or several *benchmark* programs. A simple benchmark used to test floating point speed is to sum up the harmonic series (suggested by Peter Huber).

To sum 100,000 terms of the harmonic series (in single precision) takes approximately:

- 1 second on a Ridge 32 (with software floating point).
- 1 second on a quiet (single user) VAX 11/750 (with floating point accelerator).
- 2 seconds on a Symbolics 3600 (without floating point accelerator and without high speed instruction pre-fetch cache). On a Symbolics 3600 with floating point accelerator and high speed cache it would presumably take much less than one second.
- 10 seconds on a Apollo (with hardware floating point and high speed cache).
- 35 seconds on a Sun (software floating point).
- 80 seconds on a Chromatics (software floating point).

**8. Sources.** The best way to survey the computer graphics market is to attend annual meetings, such as the COMDEX convention, the NCC meeting, the National Computer Graphics Association (NGCA) meeting, usually in June, and the annual ACM SIGGRAPH meeting, usually in July.

The next best way is to review journals that carry advertisements and announce new computing and graphics products, such as *Electronics* magazine, *IEEE Computer*, *IEEE Computer Graphics and Applications*, *Computer Graphics World*, etc.

The manufacturers of the workstations mentioned in this paper are:

Symbolics, Inc. (3600), 845 Page Mill Road, Palo Alto, California 94304; (415) 494-8081.

Chromatics, Inc. (CGC-7900), 2558 Mountain Industrial Boulevard, Tucker, Georgia 30084; (404) 493-7000.

Silicon Graphics, Inc. (Iris), 630 Clyde Court, Mountain View, California 94043; (415) 960-1980.

Sun Microsystems, Inc., 2550 Garcia Ave., Mountain View, California 94043;  
(415) 960-1330.

Ridge Computers (Ridge 32), 586 Weddell Drive, Sunnyvale, California 94089;  
(408) 745-0400.

## REFERENCES

- A. BAWDEN, R. GREENBLATT, J. HOLLOWAY, T. KNIGHT, D. MOON AND D. WEINREB (1979), *The LISP machine*, in *Artificial Intelligence: An MIT Perspective*, vol. II.
- P. H. WINSTON AND R. H. BROWN, eds., *Artificial Intelligence: An MIT Perspective*, MIT Press, Cambridge, MA.
- J. C. BEATTY (1983), *Raster graphics and color*, *American Statistician*, 37, pp. 60-75.
- R. W. BOBERG (1980), *Proposed microcomputer system 796 Bus standard*, *Computer*, 13, pp. 89-106.
- J. H. CLARK (1982), *The geometry engine: A VLSI geometry system for graphics*, Proc. 1982 SIGGRAPH Conference, published as *Computer Graphics*, 16 (3), pp. 127-133.
- DORADO (1981), *The Dorado: A high performance personal computer*, Three papers Xerox PARC Report CSL-81-1. Palo Alto, CA.
- J. D. FOLEY AND A. VAN DAM (1982), *Fundamentals of Interactive Computer Graphics*, Addison-Wesley, Reading, MA.
- P. E. GREEN, ed. (1982), *Computer Network Architectures and Protocols*, Plenum Press, New York.
- G. D. KRAFT AND W. N. TOY (1979), *Mini/Microcomputer Hardware Design*, Prentice-Hall, Englewood Cliffs, NJ.
- MOTOROLA (1982), MC68000, *16-Bit Microprocessor, User's Manual*, 3rd ed., Prentice-Hall, Englewood Cliffs, NJ.
- W. M. NEWMAN AND R. F. SPROULL (1979), *Principles of Interactive Computer Graphics*, 2nd ed., McGraw-Hill, New York.
- SYMBOLICS, INC. (1983), *3600 technical summary*, Symbolics, Inc., 5 Cambridge Center, Cambridge, MA 02142.
- A. S. TANNENBAUM (1981a), *Computer Networks*, Prentice-Hall, Englewood Cliffs, NJ., 1981.
- (1981b), *Networks protocols*, *ACM Comput. Surveys*, 13, pp. 89-106.

## ASSESSMENT OF LINEAR DEPENDENCIES IN MULTIVARIATE DATA\*

V. E. KANE†, R. C. WARD‡ AND G. J. DAVIS§

**Abstract.** A procedure to identify the linear dependency structure in multivariate data is presented. The linear dependency analysis (LDA) provides a method for assessing the number of dependencies using the eigenvalues of the sample correlation matrix. The dependency structure is then identified from the right singular vectors from a singular value decomposition of the centered and scaled data matrix. An algorithm to identify competing dependencies is given along with procedures for estimating and testing the dependency coefficients. Example data sets from regression, factor analysis, discriminant analysis, and principal component analysis are analyzed.

**Key words.** multivariate dependencies, correlations, linear dependency analysis, factor analysis, LDA

**1. Introduction.** The importance of statistical dependencies in multivariate data is easily seen by noting the widespread use of correlation measures in applied data analyses. Yet, as emphasized by Belsley, Kuh and Welsch (1980, p. 92), the standard correlation matrix examines only pairwise relationships, which can be negligible when more complex dependencies are present. Occasionally, it may be possible to use multiple or canonical correlations if pre-specified groups of variables are known. However, in the general case dependencies are thought to exist within a data set, but their number and the variables involved are unknown. When confronted with this dependency problem, the analyst may resort to factor analysis. The linear dependency analysis (LDA) discussed in the following sections is somewhat less general than factor analysis, but eliminates some of the ambiguity associated with matrix rotations and estimation problems in factor analysis.

The problem considered in LDA arises when the random observational vector  $\mathbf{X}(p \times 1)$  is partitioned into  $\mathbf{X}_1(p_1 \times 1)$  and  $\mathbf{X}_2(p_2 \times 1)$  where

$$(1.1) \quad \mathbf{X}_2 = \boldsymbol{\beta}_0 + \boldsymbol{\beta}'\mathbf{X}_1 + \boldsymbol{\epsilon}$$

with  $\boldsymbol{\beta}(p_1 \times p_2) = (\boldsymbol{\beta}_1, \dots, \boldsymbol{\beta}_{p_1})'$  a matrix of unknown coefficients and  $\boldsymbol{\epsilon}$  random error uncorrelated with  $\mathbf{X}_1$ . The  $\mathbf{X}_1$  values represent predictor variables and the  $\mathbf{X}_2$  vector estimated variables. The purpose of LDA is to: (1) determine a "meaningful" number of dependencies ( $p_2$ ), and (2) make assignments of variables to the predictor or estimated variable subsets. The first problem is the same as is encountered in standard principal component or factor analysis problems, but formulating dependencies by (1.1) produces a patterned covariance matrix which can be usefully exploited. The solution to the second problem is unique in the sense of minimizing residuals and an algorithm is given to find this solution.

When multivariate dependencies can be formulated as (1.1), several applications are apparent. (1) Dimensionality reduction. Principal component analysis is often used to reduce the number of "variables" considered in large multivariate problems. The retained principal components are sometimes difficult to interpret in terms of the measured variables. Also, there is no data reduction in terms of the measured quantities. (2) Modeling. The dependency structure is an important component in many multivariate problems since modeling variable interrelationships is important. Factor analysis focuses on unobserved underlying "factors" generating the dependency structure. The LDA model (1.1) directly addresses "observable" linear dependency relationships. (3) Redundancy. A question which arises in many large

\*Received by the editors November 21, 1983. This research was sponsored 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.

†Ford Motor Company, Box 1517A, NAAO Building, Dearborn, Michigan 48121.

‡Mathematics and Statistics Research, Oak Ridge National Laboratory, Oak Ridge, Tennessee 37830.

§Department of Mathematics, Georgia State University, Atlanta, Georgia 30303.

multivariate data sets is determination of how many variables are really necessary. Variable reduction could reduce storage or analysis costs of current data and “optimize” the collection of future data. Also, LDA provides a method to determine dependencies which cause singular covariance matrices in the extreme redundancy case. McCabe (1982) addresses the above variable selection problem considering different selection criteria than are considered here.

**2. Preliminaries.** Let  $\mathbf{X}$  have a multivariate distribution with mean  $\boldsymbol{\mu}$  and covariance matrix  $\Sigma_L$ . The partitioning of  $\mathbf{X}$  into  $(X_1 X_2)$  and assumption (1.1) give rise to a  $\Sigma_L$  of special structure. The submatrices making up  $\Sigma_L$  will provide insight into the LDA model. If the mean of  $\mathbf{X}_1$  is  $E(\mathbf{X}_1) = \boldsymbol{\mu}_1$  and the full rank covariance matrix of  $\mathbf{X}_1$  is  $\text{Var}(\mathbf{X}_1) = \Psi$ , then

$$(2.1) \quad E(\mathbf{X}) = \boldsymbol{\mu} = \begin{pmatrix} \boldsymbol{\mu}_1 \\ \boldsymbol{\beta}_0 + \boldsymbol{\beta}'\boldsymbol{\mu}_1 \end{pmatrix}, \quad \text{Var}(\mathbf{X}) = \Sigma_L = \begin{pmatrix} \Psi & \Psi\boldsymbol{\beta} \\ \boldsymbol{\beta}'\Psi & \boldsymbol{\beta}'\Psi\boldsymbol{\beta} + \Lambda \end{pmatrix}$$

where  $\boldsymbol{\epsilon}$  is assumed to have mean  $\mathbf{0}$  and covariance matrix  $\Lambda$  with  $\text{Cov}(\mathbf{X}_1, \boldsymbol{\epsilon}) = 0$ . Several identities are now apparent. First, note that

$$(2.2) \quad |\Sigma_L| = |\Psi||\Lambda|,$$

which implies that  $\Sigma_L$  is rank deficient if  $\Lambda$  is rank deficient. Also,

$$(2.3) \quad \begin{aligned} \Sigma_L &= \begin{pmatrix} \Psi & \Psi\boldsymbol{\beta} \\ \boldsymbol{\beta}'\Psi & \boldsymbol{\beta}'\Psi\boldsymbol{\beta} \end{pmatrix} + \begin{pmatrix} 0 & 0 \\ 0 & \Lambda \end{pmatrix} \\ &= \Sigma^* & + & \Lambda^* \\ &= \Gamma\Gamma' & + & \Lambda^* \end{aligned}$$

where  $\Gamma' = \Psi^{1/2} (I \boldsymbol{\beta})$ . Since the eigenvalues of the  $p \times p$  matrix  $\Gamma\Gamma'$  are the eigenvalues of the  $p_1 \times p_1$  matrix  $\Gamma'\Gamma$  and  $p_2$  zero eigenvalues,  $\text{rank}(\Sigma^*) = p_1$ . Thus, since  $\text{rank}(\Psi) = p_1$ , the additional dimensionality of  $\Sigma_L$  is due to  $\Lambda$ . Note that  $\text{rank}(\Lambda) \leq p_2$  and hence  $p_1 \leq \text{rank}(\Sigma_L) \leq p$ . For the LDA model  $\Lambda$  is associated with error variation and will be assumed diagonal with possibly some zero diagonal elements. Other patterned  $\Lambda$  error covariance matrices could be considered.

The covariance matrix from standard factor analysis (Rao (1973, p. 587)) is

$$\Sigma_{FA} = \Omega\Omega' + \Pi,$$

where  $\Omega$  is a  $p \times m$  matrix of factor loadings and  $\Pi$  is a diagonal matrix of the variances of the specific factors. Comparison of the covariance matrices from LDA and from factor analysis reveals two significant differences. If the number of factors  $m$  is  $p_1$ , then  $\Sigma_{FA}$  has  $p_1(p_1 + 1)/2$  more parameters than  $\Sigma_L$ , indicating that LDA is somewhat simpler, but less general. LDA is based on a simple partitioning of the “observed”  $\mathbf{X}$  vector as opposed to assuming the existence of “unobserved” factors, indicating that LDA is somewhat less ambiguous.

Letting the observations consist of  $n$  random samples collected as rows in the  $n \times p$  matrix  $X = (X_1 X_2)$ , the matrix version of (1.1) for the observed data becomes

$$(2.4) \quad X_2 = \mathbf{e} \boldsymbol{\beta}'_0 + X_1 \boldsymbol{\beta} + R$$

$\begin{matrix} n \times p_2 & n \times 1 & 1 \times p_2 & n \times p_1 & p_1 \times p_2 & n \times p_2 \end{matrix}$

where  $R = (\mathbf{r}_1, \dots, \mathbf{r}_n)'$  is a matrix of residuals and  $\mathbf{e} = (1, \dots, 1)'$ . The standardized data matrix is  $Z = HXD$  where  $H = I - n^{-1}\mathbf{e}\mathbf{e}'$  is the  $n \times n$  idempotent centering matrix with  $I$  the  $n \times n$  identity matrix. The  $p \times p$  scaling matrix  $D = \text{diagonal}(s_{ii}^{-1/2})$  is obtained from the

diagonal elements of the  $p \times p$  sample covariance matrix  $S = (n - 1)^{-1} X'HX$ . If  $Z = (Z_1, Z_2)$  is partitioned the same as  $X$ , then corresponding to (2.4) is the centered and scaled problem

$$(2.5) \quad Z_2 = Z_1\tilde{\beta} + \tilde{R}$$

where  $\tilde{\cdot}$  denotes the centered and scaled parameterization.

**3. Specification of the number of dependencies.** An indication of the number of dependencies  $p_2$  in the data will be obtained by considering eigenvalue relationships in the correlation matrix  $\tilde{\Sigma}_L = \Delta \Sigma_L \Delta$ . Here  $\Delta$  is a diagonal matrix with diagonal elements  $(\tilde{\Sigma}_L)_{ii}^{-1/2}$ . Partitioning  $\Delta$  into a  $p_1 \times p_1$   $\Delta_1$  and a  $p_2 \times p_2$   $\Delta_2$ , we see that

$$(3.1) \quad \tilde{\Sigma}_L = \begin{pmatrix} \Delta_1 \Psi \Delta_1 & \Delta_1 \Psi \beta \Delta_2 \\ \Delta_2 \beta' \Psi \Delta_1 & \Delta_2 (\beta' \Psi \beta + \Lambda) \Delta_2 \end{pmatrix}.$$

If we let  $\tilde{\Psi} = \Delta_1 \Psi \Delta_1$ ,  $\tilde{\beta} = \Delta_1^{-1} \beta \Delta_2$  and  $\tilde{\Lambda} = \Delta_2 \Lambda \Delta_2$ , it is evident that  $\tilde{\Sigma}_L$  has the same structure as  $\Sigma_L$  in (2.1). Since relationships among the eigenvalues of  $\tilde{\Sigma}_L$  and its submatrices will be the same as those for  $\Sigma_L$ , we drop the  $\tilde{\cdot}$  notation and assume that  $\Sigma_L$  is scaled for the remainder of the paper, except for §5 on parameter estimation and inference.

Partition  $\Sigma_L$  as in (2.3) and denote the eigenvalues of  $\Sigma_L$  by  $\sigma_1 \leq \sigma_2 \leq \dots \leq \sigma_p$ ; those of  $\Sigma^*$  by  $0, 0, \dots, 0 \leq \sigma_{p_2+1}^* \leq \dots \leq \sigma_p^*$ ; those of  $\Lambda^*$  by  $0, 0, \dots, 0 \leq \lambda_{p_1+1}^* \leq \dots \leq \lambda_p^*$ ; and those of  $\Psi$  by  $\psi_1 \leq \psi_2 \leq \dots \leq \psi_{p_1}$ . Now since  $\Sigma_L = \Sigma^* + \Lambda^*$ , it is true that (Parlett (1980, p. 192))

$$(3.2) \quad \sigma_i \geq \sigma_i^* \quad \text{for } i = 1, \dots, p,$$

and further (Parlett (1980, p. 14))

$$(3.3) \quad \sigma_{p_2} \leq \lambda_p^*.$$

For the diagonal error model, the eigenvalues  $\lambda_1 \leq \lambda_2 \leq \dots \leq \lambda_{p_2}$  of  $\Lambda$  are its diagonal elements implying  $\lambda_p^* = \lambda_{p_2}$ . Cauchy's interlace theorem (Parlett (1980, p. 186)) gives

$$(3.4) \quad \psi_1 \leq \sigma_{p_2+1},$$

so it will be true that

$$(3.5) \quad \sigma_{p_2} \leq \lambda_{p_2} \quad \text{and} \quad \psi_1 \leq \sigma_{p_2+1}.$$

It should be noted that generally  $\psi_1$  is unknown, and a reasonable guess is probably not even possible. However, an estimate for  $\lambda_{p_2}$  may be given, and the first inequality in (3.5) can be used to estimate  $p_2$ . In applications, it is not unreasonable to express a bound on the maximum proportion of unexplained error variation which is acceptable. Calling this factor  $\nu$ , in terms of standard deviations, we have

$$(3.6) \quad \left( \frac{\lambda_{p_2}}{(\beta' \Psi \beta)_{pp}} \right)^{1/2} \leq \nu.$$

For our rescaled model,  $(\beta' \Psi \beta)_{pp} + \lambda_{p_2} = 1$ , so that a useful bound on  $\sigma_{p_2}$  is

$$(3.7) \quad \sigma_{p_2} \leq \lambda_{p_2} \leq (1 + \nu^{-2})^{-1}.$$

Thus, given the eigenvalues of  $\Sigma_L$  and a maximum tolerated error variance, the value of  $p_2$  can be determined. Our algorithm provides for any nonnegative choice for  $\nu$ , and some sample upper bounds on  $\sigma_{p_2}$  for various  $\nu$  are summarized in Table 1.

Other methods can and have been used to indicate reasonable choices for  $p_2$ . A procedure

TABLE 1

Maximum % error tolerated (100ν)	Upper bound on $\sigma_{p_2}$
10	0.010
30	0.083
50	0.200
70	0.329
100	0.500
∞	1.000

sometimes used in factor analysis is to find  $p_2$  where  $\sigma_{p_2} < 1$  (Harman (1976, p. 86)) which corresponds to  $\nu = \infty$  in the LDA model where  $\Lambda$  is a diagonal matrix. Hakstian, Rogers and Cattell (1982) discuss several methods for determining  $p_2$  in a factor analysis setting. For nondiagonal  $\Lambda$ , the user can look for large gaps between the eigenvalues of the sample covariance matrix  $S$ . A theorem by Golub, Klema and Stewart (1976) states that the distance between the range of  $Z_1$  and the range of the best arbitrary (i.e., not restricted to columns of  $Z$ ) rank  $p_1$  approximation to  $Z$  is bounded by  $\lambda_{p_2}/\psi_1$ . Letting  $\gamma$  denote  $\psi_1/\lambda_{p_2}$ ,  $\gamma$  should then be as large as possible, certainly greater than 1, which implies  $\lambda_{p_2} < \psi_1$ . Since an upper bound for  $\gamma$  is  $\sigma_{p_2+1}/\sigma_{p_2}$ , the quantity  $\gamma_N = \gamma[\sigma_{p_2+1}/\sigma_{p_2}]^{-1}$  will be used as a normalized measure of  $\gamma$  that is bounded above by 1. Note that  $\sigma_{p_2+1}/\sigma_{p_2}$  is independent of the variable assignments. The above arguments imply that selection of  $p_2$  corresponding to a large gap in the eigenvalues has the potential for a good rank  $p_1$  approximation to  $Z$ . The effectiveness of the approximation can be evaluated by  $\gamma_N$  and  $\gamma$ .

**4. Identification of dependency relationships.** Consider the number of dependencies  $p_2$  fixed with the problem now being determination of the “best”  $p_1$  predictor variables. The assignment of a variable to the predictor or estimated variable subset corresponds to choosing column permutations of  $Z$ . Thus, (2.5) can be rewritten as

$$(4.1) \quad R = ZP \begin{bmatrix} -\beta \\ I \end{bmatrix},$$

where  $P$  is  $p \times p$  permutation matrix with  $PP' = I$ . Let the singular value decomposition of  $Z$  be given by

$$(4.2) \quad Z = \begin{matrix} U & W & V' \\ n \times p & p \times p & p \times p \end{matrix}$$

where  $W = \text{diagonal}(w_1, \dots, w_p)$  is the matrix of singular values of  $Z$  with  $w_1 \geq \dots \geq w_p$  and  $U'U = I, V'V = I$ . Since  $P'V$  forms an orthonormal basis for  $R^p$ , there exists a  $p \times p_2$  matrix  $C$  such that

$$(4.3) \quad P'VC = \begin{bmatrix} -\beta \\ I \end{bmatrix}.$$

Thus, if we combine (4.1), (4.2), and (4.3), the residual matrix becomes

$$(4.4) \quad R = UWV'P \begin{bmatrix} -\beta \\ I \end{bmatrix} = UWC.$$

A natural criterion for selection of the best permutation matrix  $P$  is to minimize the sum of squares of the residuals, which can be written in terms of the Frobenius norm

$$(4.5) \quad r = \|R\|_F^2 = \sum_{i=1}^n \sum_{j=1}^{p_2} r_{ij}^2 = \text{tr} R'R = \text{tr} C'W^2C = \sum_{i=1}^p \sum_{j=1}^{p_2} w_i^2 c_{ij}^2.$$

The problem of determining the permutation matrix  $P$  such that  $r$  is minimal results in a combinatorial optimization problem, that is, trying all possible combinations of estimated and predictor variables. We attack the problem by using a sequential variable deletion process which may reduce the cost of trying particular permutations by detecting permutations that result in a large  $r$  early in the calculations. Our algorithm examines all possible permutations, eliminating computations whenever possible and producing all potential dependency structures resulting in small residues. An outline of the procedure is presented below with specific details found in Ward et al. (1982).

Assume that a permutation matrix  $P_*$  is given. Then the  $\beta_*$  that minimizes  $r$  is given by

$$\beta_* = [W \quad V_{*1}]^+ [W \quad V_{*2}],$$

where

$$V'P_* = \begin{bmatrix} V_{*1} & V_{*2} \\ p \times p_1 & p \times p_2 \end{bmatrix}$$

and the superscript  $+$  denotes the Moore-Penrose inverse. Other equivalent formulations are  $\beta_* = Z_1^+ Z_2$  where  $ZP_* = [Z_1 \quad Z_2]$  and  $\beta_* = \Sigma_{11}^{-1} \Sigma_{12}$ , using (2.4) and (3.1) with  $P_*$  applied to  $X$ . Thus,

$$r_* = \left\| W V' P_* \begin{bmatrix} -\beta_* \\ I \end{bmatrix} \right\|_F^2.$$

Since the computation of  $\beta_*$  and  $r_*$  is relatively expensive, it is not practical to compute them for many permutation matrices. Instead, we note that those matrices  $[-\beta]$  corresponding to small residuals will have strong components in the subspace spanned by the columns of  $V_2$ , the last  $p_2$  columns of  $P'V$ , and weak components in its orthogonal complement (see Golub et al. (1976)). Using this  $p_2$ -dimensional subspace, (4.3) becomes

$$V_2 C_2 = \begin{bmatrix} -\beta \\ I \end{bmatrix},$$

yielding

$$(4.6) \quad C_2 = V_{22}^{-1}, \quad \beta = -V_{21} V_{22}^{-1}, \quad r = \|W_2 V_{22}^{-1}\|_F^2$$

where

$$V_2 = \begin{bmatrix} V_{21} \\ V_{22} \end{bmatrix}, \quad V_{22} \text{ is } p_2 \times p_2, \quad W = \begin{bmatrix} W_1 & 0 \\ 0 & W_2 \end{bmatrix}, \quad W_2 \text{ is } p_2 \times p_2.$$

The effect of this restriction of the basis space for  $C$  will be examined in the examples. It is shown that (4.6) approximates the full residual (4.5) with sufficient accuracy to enable determining reasonable choices of predictor variables. The computational saving can be appreciable if a large number of subsets is being evaluated.

Analogous to selecting variables in linear regression problems, a permutation which produces a residual close to the optimum can be found by attempting to maximize the determinant of  $V_{22}$ . Techniques which have been proposed in the past are Gaussian elimination on  $V_2$  with full pivoting and  $QR$  elimination on  $V_2$  with row pivoting. We use the latter



method to determine a permutation matrix  $P_0$  such that its residual  $r_0$  is close to optimal. For subsequent reference, the matrix  $P_0$  will be said to establish a “benchmark” permutation. By sequentially selecting variables to be estimated, computing a residual corresponding to the current subset of estimated variables, and comparing this residual to the benchmark residual  $r_0$ , the permutations corresponding to the poor choices for estimated variables can frequently be eliminated before the computations for those permutations are completed. By ordering the variables so that the most likely variables to be classified as predictor variables are eliminated first, the total number of permutations that must be tried in analyzing all possible subsets of predictor and estimated variables may be significantly reduced. For example, if  $p_2 = 4$  and variables 2 and 3 are important as predictor variables, all permutations involving these variables can be eliminated if the residual from just choosing these two as estimated variables is significantly larger than  $r_0$ .

The user has the option to compute the benchmark solution and eliminate the computations associated with trying all possible subsets. This option may be attractive when there are a large number of variables or to obtain an initial assessment of the problem.

**5. Parameter estimation and inference.** The predictor and estimated variables are now considered known since  $\mathbf{X}$  is now ordered, and standard estimation methods, such as those in Rao (1973, Chap. 8), can be used. Consider the sample covariance matrix  $S$  to be partitioned the same as  $\Sigma_L$  with

$$S = \begin{pmatrix} S_{11} & S_{12} \\ S'_{12} & S_{22} \end{pmatrix}.$$

Standard moment estimation gives

$$(5.1) \quad \begin{aligned} \hat{\Psi} &= S_{11}, & \hat{\boldsymbol{\mu}}_1 &= \bar{\mathbf{x}}_1, \\ \hat{\boldsymbol{\beta}} &= S_{11}^{-1} S_{12}, & \hat{\boldsymbol{\beta}}_0 &= \bar{\mathbf{x}}_2 - \hat{\boldsymbol{\beta}}' \bar{\mathbf{x}}_1, \\ \hat{\Lambda} &= S_{22} - S'_{12} S_{11}^{-1} S_{12} \equiv S_{22 \cdot 1}, \end{aligned}$$

where  $\Lambda$  is assumed a full matrix with  $p_2(p_2 + 1)/2$  parameters. The maximum likelihood (ML) estimates correspond to (5.1) if  $\mathbf{X}$  is assumed to have a multivariate normal distribution. This follows directly by noting that  $\mathbf{X}_1 \sim N(\boldsymbol{\mu}_1, \Psi)$  and  $\mathbf{X}_2 | \mathbf{X}_1 = \mathbf{x}_1 \sim N(\boldsymbol{\beta}_0 + \boldsymbol{\beta}' \mathbf{x}_1, \Lambda)$ . Thus, the likelihood function can be partitioned and the ML estimates obtained directly.

It may often be of interest to test whether  $\Lambda$  is a diagonal matrix since the interpretation of the LDA model is more appealing and the relationship (3.7) becomes useful. If  $\Lambda$  is assumed diagonal, the ML estimator is

$$(5.2) \quad \hat{\Lambda}_d = \text{diagonal}(S_{22} - S'_{12} S_{11}^{-1} S_{12}).$$

The hypothesis test of interest is

- $H_0$ :  $\Lambda$  a diagonal matrix;
- $H_1$ :  $\Lambda$  a full matrix.

The likelihood ratio test (Anderson (1958, p. 230)) is

$$(5.3) \quad LR = -2 \log \frac{L_{H_0}(\hat{\boldsymbol{\mu}}_1, \hat{\boldsymbol{\beta}}_0, \hat{\boldsymbol{\beta}}, \hat{\Psi}, \hat{\Lambda}_d)}{L_{H_1}(\hat{\boldsymbol{\mu}}_1, \hat{\boldsymbol{\beta}}_0, \hat{\boldsymbol{\beta}}, \hat{\Psi}, \hat{\Lambda})},$$

where  $L_{H_0}$  and  $L_{H_1}$  are the likelihoods under the null and alternative hypotheses, respectively. After some algebra, (5.3) simplifies to

$$(5.4) \quad \text{LR} = -n \log |\hat{\Lambda}_d^{-1} S_{22 \cdot 1}|.$$

Computation of (5.4) uses (2.2) and

$$(5.5) \quad \Sigma_L^{-1} = \begin{pmatrix} \Psi^{-1} + \beta \Lambda^{-1} \beta' & -\beta \Lambda^{-1} \\ -\Lambda^{-1} \beta' & \Lambda^{-1} \end{pmatrix}.$$

Since there are  $p_2(p_2 + 1)/2$  parameters in  $\Lambda$  under  $H_1$  and  $p_2$  parameters under  $H_0$ , LR has an asymptotic  $\chi^2$  distribution with  $p_2(p_2 - 1)/2$  degrees of freedom.

It is of interest to determine which variables are important in each of the dependencies. Corresponding to regression analysis, testing  $\beta_0$  and  $\beta$  should be accomplished by conditioning on  $X_1$ . The asymptotic distribution of the ML estimates  $\hat{\beta}_0, \hat{\beta}_1, \dots, \hat{\beta}_{p_1}$  given  $X_1$  is normal with mean  $\beta_0, \beta_1, \dots, \beta_{p_1}$  and covariance matrix

$$(5.6) \quad \Theta = n^{-1} \Lambda \otimes \begin{pmatrix} 1 + \bar{x}_1' S_{11}^{-1} \bar{x}_1 & -\bar{x}_1' S_{11}^{-1} \\ -S_{11}^{-1} \bar{x}_1 & S_{11}^{-1} \end{pmatrix},$$

which is obtained from the information matrix (Rao (1973, Chap. 5)). The unconditional asymptotic distribution of the  $\hat{\beta}$ 's is the same as the conditional distribution, but with  $\mu_1$  and  $\Psi$  replacing  $\bar{x}_1$  and  $S_{11}$  in (5.6). The resulting asymptotic 95% confidence intervals are

$$(5.7) \quad \begin{aligned} \hat{\beta}_0 \pm 1.96 [(1 + \bar{x}_1' S_{11}^{-1} \bar{x}_1) \hat{\Lambda}_d]^{1/2} n^{-1/2} \mathbf{e}, \\ \hat{\beta}_i \pm 1.96 [\delta_i \hat{\Lambda}_d]^{1/2} n^{-1/2} \mathbf{e} \end{aligned}$$

where  $\hat{\Lambda}$  replaces  $\Lambda$  in (5.6) and  $\delta_i$  is the  $i$ th diagonal of  $S_{11}^{-1}$ . From (5.6) as the dependencies become less variable the diagonal elements of  $\Lambda$  decrease, and the variation of the  $\hat{\beta}$ 's decreases. Also, as the predictors become more variable the diagonal elements of  $S_{11}^{-1}$  decrease, and the variation of the  $\hat{\beta}$ 's decreases.

**6. Competing dependencies.** An important feature of LDA is that various sets of estimated variables having small residuals are considered. Each set of variables corresponds to a different "competing" dependency structure. Note that the confidence intervals given in §5 enable the user to set certain  $\beta$ 's to zero to simplify a particular dependency.

A simple example illustrates the competing dependencies concept. Ignoring the error term, consider an example where  $p = 6$  and  $p_2 = 2$ .

*Case 1. Estimated variables  $x_5$  and  $x_6$ .*

$$\begin{aligned} x_5 &= x_1 + 2x_3 + x_4, \\ x_6 &= x_2 - 5x_4. \end{aligned}$$

*Case 2. Estimated variables  $x_2$  and  $x_3$ .*

$$\begin{aligned} x_2 &= 5x_4 + x_6, \\ x_3 &= -\frac{1}{2}x_1 - \frac{1}{2}x_4 + \frac{1}{2}x_5. \end{aligned}$$

*Case 3. Estimated variables  $x_4$  and  $x_6$ .*

$$\begin{aligned} x_4 &= -x_1 - 2x_3 + x_5, \\ x_6 &= 5x_1 + x_2 + 10x_3 - 5x_5. \end{aligned}$$

Each of these cases is a different formulation of the same dependency structure, and each has a different residual. The user may wish to select a particular set of estimated variables even if the residual is slightly larger. The criteria influencing these choices can

often be quite subjective. Usually one wishes to select the dependency structure which lends itself to the most meaningful physical interpretation, but other factors may be involved. For example, in data collection, some variables may be easily observed while others may be difficult and/or expensive to obtain. The user may wish to estimate the more difficult variables to obtain even at the cost of a larger, but acceptable, residual. Another consideration is the "simplicity" of a dependency structure, i.e., the number of zero coefficients. In our example above, the formulations in cases 1 and 2 would be considered simpler than that in case 3. Simplicity is often the key in meaningfully describing particularly large data sets.

As our algorithms can be used to identify and analyze all such competing dependencies, the user is able to choose the formulation most appropriate to any given application.

**7. Examples.** The LDA model is appropriate for various types of multivariate analyses. Four examples are considered. The first example deals with the collinearity problem in regression analysis. Beaton, Rubin and Barone (1976) considered the Longley data set where  $p = 6$  independent variables and  $n = 16$  observations. In this example there are a number of simple dependencies since there are several correlations greater than 0.99. The eigenvalues of the correlation matrix are 4.6, 1.2, 0.02, 0.015, 0.0026, 0.00038. A case could be made for selecting  $p_2$  to be 3 or 4, but  $p_2 = 3$  was selected to compare the LDA approach with Beaton's subjective selection of variables 2, 5, and 6 for elimination. The results of the variable selection considering all permutations of three variables appear in Table 2. The benchmark set of variables selected by the method described in §4 was the minimum residual variables 1, 5, and 6. However, this example illustrates that selecting competing dependencies can be useful. The second set of predicted variables 1, 2, and 5 is probably a better choice than the first set since the multiple correlation coefficient (i.e., considering the dependent variables and 3 predictor variables) is somewhat higher. The condition number  $k(\Psi)$  does not change appreciably.

TABLE 2  
LDA results for Longley's example.

Predicted variables			Residual	Reduced space residual	$\gamma$	$\gamma_N$	LR <sup>(a)</sup>	$R^2$	$\kappa(\Psi)$
1	5	6*	.545	.546	9.94	.73	11.5	.948	10.6
1	2	5	.608	.609	8.15	.60	21.2	.982	13.7
1	2	6	.789	.796	3.75	.28	49.6	.974	12.2
2	5	6	.927	.944	2.47	.18	73.2	.979	12.7

Asterisk denotes benchmark solution. <sup>(a)</sup>Chi-square 95% critical value is 7.8.

TABLE 3  
LDA results for Harman's factor analysis example.

Predicted variables			Residual	Reduced space residual	$\gamma$	$\gamma_N$	LR <sup>(a)</sup>
1	2	4	2.378	2.430	3.28	.39	4.3
2	3	4*	2.384	2.427	3.48	.42	4.5
3	4	5	2.619	2.746	2.08	.25	6.3
1	4	5	2.654	2.788	1.70	.20	6.8

Asterisk denotes benchmark solution. <sup>(a)</sup>Chi-square 95% critical value is 7.8.

The second example considers a factor analysis problem with  $p = 5$  and  $n = 12$  given in Harman (1976, Table 2.1). The eigenvalues of the correlation matrix are 2.9, 1.8, 0.21, 0.10, 0.015 and Harman chooses  $p_2 = 3$  which corresponds to about a 50% error LDA model. The LDA results appear in Table 3 where it is apparent that the diagonal error matrix  $\Lambda$  seems appropriate. The first two sets of predicted variables appear to give similar results. Using the confidence interval procedure discussed in §5 gives the dependencies shown in Table 4 (where \* denotes a nonzero  $\beta$  term in the dependency):

TABLE 4

Predicted variable	Variable		
	Intercept	3	5
1		*	
2	*		*
4	*	*	*

Predicted variable	Variable		
	Intercept	1	5
2	*		*
3		*	
4	*	*	*

Either set of competing dependencies seems to give an equally simple model. The high correlations  $\text{corr}(X_2, X_5) = 0.86$  and  $\text{corr}(X_1, X_3) = 0.97$  appear to play an important role. Note that the minimum residual does not correspond to the benchmark solution. The rotated factor pattern matrix using a varimax rotation in a factor analysis appears in Table 5. While factor 2 is dominated by the high  $X_1$  and  $X_3$  correlation, the information is different in the two analysis approaches.

TABLE 5

Variable	Factor 1	Factor 2
1	0.02	0.99
2	0.94	-0.01
3	0.14	0.98
4	0.82	0.45
5	0.97	-0.01

The third example is application of LDA to a linear discriminant analysis problem involving Fisher's (1936) iris data where  $p = 4$  and  $n = 150$  with 3 groups present. The 150 observations were centered by their respective group means so that a common mean and correlation matrix would be appropriate for all observations. The eigenvalues of the correlation matrix are 2.5, 0.73, 0.58, and 0.19;  $p_2 = 1$  was selected. The results of the LDA procedure appear in Table 6. Note that elimination of variable 1 (sepal length) gives 3 misclassified samples which remains unchanged when all 4 variables are used. Typically, the last two cases would not be considered since  $\gamma < 1$ .

TABLE 6  
LDA results for Fisher's discriminant analysis example.

Predicted variable	Residual	Reduced space residual	$\gamma$	$\gamma_N$	Number misclassified
1	7.21	7.91	1.40	.46	3
3	7.38	8.29	1.22	.39	6
2	9.51	18.79	.38	—	4
4	9.83	21.62	.34	—	5

The final example is an application of LDA to a principal components analysis problem given in Ahamad (1967) where  $p = 18$  and  $n = 14$ . The eigenvalues of the correlation matrix are 10.0, 3.1, 1.3, 1.1, 0.94, 0.80, 0.30, 0.22, 0.14, 0.031, 0.016, 0.012, 0.0045, 0, 0, 0, 0, 0. A value of  $p_2 = 9$  seems appropriate. The benchmark predicted variables are 2, 4, 5, 6, 7, 8, 9, 11, and 12 which has a residual of 1.81. For  $p_2 = 9$  there are 48,620 possible subsets which would require excessive computational time. The benchmark case would probably be considered adequate by most users even if the minimum residual has not been obtained.

Considering  $p_2 = 5$  enables the user to retain the maximum number of variables. Table 7 gives a partial list of variables that all result in a residual of essentially zero. There are over 8,000 such formulations. Note that ranking the competing dependencies by the computed residual is done for convenience only since all sets have comparable residuals. The benchmark subset is 2, 7, 8, 11, 12 which has a residual of  $1.68 \times 10^{-15}$ .

TABLE 7  
LDA results for Ahamad's principal component example.

Predicted variables	Residual ( $\times 10^{-15}$ )
7,8,11,12,13	1.54
5,7,8,11,13	1.57
7,8,9,11,12	1.58
4,7,11,12,13	1.58
3,7,8,11,12	1.61
7,10,11,12,13	1.62
5,7,8,9,11	1.62
6,7,8,11,12	1.62
7,9,11,12,13	1.63
7,8,9,11,18	1.65

**8. Conclusion.** Dependencies can play an important role in almost any type of multivariate analysis. The LDA procedure provides a framework in which these dependencies can be determined. Computation of the benchmark solution is straightforward and can be performed easily on modern computers. However, the user may wish to determine the competing dependencies, which is a combinatorial problem that can require appreciable computational time for a large number of variables. The examples illustrate the potential usefulness of LDA in four different types of multivariate analyses.

## REFERENCES

- B. AHAMAD (1967), *An analysis of crimes by the method of principal components*, Appl. Stat., 16, pp. 17–35.
- T. W. ANDERSON (1958), *An Introduction to Multivariate Statistical Analysis*, John Wiley, New York.
- A. E. BEATON, D. B., RUBIN AND J. L. BARONE (1976), *The acceptability of regression solutions: Another look at computational accuracy*, J. Amer. Statist. Assoc., 71, pp. 158–168.
- D. A. BELSEY, E. KUH AND R. E. WELSCH (1980), *Regression Diagnostics*, John Wiley, New York.
- R. A. FISHER (1936), *The use of multiple measurements in taxonomic problems*, Ann. Eugenics, 7, pp. 179–188.
- G. GOLUB, V. KLEMA AND G. W. STEWART (1976), *Rank degeneracy and least squares problems*, Technical Report STAN-SC-76-559, Stanford Univ., Stanford, CA.
- A. R. HAKSTIAN, W. T. ROGERS AND R. B. CATTELL (1982), *The behavior of number-of-factors rules with simulated data*, Multi-var. Behav. Res., 17, pp. 193–219.
- H. H. HARMAN (1976), *Modern Factor Analysis*, Univ. Chicago Press, Chicago.
- G. P. MCCABE (1982), *Principal variables*, Univ. Tech. Rep. 82–3, Purdue Univ., West Lafayette, IN.
- B. N. PARLETT (1980), *The Symmetric Eigenvalue Problem*, Prentice-Hall, Englewood Cliffs, NJ.
- C. R. RAO (1973), *Linear Statistical Inference and Its Applications*, John Wiley, New York.
- R. C. WARD, G. J. DAVIS AND V. E. KANE (1982), *An algorithm for assessing linear dependencies in multivariate data*, ACM Trans. Math. Software, to appear.

## ALGORITHMS FOR NONLINEAR LEAST SQUARES WITH LINEAR INEQUALITY CONSTRAINTS\*

S. J. WRIGHT† AND J. N. HOLT‡

**Abstract.** Two algorithms for solving nonlinear least squares problems with general linear inequality constraints are described. At each step, the problem is reduced to an unconstrained linear least squares problem in the subspace defined by the active constraints, which is solved using the Levenberg–Marquardt method. The desirability of leaving an active constraint is evaluated at each step, using a different technique for each of the two algorithms. Each step is constrained to be within a circular region of trust about the current approximate minimizer, whose radius is updated according to the quality of the step after each iteration. Comparisons of the relative performance of the two algorithms on small problems and on a larger exponential data-fitting problem are presented.

**Key words.** nonlinear least squares, linear inequality constraints, Levenberg–Marquardt method, singular value decomposition

**1. Introduction.** A great deal has been published recently on the unconstrained nonlinear least squares problem. The best-known algorithms include those of Gill and Murray [6], Moré [15], and Dennis, Gay, and Welsch [3]. (See also Dennis [2] and Nazareth [16] for reviews.) An interesting comparison of the performance of several widely available algorithms on a standard set of test problems has been presented by Hiebert [9].

Little has appeared, however, on the constrained nonlinear least squares problem. Holt and Fletcher [11] deal with certain simple linear constraints, and Lindström [14] presents an algorithm for nonlinear constraints. Of course, these problems could also be handled by standard nonlinear programming codes, but these do not exploit the structure of the sum-of-squares objective function.

The algorithm of Holt and Fletcher allows only bounds and monotonicity constraints on the variables. The two algorithms described here are based on the reasoning of that paper, but allow for general linear constraints. One is a direct extension of the method of Holt and Fletcher, while the other makes use of the method of Bunch and Nielsen [1] for updating least squares solutions when an extra variable (corresponding to relaxation of an active constraint) is added to the problem.

Computational experience with the two algorithms is reported, using some standard problems with added constraints as a basis for comparison.

### 2. Overview of the algorithms.

**2.1. Statement of the problem.** We consider the following problem:

$$(2.1) \quad \underset{\mathbf{x}}{\text{minimize}} \ z(\mathbf{x}) = \|\mathbf{f}(\mathbf{x})\|_2^2 = \sum_{i=1}^m (f_i(\mathbf{x}))^2$$

$$(2.2) \quad \text{subject to } A^T \mathbf{x} \geq \mathbf{b},$$

where  $\mathbf{x} \in E^n$ ,  $\mathbf{b} \in E^s$ ,  $A \in E^n \times E^s$ , and  $f_i \in C^2$ ,  $i = 1, 2, \dots, m$ . There must be at least one point  $\mathbf{x} \in E^n$  which satisfies (2.2) (i.e. is feasible). At any such point, we divide the columns of  $A$  into two matrices  $C$  and  $D$ , and divide  $\mathbf{b}$  correspondingly into two vectors  $\mathbf{c}$  and  $\mathbf{d}$ , such that

$$(2.3) \quad \begin{aligned} C^T \mathbf{x} &= \mathbf{c}, & C &\in E^n \times E^t, & t &\leq s, \\ D^T \mathbf{x} &> \mathbf{d}, & D &\in E^n \times E^{s-t}. \end{aligned}$$

\*Received by the editors October 19, 1982, and in revised form March 29, 1984.

†Department of Electrical and Computer Engineering, University of Arizona, Tucson, Arizona 85721.

‡Department of Mathematics, University of Queensland, St. Lucia, Brisbane 4067, Australia.

Constraints corresponding to  $C$  are known as active constraints at  $\mathbf{x}$ , while those in  $D$  are inactive constraints at  $\mathbf{x}$ . We will refer to the set of indices of active constraints at  $\mathbf{x}$  as  $\mathcal{A}(\mathbf{x})$ . It is assumed that for all feasible  $\mathbf{x}$ , the matrix  $C$  has full rank, otherwise there is redundancy in the constraint specification.

The algorithms also allow for linear equality constraints, which are permanently stored in the matrix  $C$ , and are treated in the same way as active constraints except that they are not considered for deletion. We can thus virtually eliminate them from the problem, and they are not discussed further.

Throughout the paper,  $\|\cdot\|$  will denote  $\|\cdot\|_2$ .

**2.2. Principle of the algorithms.** Given an initial point  $\mathbf{x}^{(0)}$ , we aim to generate a sequence of updates  $\{\delta\mathbf{x}^{(k)}\}$ ,  $k = 0, 1, \dots$ , so that  $\{\mathbf{x}^{(k+1)}\}$  defined by

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \delta\mathbf{x}^{(k)}$$

converges to a constrained stationary point of (2.1), while retaining feasibility at each iteration.

We will need to make use of the concept of a “working set” of constraints. That is, a set of constraints assumed to hold as equality constraints throughout some subsection of the algorithm. (See Gill, Murray and Wright [7].) Such a set will be denoted by  $\mathcal{A}_w$ .

The general iteration of either algorithm, from a current iterate  $\mathbf{x}$  with active constraint set  $\mathcal{A}(\mathbf{x})$ , is described by the informal code in Fig. 1.

- (i) compute a candidate update  $\mathbf{y}_0$  using  $\mathcal{A}_w \equiv \mathcal{A}(\mathbf{x})$ , set  $\mathcal{A}_{w^*} \equiv \mathcal{A}_w$  and  $\mathbf{y}^* = \mathbf{y}_0$ .
- (ii) **if** (an anti-zigzagging test is passed)
  - then if** (not “crude” convergence)
    - then** compute further candidate updates  $\mathbf{y}_i$ , using  $\mathcal{A}_w = \mathcal{A}(\mathbf{x}) - i$ ,  $i \in \mathcal{A}(\mathbf{x})$ 
      - if** (one of these, say  $\mathbf{y}_r$ , is acceptable according to predictive criterion)
        - then** set  $\mathcal{A}_{w^*} = \mathcal{A}(\mathbf{x}) - r$  and  $\mathbf{y}^* = \mathbf{y}_r$ .
      - else** compute first-order Lagrange multiplier estimates,  $\lambda_i$ ,  $i \in \mathcal{A}(\mathbf{x})$ . Let  $\lambda_q = \min_i(\lambda_i)$ .
        - if** ( $\lambda_q < 0$ )
          - then** compute a candidate update  $\mathbf{y}_q$  using  $\mathcal{A}_w \equiv \mathcal{A}(\mathbf{x}) - q$ , set  $\mathcal{A}_{w^*} \equiv \mathcal{A}_w$  and  $\mathbf{y}^* = \mathbf{y}_q$ .
          - else** set “fine” convergence flag, go to (iii).
- (iii) scale  $\mathbf{y}^*$  if necessary to avoid infeasibility, i.e.  $\mathbf{y}^* := \alpha\mathbf{y}^*$ ,  $0 < \alpha \leq 1$ .
- (iv) **if** (sufficient function decrease at  $\mathbf{x} + \mathbf{y}^*$ )
  - then** set  $\delta\mathbf{x} = \mathbf{y}^*$  and update  $\mathbf{x}$  to  $\mathbf{x} + \delta\mathbf{x}$ , increase trust region if appropriate.
  - else** decrease trust region,
    - generate a new candidate update using  $\mathcal{A}_{w^*}$ ,
    - go to (iii)

FIG. 1. Informal code for a general iteration of either algorithm.

In step (i) of the iteration, we solve the constrained linear least squares problem

$$(2.4) \quad \underset{\mathbf{y}}{\text{minimize}} \{S_0 = \|\mathbf{f} + J\mathbf{y}\|^2, \|\mathbf{y}\| \leq h, C^T\mathbf{y} = \mathbf{0}\}$$

where  $\mathbf{f}$  and  $J$  are the function value vector and its Jacobian matrix at  $\mathbf{x}$ . The condition  $C^T\mathbf{y} = \mathbf{0}$  ensures that  $\mathcal{A}_w = \mathcal{A}(\mathbf{x})$ .

$h$  is a parameter defining a region of trust for the linearization. The region of trust is dynamically adjusted at each iteration as described in §6. Basically, it will be small if, in recent iterations, the actual reduction in the sum of squares has been small relative to the reduction predicted by the linear model and large if the prediction has been accurate. As



we shall see in §3, (2.4) can be reformulated as an unconstrained linear least squares problem in  $E^{n-t}$ .

As can be seen from the informal code, in step (ii) of the iteration, the possibility exists of relaxing the working set to generate further candidate updates. We aim here to solve the problems

$$(2.5) \quad \underset{\mathbf{y}}{\text{minimize}} \{S_i = \|\mathbf{f} + J\mathbf{y}\|^2, \|\mathbf{y}\| \leq h, \bar{C}^T \mathbf{y} = \mathbf{0}\}, \quad i = 1, \dots, t.$$

Each of these problems attempts to predict the effect of relaxing one of the constraints active at  $\mathbf{x}$ .  $\bar{C}$  is the matrix  $C$  of (2.3) with the  $i$ th column,  $\mathbf{c}_i$ , deleted. The constraints  $\bar{C}^T \mathbf{y} = \mathbf{0}$  ensure that  $\mathcal{A}_w = \mathcal{A}(\mathbf{x}) - i$ .  $\mathbf{y}_i$ , the solution of (2.5), can only be accepted as a feasible candidate update if

$$\mathbf{c}_i^T \mathbf{y}_i \geq 0,$$

indicating movement into the feasible region. Eventually, when “crude” convergence applies, Lagrange multiplier estimates are used as in conventional active set methods.

The crucial difference between the two algorithms presented in this paper is that the first computes approximate solutions to (2.5), while the second solves (2.5) exactly.

In subsequent sections, the informal concepts introduced in Fig. 1 are formalized, numerical details of the computation are explained, and convergence results are presented. Finally some numerical examples complete the paper.

**3. Solution of problem (2.4).** For the matrix  $C$  of active constraints at  $\mathbf{x}$ , we can find matrices  $Q$  and  $R$  such that

$$QC = \begin{bmatrix} R \\ \dots \\ 0 \end{bmatrix}, \quad Q \in E^n \times E^n, QQ^T = Q^T Q = I, R \in E^t \times E^t, \text{ upper triangular.}$$

If we define  $n' = n - t$ , and partition  $Q$  as

$$Q = \begin{bmatrix} Q_1 \\ \dots \\ Q_2 \end{bmatrix} \begin{matrix} \}t \\ \\ \}n' \end{matrix}$$

then

$$(3.1) \quad \mathbf{0} = C^T \mathbf{y} = [R^T \ ; \ 0] Q \mathbf{y} = R^T Q_1 \mathbf{y} \Rightarrow Q_1 \mathbf{y} = \mathbf{0}.$$

We define the vector  $\mathbf{p}$  by

$$Q \mathbf{y} = \begin{bmatrix} Q_1 \\ \dots \\ Q_2 \end{bmatrix} \mathbf{y} = \begin{bmatrix} \mathbf{p}_1 \\ \dots \\ \mathbf{p}_2 \end{bmatrix} = \mathbf{p};$$

then from (3.1),  $\mathbf{p}_1 = \mathbf{0}$  and hence

$$(3.2) \quad \mathbf{y} = Q_2^T \mathbf{p}_2.$$

Since  $\|\mathbf{y}\| = \|\mathbf{p}\| = \|\mathbf{p}_2\|$ , (2.4) becomes

$$(3.3) \quad \underset{\mathbf{p}_2}{\text{minimize}} \{S_0 = \|\mathbf{f} + JQ_2^T \mathbf{p}_2\|^2, \|\mathbf{p}_2\| \leq h\}.$$

It is a simple matter to show that the solution of (3.3) can be written as the solution  $\mathbf{p}_2(\lambda)$ , for some  $\lambda \geq 0$ , of the unconstrained problem

$$(3.4) \quad \underset{\mathbf{p}_2}{\text{minimize}} \{ \|\mathbf{f} + JQ_2^T \mathbf{p}_2\|^2 + \lambda^2 \|\mathbf{p}_2\|^2 \}.$$

See for example Lawson and Hanson [13]. That is

$$(3.5) \quad \mathbf{p}_2(\lambda) = -((JQ_2^T)^T(JQ_2^T) + \lambda^2 I)^{-1}(JQ_2^T)^T \mathbf{f}.$$

If  $JQ_2^T$  is rank deficient, then we find  $\mathbf{p}_2(0)$  by taking the limit of (3.5) to get

$$\mathbf{p}_2(0) = -(JQ_2^T)^\dagger \mathbf{f}$$

where  $(JQ_2^T)^\dagger$  is the Moore–Penrose pseudoinverse of  $JQ_2^T$ . If  $\|\mathbf{p}_2(0)\| \leq h$ , then  $\mathbf{p}_2(0)$  is the solution to (3.3). Otherwise there is a  $\lambda^* > 0$  such that  $\|\mathbf{p}_2(\lambda^*)\| = h$ , in which case  $\mathbf{p}_2(\lambda^*)$  is the unique solution to (3.3).

To solve (3.5), we find the singular value decomposition (SVD) of  $JQ_2^T$ , written as

$$JQ_2^T = U \begin{bmatrix} S \\ \dots \\ 0 \end{bmatrix} V^T$$

where  $U \in E^n \times E^n$  orthogonal,  $V \in E^{n'} \times E^{n'}$  orthogonal, and  $S \in E^{n'} \times E^{n'}$  diagonal, with diagonal elements  $s_1 \geq s_2 \geq \dots \geq s_r > 0$ , where  $r \leq n'$  is the rank of  $JQ_2^T$ . The  $s_i$ 's are the singular values of the matrix. The solution of (3.5) is then

$$(3.6) \quad \mathbf{p}_2(\lambda) = - \sum_{i=1}^r \frac{(\mathbf{u}_i^T \mathbf{f}) s_i}{s_i^2 + \lambda^2} \mathbf{v}_i = -VS^+T\bar{U}^T \mathbf{f},$$

where  $\bar{U}$  consists of the first  $n'$  columns of  $U$ ,  $S^+$  is the diagonal matrix of order  $n'$  with diagonal elements

$$s_{ii}^+ = \begin{cases} s_i^{-1}, & i = 1, 2, \dots, r, \\ 0, & i > r, \end{cases}$$

and  $T$  is the diagonal matrix of order  $n'$  with diagonal elements

$$t_{ii} = \begin{cases} s_i^2 / (s_i^2 + \lambda^2), & i = 1, 2, \dots, r, \\ 0, & i > r. \end{cases}$$

We note from (3.6) that

$$(3.7) \quad \|\mathbf{p}_2(\lambda)\|^2 = \sum_{i=1}^r \frac{(\mathbf{u}_i^T \mathbf{f})^2 s_i^2}{(s_i^2 + \lambda^2)^2}$$

and since we require  $\|\mathbf{p}_2(\lambda)\| \leq h$ , we choose  $\lambda$  to be the smallest nonnegative value for which this constraint holds. The numerical solution of the resulting equation is discussed in §6.

In some algorithms, scaling is introduced into (2.4) by replacing the condition  $\|\mathbf{y}\| \leq h$  with  $\|B\mathbf{y}\| \leq h$ , where  $B$  is some diagonal matrix.

**4. Handling of constraints.**

**4.1. Constraint relaxation (problem (2.5)).** We now consider reformulation of the problem (3.4) to take account of relaxation of one of the active constraints.

Suppose the  $i$ th active constraint is to be relaxed. Recalling that  $\bar{C}$  is the matrix  $C$  with column  $i$  deleted, then

$$Q\bar{C} = \begin{bmatrix} \tilde{R} \\ \dots \\ 0 \end{bmatrix}$$

where  $\tilde{R}$  is the matrix  $R$  with column  $i$  deleted. To return  $\tilde{R}$  to upper triangular form, we need to premultiply by  $(t - i)$  Givens rotations (see, for example, Lawson and Hanson [13]):

$$(G_t G_{t-1} \cdots G_{i+1}) Q \bar{C} = \begin{bmatrix} \bar{R} \\ \cdots \\ 0 \end{bmatrix}$$

where  $\bar{R} \in E^{t-1} \times F^{t-1}$  is upper triangular. Thus  $\bar{Q}$  defined by

$$\bar{Q} = G_t \cdots G_{i+1} Q$$

is the orthogonal matrix which reduces  $\bar{C}$  to upper triangular form. Since the Givens rotation  $G_j$  affects only rows  $j - 1$  and  $j$  of the matrix  $Q$ , it can be seen that the last  $n'$  rows of  $Q$  and  $\bar{Q}$  are the same, hence  $\bar{Q}$  has the form

$$\bar{Q} = \begin{bmatrix} \bar{Q}_1 \\ \mathbf{q}_i^T \\ Q_2 \end{bmatrix}$$

where  $\bar{Q}_1$  has  $(t - 1)$  rows and  $\mathbf{q}_i$  is a column  $n$ -vector.

The resulting augmented linear least squares problem has an extra variable, and can be written as

$$(4.1) \quad \underset{\mathbf{p}_2}{\text{minimize}} \|J[\mathbf{q}_i \ ; \ Q_2^T] \bar{\mathbf{p}}_2 + \mathbf{f}\|^2 + \lambda^2 \|\bar{\mathbf{p}}_2\|^2.$$

We write  $\bar{\mathbf{p}}_2^T = [\rho \ ; \ \mathbf{p}_2^T]$  so that  $\rho \mathbf{q}_i$  is the component of  $\mathbf{y}$  which is normal to the manifold of currently active constraints.

To ensure that the  $i$ th constraint is indeed being relaxed and not violated, we require that

$$(4.2) \quad 0 < \mathbf{c}_i^T \mathbf{y} = \mathbf{c}_i^T [Q_2^T \bar{\mathbf{p}}_2 + \rho \mathbf{q}_i] = \rho \mathbf{c}_i^T \mathbf{q}_i,$$

so  $\rho$  must be nonzero with the same sign as  $\mathbf{c}_i^T \mathbf{q}_i$ .

Methods for finding a solution of (4.1) are discussed in §5.

**4.2. Addition of a constraint to the active set.** Once the search direction has been determined by selecting an acceptable candidate update, say  $\mathbf{s}$ , we may not be able to take the full step, since this may cause violation of previously inactive constraints. Using the notation of (2.3), we choose  $\alpha$  as the largest value in the interval  $(0, 1]$  for which

$$(4.3) \quad D^T(\mathbf{x} + \alpha \mathbf{s}) \cong \mathbf{d}$$

is satisfied, where, from (2.3),  $D^T$  describes the inactive constraints at  $\mathbf{x}$ . If strict inequality holds in (4.3) for  $\alpha = 1$ , then no new constraints have been encountered. Otherwise, we add a constraint to the active set by removing the appropriate column from  $D$  and appending it to the active constraint matrix  $C$ . Then

$$Q[C \ ; \ \mathbf{c}_{\text{new}}] = \begin{bmatrix} R \\ \cdots \\ 0 \end{bmatrix} \begin{bmatrix} Q\mathbf{c}_{\text{new}} \end{bmatrix}$$

and we update the  $QR$  decomposition by premultiplying by a Householder reflector.  $R$  is not changed in this process.

**5. Solution of the augmented problem (4.1) and constraint relaxation criteria.** The working set of constraints here is the active set at  $\mathbf{x}$  less one constraint.

**5.1. Method of the first algorithm.** In the first algorithm, we use a simple method of finding an approximate solution to the augmented problem (4.1), given the SVD of  $JQ_2^T$ , which is used in the solution of the original problem (3.7). The method is an extension of the method of Holt and Fletcher [11] for bounds and monotonicity constraints.

When  $\bar{\mathbf{p}}_2^T = [\rho \mid \mathbf{p}_2]$  we can rewrite (4.1) as

$$(5.1) \quad \underset{\rho, \mathbf{p}_2}{\text{minimize}} \quad \|JQ_2^T \mathbf{p}_2 + (\mathbf{f} + \rho J\mathbf{q}_i)\|^2 + \lambda^2 \rho^2 + \lambda^2 \|\mathbf{p}_2\|^2.$$

For fixed  $\rho$ , the solution of (5.1) is

$$(5.2) \quad \mathbf{p}_2 = - \sum_{j=1}^r \frac{\mathbf{u}_j^T (\mathbf{f} + \rho J\mathbf{q}_i) s_j}{s_j^2 + \lambda^2} \mathbf{v}_j,$$

where  $\mathbf{u}_j, s_j, \mathbf{v}_j$  arise from the SVD of  $JQ_2^T$ . If we use the notation

$$\beta_{ij}(\rho) = \mathbf{u}_j^T \mathbf{f} + \rho \mathbf{u}_j^T J\mathbf{q}_i,$$

then, from Lawson and Hanson [13], the predicted sum of squares at the solution will be

$$(5.3) \quad P_i(\rho) = \sum_{j=1}^r \beta_{ij}^2(\rho) \left[ \frac{\lambda^2}{s_j^2 + \lambda^2} \right]^2 + \sum_{j=r+1}^m \beta_{ij}^2(\rho).$$

Following Holt and Fletcher, we define vectors  $\boldsymbol{\eta}$  and  $\boldsymbol{\kappa}$  by

$$\boldsymbol{\eta}_j = \begin{cases} \beta_{ij}(0) \frac{\lambda^2}{s_j^2 + \lambda^2}, & j = 1, \dots, r, \\ \beta_{ij}(0), & j > r, \end{cases}$$

$$\boldsymbol{\kappa}_j = \begin{cases} \mathbf{u}_j^T J\mathbf{q}_i \frac{\lambda^2}{s_j^2 + \lambda^2}, & j = 1, \dots, r, \\ \mathbf{u}_j^T J\mathbf{q}_i, & j > r. \end{cases}$$

Then from (5.3),

$$P_i(\rho) = \boldsymbol{\eta}^T \boldsymbol{\eta} + \rho^2 \boldsymbol{\kappa}^T \boldsymbol{\kappa} + 2\rho \boldsymbol{\eta}^T \boldsymbol{\kappa},$$

and hence the minimum sum of squares is attained when

$$\rho = \rho_i^* = - \frac{\boldsymbol{\eta}^T \boldsymbol{\kappa}}{\boldsymbol{\kappa}^T \boldsymbol{\kappa}}.$$

Using (4.2), we check to see that  $\rho_i^*$  has the correct sign. If so, we compare it with two positive tolerances  $\varepsilon_1$  and  $\varepsilon_2$ . If  $|\rho_i^*| < \varepsilon_1$ , then it is too small and we do not consider constraint  $i$  for relaxation. If  $|\rho_i^*| > \varepsilon_2$ , then its magnitude is decreased to  $\varepsilon_2$ .

Once  $P_i(\rho_i^*) = P_i^*$  has been calculated for each active constraint, we find the minimum, say  $P_k^*$ . Following Holt and Fletcher, the  $k$ th constraint is relaxed if

$$P_k^* < \frac{1}{2} P(0),$$

where  $P(0)$  is the predicted minimum value of the sum of squares on the current active constraint manifold.

The value of  $\lambda$  used during the solution of the augmented problem is the same value used in (3.4) to ensure that  $\|\mathbf{p}_2\| \leq h$ . However, despite the limit on the size of  $\rho$ , since the working set has changed, this  $\lambda$  may not be optimal for the constraint  $\|\bar{\mathbf{p}}_2\| \leq h$ . We could be unwittingly altering the region of trust in solving the augmented problem. This difficulty

is overcome in the second algorithm, which allows the optimal choice of  $\lambda$  for the augmented problem.

**5.2. Method of the second algorithm.** In this algorithm, we explicitly find the solution of (4.1) by updating the SVD of  $JQ_2^T$  to take account of the appended column  $J\mathbf{q}_i$ . This allows us to solve directly for  $\bar{\mathbf{p}}_2$  and to choose  $\lambda_i$  as the smallest nonnegative value of  $\lambda$  such that  $\|\bar{\mathbf{p}}_2\| < h$ .

The SVD update technique is due to Bunch and Nielsen [1]. To find the solution  $\bar{\mathbf{p}}_2$ , we need the updated values of the matrices  $S$  and  $V$ , and the updated value of  $U^T\mathbf{f}$ . By solving the secular equation

$$(5.4) \quad g(\tau) = 1 + \sum_{j=1}^{n'} \frac{(U^T J \mathbf{q}_i)_j^2}{s_j^2 - \tau} - \frac{\sum_{j=n'+1}^m (U^T J \mathbf{q}_i)_j^2}{\tau} = 0$$

for  $\tau_1, \dots, \tau_{n'+1}$ , we obtain the squares of the new singular values  $\tilde{s}_k$ . In solving (5.4), use is made of theorems concerning distribution of the singular values. Deflation of the problems occurs when

- (i)  $JQ_2^T$  is rank deficient,
- (ii) some of the singular values of  $JQ_2^T$  are equal,
- (iii) any of the first  $n' + 1$  elements of  $U^T J \mathbf{q}_i$  are zero.

For each new singular value  $\tilde{s}_k$ , we define a diagonal matrix  $T_k \in E^{n'} \times E^{n'}$ , with  $j$ th diagonal element  $(s_j^2 - \tilde{s}_k^2)$ . The  $k$ th column of the updated  $V$  matrix is then given by

$$\tilde{\mathbf{v}}_k = \frac{1}{\gamma_k} \begin{bmatrix} \|J\mathbf{q}_i\| V T_k^{-1} S \bar{\mathbf{w}} \\ -1 \end{bmatrix}$$

where  $\bar{\mathbf{w}}$  consists of the first  $n'$  elements of  $U^T J \mathbf{q}_i$ , and  $\gamma_k$  is a positive scalar chosen to normalize  $\tilde{\mathbf{v}}_k$ . The  $k$ th element of the updated  $U^T\mathbf{f}$  vector is given by the formula

$$\tilde{\mathbf{u}}_k^T \mathbf{f} = \frac{1}{\gamma_k \tilde{s}_k} [\|J\mathbf{q}_i\| \bar{\mathbf{w}}^T T_k^{-1} S^2 (U^T \mathbf{f}) - (J\mathbf{q}_i)^T \mathbf{f}].$$

Once the updated SVD has been obtained, we calculate  $\lambda_i$  to fit the step length parameter  $h$ . Then a formula similar to (5.3) is used to calculate the predicted sum of squares  $P_i^*$  if constraint  $i$  is the one being relaxed. If  $P_i$  is the best result so far obtained, we explicitly find  $\bar{\mathbf{p}}_2$  and check the sign of  $\rho$ ; if this is correct, we calculate  $y$  and store it.

After all active constraints have been checked, and  $P_k^*$  is the minimum predicted sum of squares, we relax the  $k$ th active constraint if

$$P_k^* < (.9)P(0).$$

The relaxation threshold is higher than in the first algorithm because we have found the true solution of the augmented problem, and not just an approximation to it.

**5.3. Complexity of the algorithms.** Operations common to both algorithms include the formation of  $JQ_2^T$ , the formation of  $\mathbf{q}_i$  and  $J\mathbf{q}_i$  for each active constraint, the SVD of  $JQ_2^T$ , and function and Jacobian evaluations. Together, these require  $O(mn^2)$  multiplications and divisions.

The first algorithm requires very little extra work, only  $4m + 5n$  operations per constraint. Most of the work in the second algorithm lies in the solution of the secular equation (5.4) during the SVD update. The complexity is a fairly large multiple of  $n^2$  and  $n$ . For small- and medium-sized problems, say  $n \leq 10$ ,  $m \leq 40$ , the first algorithm can be significantly faster. For larger problems, however, the operations in common to the two algorithms tend to take much longer than those peculiar to each algorithm. The second algorithm is

avored in these cases because it usually takes fewer iterations to reach a solution. These observations are borne out by the numerical results of §8.

It would be possible to implement an algorithm similar in principle to those described here, but using the  $QR$  factorization of  $JQ_2$  instead of the singular value decomposition. Firstly, the solution of the augmented problem would be easier, since it is a trivial matter to update a  $QR$  factorization after the addition of a column. On the other hand, solution of (6.1) (i.e. selection of damping parameter  $\lambda$  to fit step size parameter  $h_{\max}$ ) becomes more lengthy since the quantities  $U_i^T \mathbf{f}$  and  $s_i$  are not explicitly available. Instead, it will be necessary to calculate  $\mathbf{y}$  for a number of different values of  $\lambda$ , as in Moré [15]. Since at each iteration we need to solve a number of equations similar in form to (6.1) (one for each augmented problem), this represents substantial extra work. Because of these two opposing considerations, it is not possible to say whether the use of a  $QR$  factorization would result in faster or slower algorithms in general than the ones presented here. This would depend on the size of  $JQ_2$  and the size of the actual constraint set at each step.

**6. Trust region control.** A global trust region parameter,  $h_{\max}$ , is maintained and updated as the iterations progress. There are good reasons for explicitly considering  $h_{\max}$  rather than the corresponding  $\lambda$ . (See Fletcher [5].) The rules for updating  $h_{\max}$  are described later in this section. The value of  $h_{\max}$  at the start of an iteration measures the maximum allowable stepsize permitted in this iteration. Of course, it may not be achievable because it may exceed the current Gauss–Newton steplength (i.e. corresponding to  $\lambda = 0$ ). For this reason we maintain, in a given iteration, a “working” trust region,  $h$ , and the corresponding damping parameter  $\lambda$ , derived from  $h_{\max}$  as follows:

- (i) Compute  $h_0$  (corresponding to  $\lambda = \lambda_{\min}$ , where  $\lambda_{\min} > 0$ , the minimum allowable value of  $\lambda$ , is necessary for the convergence properties of §7 to hold).
- (ii) Compute  $h_{\text{critical}} = 0.5h_0$ , and the corresponding  $\lambda_{\text{critical}}$ .
- (iii) **if** ( $h_{\max} \cong h_0$ ),  
     **then** (set  $h = h_0$ ,  $\lambda = \lambda_{\min}$ )  
     **else if** ( $h_0 > h_{\max} \cong h_{\text{critical}}$ )  
         **then** (set  $h = h_{\text{critical}}$  and  $\lambda = \lambda_{\text{critical}}$ )  
         **else if** ( $h_{\text{critical}} > h_{\max}$ )  
             **then** (set  $h = h_{\max}$  and obtain  $\lambda$  by solving

$$(6.1) \quad h_{\max}^2 = \sum_{j=1}^n \frac{(\mathbf{u}_j^T \mathbf{f}) s_j^2}{(s_j^2 + \lambda^2)^2}.$$

This algorithm is essentially that used by Holt and Fletcher [11] using the ideas of Fletcher [4]. The reader is referred to [11, §5] for a discussion of its motivation.

Using the working trust region parameter  $h$ , we compute an update  $\delta \mathbf{x}$  and  $z(\mathbf{x} + \delta \mathbf{x})$ .

The strategy for modifying the trust region depends on the ratio,  $\psi$ , of actual to predicted decrease in the objective function. Following the ideas of Fletcher [4] and Moré [15], we aim to keep this ratio at an acceptable level.  $\psi$  is compared with two constants  $\pi_1$  and  $\pi_2$  ( $0 < \pi_1 < \pi_2 < 1$ ). If  $\psi < \pi_1$ , insufficient predicted decrease has occurred and we decrease  $h$  by fitting a quadratic in  $\mu$  to  $z(\mathbf{x} + \mu \delta \mathbf{x})$  using the values of  $z(\mathbf{x})$ ,  $z(\mathbf{x} + \delta \mathbf{x})$  and  $\delta \mathbf{x}^T \nabla z(\mathbf{x})$  as interpolation data. If  $\mu^*$  minimizes the quadratic, we decrease  $h$  by multiplying it by  $\mu^*$ , unless  $\mu^*$  is not in  $(1/10, 1/2)$  in which case we first set  $\mu^*$  to the nearest endpoint. This is identical to the procedure of Moré [15]. An inner iteration is then performed with the decreased  $h$ , and the procedure repeated if necessary until a sufficient decrease is obtained. At the completion of the iteration, following Holt and Fletcher [11], the global trust region parameter  $h_{\max}$  is updated as follows.

**if** ( $h < \min(h_{\text{critical}}, h_{\text{max}})$ , (; an inner iteration occurred))  
**then** (set  $h_{\text{max}} = h$  (; a decrease))  
**else if** ( $\psi > \pi_2$  and  $\alpha = 1$ )  
**then** (set  $h_{\text{max}} + h$  (; an increase))  
**else** (leave  $h_{\text{max}}$  unchanged).

The method of computing the Levenberg–Marquardt parameter  $\lambda$  corresponding to the trust region  $h$  in (6.1) is essentially that of Hebden [8] with the safeguarding scheme of Moré [15]. We define the function

$$(6.2) \quad \phi(\lambda^2) = \sum_{j=1}^n \left[ \frac{(\mathbf{u}_j^T \mathbf{f})^2 s_j^2}{(s_j^2 + \lambda^2)^2} \right],$$

and aim to solve the equation

$$\phi(\lambda^2) = h_{\text{max}}^2$$

for  $\lambda^2$  iteratively. We make the rational approximation

$$\phi(\lambda^2) \approx \bar{\phi}(\lambda^2) = \frac{p}{(q + \lambda^2)^2}$$

where  $p$  and  $q$  are determined by enforcing the conditions  $\phi = \bar{\phi}$  and  $\phi' = \bar{\phi}'$  at the current iterate  $\lambda_k$ , say. We then find the next iterate by solving the system

$$\bar{\phi}(\lambda^2) = h_{\text{max}}^2$$

for  $\lambda$ .

Algorithm (5.5) of [15] is used to safeguard the iterates generated, and achieve quadratic convergence to  $\lambda$  which solves (6.1).

### 7. Convergence of the algorithm.

**7.1. Use of Lagrange multipliers.** The tests described in §5 are not the only tests used to determine whether or not constraint relaxation will occur. When the iterates appear to be converging, a Lagrange multiplier test is preferred, to satisfy the conditions of our convergence result. First-order Lagrange multiplier estimates can be obtained by solving the linear least squares problem

$$\min_{\lambda} \|C\lambda - J^T \mathbf{f}\|$$

where  $C$ , as defined earlier, is the active constraint matrix. We choose  $\lambda_q$  to minimize  $\lambda_i$ ,  $i = 1, \dots, t$ . If  $\lambda_q < 0$ , we relax the corresponding active constraint. If  $\lambda_q \geq 0$ , we continue iterating on the current manifold.

**7.2. Avoidance of zigzagging.** Zigzagging is a phenomenon which can adversely affect the performance of active set methods such as the one being described here. It occurs when the iterates do not eventually stay on the one active constraint manifold, but instead oscillate between different manifolds. This tends to considerably slow down, or even prevent, convergence.

Zigzagging will most likely occur if we perform some sort of relaxation test at each iteration. We aim to find a rule which will prevent zigzagging and yet allow the active constraint manifold to be changed when it appears likely that it is not the one on which the solution lies.

Fletcher [5] proposes a rule based on a second-order estimate of function reduction at each stage. Following him, we define this estimate as

$$\Delta^{(k)} = \mathbf{f}^T(JQ_2^T)(Q_2J^TJQ_2^T + \lambda^2I)^{-1}(Q_2J^T)\mathbf{f}$$

where  $\mathbf{f}$  and  $J$  are evaluated at  $\mathbf{x}^{(k)}$ . Using the fact that

$$\delta\mathbf{x}^{(k)} = -Q_2^T(Q_2J^TJQ_2^T + \lambda^2I)^{-1}Q_2J^T\mathbf{f}$$

we obtain

$$(7.1) \quad \Delta^{(k)} = \delta\mathbf{x}^{(k)T}(J^TJ + \lambda^2I)\delta\mathbf{x}^{(k)}.$$

Also we define a sequence of integers  $\{l(k)\}$  such that  $l(k)$  is the greatest iteration index prior to  $k$  on which a constraint is relaxed. We only consider constraint relaxation at iteration  $k$  if

$$(7.2) \quad \Delta^{(k)} \leq z^{(k)} - z^{(l(k))}.$$

This is the anti-zigzagging test referred to in §2. The idea is that if zigzagging occurs, the right-hand side of (7.2) will go to zero, and hence  $\Delta^{(k)} \rightarrow 0$ . This suggests that convergence will occur for the subsequence of points on the same manifold. Fletcher [5] has a convergence result [5, Thm. 11.3.1] for certain algorithms which use this rule. We will show in the next subsection that this result can be applied to the second of our methods.

**7.3. Convergence to a Kuhn–Tucker point.** We present in this section a convergence result for the second algorithm, in which  $\delta\mathbf{x}$  at each iteration is the exact minimizer of  $\|J\delta\mathbf{x} + \mathbf{f}\|^2$  on a certain active constraint manifold (see §5.2).

As mentioned in 7.1 and shown in the informal code in §2, we use Lagrange multiplier relaxation tests in place of the tests described in §5 when “crude” convergence has occurred on the current manifold. We use criteria similar to those of Dennis, Gay and Welsch [3] – our “crude convergence flag” is set if any of the following three conditions hold:

- (a)  $\frac{\max|\delta x_i|}{\max(|x_i + \delta x_i| + |x_i|)} < \varepsilon_x$  (relative  $x$ -convergence),
- (b)  $z(\mathbf{x}) < \varepsilon_A$  (absolute function convergence),
- (c)  $\frac{z(\mathbf{x}) - z(\mathbf{x} + \delta\mathbf{x})}{z(\mathbf{x})} < \varepsilon_R$  (relative function convergence),

where  $\varepsilon_x$ ,  $\varepsilon_A$  and  $\varepsilon_R$  are small positive tolerances. Since  $\{z^{(k)}\}$  is a decreasing sequence, bounded below by 0, it can be seen that there exists an integer  $K$  such that one of the conditions (b) or (c) will hold for all  $k > K$ . Hence after the  $K$ th iteration, the crude convergence flag will always be set at step (ii) of the informal code of §2, and so the Lagrange multiplier test will always be used after this stage. In proving convergence, then, we can ignore the first branch of the conditional statement in step (ii) of the code, since the “predictive” relaxation tests are performed only finitely often.

Before applying Fletcher’s result to our second algorithm we need to show that the sequence of iterates cannot have multiple accumulation points. To do this, we need the “sufficient decrease” condition in step (iv) of Fig.1, which is

$$(7.3) \quad \psi = \frac{z^{(k)} - z^{(k+1)}}{z^{(k)} - (\|\mathbf{f} + J\delta\mathbf{x}^{(k)}\|^2 + \lambda^2\|\delta\mathbf{x}^{(k)}\|^2)} \geq \pi_1$$

(where  $\mathbf{f}$  and  $J$  are evaluated at  $\mathbf{x}^{(k)}$ ). Expanding the denominator and dropping the superscript on  $\delta\mathbf{x}^{(k)}$ , we obtain



$$z^{(k)} - (\|\mathbf{f} + J\delta\mathbf{x}\|^2 + \lambda^2\|\delta\mathbf{x}\|^2) = -2\mathbf{f}^T J\delta\mathbf{x} - \delta\mathbf{x}^T (J^T J + \lambda^2 I)\delta\mathbf{x}.$$

But

$$\begin{aligned} -\mathbf{f}^T J\delta\mathbf{x} &= \mathbf{f}^T J Q_2^T (Q_2 J^T J Q_2^T + \lambda^2 I)^{-1} Q_2 J^T \mathbf{f} \\ &= \delta\mathbf{x}^T Q_2^T (Q_2 J^T J Q_2^T + \lambda^2 I) Q_2 \delta\mathbf{x} \\ &= \delta\mathbf{x}^T (Q_2^T Q_2) (J^T J + \lambda^2 I) (Q_2^T Q_2) \delta\mathbf{x} \end{aligned}$$

since  $\delta\mathbf{x}$  is the exact solution of a problem of the form (3.4) or (4.1). For notational simplicity,  $Q_2$  here is either the  $Q_2$  of (3.4) or  $(q_i \mid Q_2^T)^T$  of (4.1) as appropriate. Since  $\delta\mathbf{x} = Q_2^T \mathbf{p}_2$  for some vector  $\mathbf{p}_2$ , we have

$$Q_2 \delta\mathbf{x} = \mathbf{p}_2$$

and so the denominator is equal to

$$\mathbf{p}_2^T Q_2 (J^T J + \lambda^2 I) Q_2^T \mathbf{p}_2 = \delta\mathbf{x}^T (J^T J + \lambda^2 I) \delta\mathbf{x} \geq \lambda_{\min}^2 \|\delta\mathbf{x}^{(k)}\|^2 \quad \text{since } \lambda \geq \lambda_{\min}.$$

Hence, from the sufficient decrease condition,

$$(7.4) \quad z^{(k)} - z^{(k+1)} \geq \pi_1 \lambda_{\min}^2 \|\delta\mathbf{x}^{(k)}\|^2 \Leftrightarrow \|\delta\mathbf{x}^{(k)}\|^2 \leq \frac{1}{\pi_1 \lambda_{\min}^2} (z^{(k)} - z^{(k+1)}).$$

Note that (7.4) still applies if  $\delta\mathbf{x}^{(k)}$  is scaled to retain feasibility.

We can derive a similar inequality from (7.3) involving the reduced gradient. We do this by making the observation that since the objective function is  $C^2$  in the region of interest, the 2-norm of  $J$  is bounded. Hence the eigenvalues of  $J^T J$  and  $Q_2 J^T J Q_2^T$  are bounded above. It can also be shown, using a Taylor series argument, that  $\psi \rightarrow 2$  as  $\lambda \rightarrow \infty$  in (7.3) and hence we can always choose a finite value of  $\lambda$  for which the sufficient decrease condition is satisfied. Since the denominator in (7.3) is equal to

$$\mathbf{f}^T J Q_2^T (Q_2 J^T J Q_2^T + \lambda^2 I)^{-1} Q_2 J^T \mathbf{f}$$

and the eigenvalues of  $(Q_2 J^T J Q_2^T + \lambda^2 I)$  are bounded above and below, there exists  $\sigma > 0$  such that

$$\mathbf{f}^T J Q_2^T (Q_2 J^T J Q_2^T + \lambda^2 I)^{-1} Q_2 J^T \mathbf{f} \geq \sigma \|Q_2 J^T \mathbf{f}\|^2.$$

Hence from (7.3),

$$(7.5) \quad z^{(k)} - z^{(k+1)} \geq \pi_1 \sigma \|Q_2 J^T \mathbf{f}\|^2$$

where  $Q_2$  refers to the manifold on which  $\delta\mathbf{x}^{(k)}$  lies. If  $\mathbf{x}^{(k)}$  and  $\mathbf{x}^{(k+1)}$  are both on the same active constraint manifold, then  $Q_2$  refers to the manifold of  $\mathbf{x}^{(k)}$ . If not, then we can use (7.1) and (7.2) to show that

$$(7.6) \quad z^{(k)} - z^{(l(k))} \geq \Delta^{(k)} \geq \pi_1 \sigma \|Q_2 J^T \mathbf{f}\|^2$$

where, again,  $Q_2$  refers to the manifold of  $\mathbf{x}^{(k)}$ . In either case, since  $\{z^{(k)}\}$  is bounded below and decreasing, we can see that the reduced gradient  $Q_2 J^T \mathbf{f}$  approaches zero for any subsequence of points which lie on the same active constraint manifold.

The following lemma shows that the sequence of iterates  $\{\mathbf{x}^{(k)}\}$  can have only one accumulation point.

LEMMA. *If we assume that  $G(\mathbf{x})$ , the Hessian of  $z(\mathbf{x})$ , satisfies  $\mathbf{u}^T G(\mathbf{x}) \mathbf{u} \geq a > 0$  for all  $\mathbf{u}$  with  $\|\mathbf{u}\|_2 = 1$  and  $\mathbf{x} \in \{\mathbf{x} \mid z(\mathbf{x}) \leq z(\mathbf{x}^{(1)}) \text{ and } \mathbf{x} \text{ feasible}\}$ , then the sequence of iterates generated by the second algorithm cannot have multiple accumulation points.*

*Proof.* Firstly, we prove that there cannot be an infinite number of accumulation points. If this were true, then there must be at least one active constraint manifold  $\mathcal{A}'$  which holds an infinity of such points, denoted by  $\{y_i\}_{i=1}^\infty$ . Since each  $y_i$  is the limit of a subsequence from  $\{x^{(k)}\}$ , we deduce from (7.5) and (7.6) that

$$Q_2 J^T f(y_i) = 0 \quad \text{for all } i,$$

where  $Q_2$  is appropriate to  $\mathcal{A}'$ . If we restrict our region of interest  $\{x | z(x) \leq z(x^{(1)}) \text{ and } x \text{ feasible}\}$  to  $\mathcal{A}'$ , we have a compact set, and so the sequence  $\{y_i\}$  must have at least one accumulation point, say  $y$ . Assume WLOG that  $y_i \rightarrow y$ . Then

$$\frac{(y_i - y)^T (J^T f(y_i) - J^T f(y))}{\|y_i - y\|^2} \rightarrow \frac{1}{2} \frac{(y_i - y)^T G(y) (y_i - y)}{\|y_i - y\|^2} \quad \text{as } i \rightarrow \infty.$$

But since we can find  $a_i$  such that

$$y_i - y = Q_2^T a_i,$$

we have

$$0 = \frac{a_i^T (Q_2 J^T f(y_i) - Q_2 J^T f(y))}{\|a_i\|^2}$$

and hence

$$\frac{1}{2} \frac{a_i^T Q_2 G(y) Q_2^T a_i}{\|a_i\|^2} \rightarrow 0 \quad \text{as } i \rightarrow \infty.$$

But this gives a contradiction since the eigenvalues of  $G$  are bounded away from zero.

Hence there can be only a finite number of accumulation points. If there are more than one, then we can find two accumulation points  $x'$  and  $x''$  and a subsequence  $\{x^{(k_i)}\}$  such that

$$x^{(k_i)} \rightarrow x' \quad \text{and} \quad x^{(k_{i+1})} \rightarrow x''.$$

Hence  $\|\delta x^{(k_i)}\| \rightarrow \|x' - x''\|$  which contradicts (7.4) as  $k_i \rightarrow \infty$ . Hence there is only one accumulation point.  $\square$

It follows from the compactness of our region of interest that this accumulation point  $x^*$  is in fact the limit of the sequence  $\{x^{(k)}\}$ . We now present our convergence result.

**THEOREM.** *If we assume that  $\lambda \geq \lambda_{\min}$  at each step, that the active constraint matrix  $C$  has full rank at all feasible  $x$ , and that the eigenvalues of the Hessian are bounded away from zero in the region.*

$$\{x | z(x) \leq z(x^{(1)}) \text{ and } x \text{ feasible}\},$$

*then our second method, which solves the augmented problem (4.1) as in §5.2, will converge to a Kuhn–Tucker point  $x^*$  from any feasible initial point. In addition, if strict complementarity holds at  $x^*$  (i.e. all Lagrange multipliers positive) then no zigzagging can occur and  $\mathcal{A}(x^{(k)}) = \mathcal{A}(x^*)$  for  $k$  sufficiently large.*

*Proof.* If  $A(x^{(k)})$  is constant for  $k$  sufficiently large, then  $Q_2$  will be constant. Hence from (7.5) the reduced gradient approaches zero. Since the active set does not change, all Lagrange multipliers must be non-negative and so  $x^*$  is a Kuhn–Tucker point.

Otherwise the sequence of integers  $\{l(k)\}$  in (7.2) is unbounded. Because  $\{z^{(k)}\}$  is bounded below and decreasing,  $z^{(k)} - z^{(l(k))} \rightarrow 0$  and hence  $\Delta^{(k)} \rightarrow 0$  on the subsequence of iterates for which (7.2) holds. We have shown in the lemma that  $x^{(k)} \rightarrow x^*$ , so the remainder of the proof follows that of Fletcher [5, p.117].  $\square$

*Note.* For the first of our algorithms (which solves (4.1) approximately, as described in §5.1), convergence to a Kuhn–Tucker point follows as in the above theorem in the case where  $\mathcal{A}(x^{(k)})$  is constant for  $k$  sufficiently large. However the argument used to show that  $\{x^{(k)}\}$  does not have multiple accumulation points cannot be applied. Nevertheless, the algorithm performs well in practice, and it would be of interest if a similar theorem to the above could be proved for it.

**7.4. Termination details.** We only test for termination if the fine convergence flag has been set in step (ii). This flag can be deactivated if the set of active constraints is subsequently changed. Termination can take two forms:

(a) *Fine convergence.* Following Dennis, Gay and Welsch, this can only occur when the condition

$$(7.7) \quad z(\mathbf{x}) - z(\mathbf{x} + \delta\mathbf{x}) \leq 2(z(\mathbf{x}) - P)$$

is satisfied, where  $P$  is the predicted sum of squares at  $\mathbf{x} + \delta\mathbf{x}$ . If (7.7) is not satisfied, we regard our model as being unsatisfactory at  $\mathbf{x}$ . In addition, one of the conditions (a), (b) or (c) above must hold, with the right-hand sides scaled by a “fine convergence factor”, denoted by  $C_F$ .

(b) *False convergence.* This occurs when the iterates appear to be converging, but (7.7) does not hold. The criterion is

$$\frac{\max|\delta x_i|}{\max(|x_i + \delta x_i| + |x_i|)} < \varepsilon_F.$$

**7.5. Numerical parameters.** All computations have been carried out in single precision on the DEC-KL-10 for which machine epsilon is  $0.745 \times 10^{-8}$ , except for the SVD update routine, which is programmed in double precision. In the SVD of  $JQ_2^T$ , singular values which differ by less than  $10^{-6}$  are taken to be equal; in particular, singular values less than  $10^{-6}$  are taken to be zero.

The machine-independent numerical parameters are

- (i) the bounds on the magnitude of  $\rho_i^*$  in the first algorithm, which are set to  $\varepsilon_1 = 10^{-4} \cdot (1 + \|\mathbf{x}\|)$  and  $\varepsilon_2 = h/2$ ,
- (ii) the parameters  $\pi_1$  and  $\pi_2$  in damping control which are set to  $\pi_1 = .1$  and  $\pi_2 = .75$ ,
- (iii) the fine convergence factor  $C_F$ , which is set to  $10^{-3}$ .

The other numerical parameters to be set by the user are to some extent machine dependent. In the current implementation they are set to  $\varepsilon_X = 10^{-6}$ ,  $\varepsilon_A = 10^{-8}$ ,  $\varepsilon_R = 10^{-6}$ ,  $\varepsilon_F = 10^{-5}$  and  $\lambda_{\min} = 10^{-6}$ .  $\lambda_{\min}$  should be set no greater than the smallest nonzero singular value able to be returned by the SVD routine.

**8. Numerical examples.** The two algorithms were tried first on a set of small test problems which were modified from examples in Hock and Schittkowski [10]. They had 2 to 5 variables, 2 to 7  $f_i$ 's in the sum of squares, and up to 8 constraints. There was a mixture of zero-residual and nonzero residual problems. Both algorithms gave the correct solutions for all 16 examples. A total of 100 iterations and 5.99 seconds of DEC-10 CPU time (over all the examples) was required by the first algorithm, as against 97 iterations and 7.16 seconds for the second algorithm. As expected, the first algorithm gave decidedly better performance on these small examples.

A larger example due to Osborne [17] was also tested. This is an exponential data-fitting problem with  $n = 11$  and  $m = 65$ , with

$$f_i(x) = y_i - x_1 e^{-x_5 t_i} - x_2 e^{-x_6(t_i - x_9)^2} - x_3 e^{-x_7(t_i - x_{10})^2} - x_4 e^{-x_8(t_i - x_{11})^2}$$

where the values of  $y_i$  and  $t_i$  are given in [17]. Two constraints were applied, being modifications of the equality constraints in Kaufman and Pereyra [12]:

$$\begin{aligned} x_1 + 2x_2 + 3x_3 + 4x_4 &\cong 6.270063, \\ x_1 + x_3 &\cong 1.741584. \end{aligned}$$

These were chosen to be slightly inactive at the solution, and hence were a good test of the power of the methods to leave constraints.

Two test runs were produced. On the first, the starting vector was  $(1.3, .65, .65, .7, .6, 3., 5., 7., 2., 4.5, 4.5)^T$ . On the second, the same vector applied except that  $x_{11}$  was initially 5.5 and not 4.5. Both algorithms converged to the correct solution; however, the first had to resort to Lagrange multiplier estimates to relax one of the constraints. A comparison of iterations and running times is shown in Table 1.

TABLE 1  
Iterations/CPU time (secs) for two test runs of the Osborne problem.

Algorithm \ Test	I	II
1	16/9.89	10/5.75
2	12/8.71	8/4.91

TABLE 2  
Test run of first algorithm.

Iteration	$z(\mathbf{x})$	Active constraints	$h_{\max}$	$h_0$
1	5.262 (4 inner iterations)	1	.4098	480.74
2	4.885	1,2	.4098	22.82
3	2.759	1,2	.8196	14.86
4	1.008	1,2	1.639	2.532
5	.4026	1,2	3.278	7.087
6	.2927	1,2	5.612	4.668
7	.1121 (1 inner iteration)	1	7.147	3.070
8	.7441 (-1) (1 inner iteration)	1	7.147	2.183
9	.5767 (-1)	1	7.147	1.312
10	.4154 (-1)	1	7.383	.2358
11	.4146 (-1)	1	7.405	.2225 (-1)
12	.4101 (-1) (constraint 1 relaxed after examination of Lagrange multipliers)	-	7.405	.1390 (-2)
13	.4014 (-1)	-	7.514	.4843
14	.4014 (-1)	-	7.526	.1091
15	.4014 (-1)	-	7.526	.1267 (-1)
16	.401377 (-1)	-	7.526	.1143 (-2)

TABLE 3  
*Test run of second algorithm.*

Iteration	$z(\mathbf{x})$	Active constraints	$h_{\max}$	$h_0$
1	5.262 (4 inner iterations)	1	.4098	480.74
2	4.885	1,2	.4098	22.82
3	2.759	1,2	.8196	14.86
4	1.008	1,2	1.639	2.532
5	.4026	1,2	3.278	7.067
6	.2227	1	3.278	4.668
7	.1243 (1 inner iteration)	1	3.278	2.852
8	.1082 (1 inner iteration)	–	3.278	2.038
9	.4241 (–1)	–	4.062	.7836
10	.4014 (–1)	–	4.162	.1003
11	.4014 (–1)	–	4.168	.5710 (–2)
12	.401377 (–1)	–	4.168	.1424 (–2)

It is seen that the savings in iteration count for the second algorithm leads to a significant reduction in running time. The greater power of the second algorithm to leave constraints is further illustrated by Tables 2 and 3, which compare performance on the first test run.

**9. Conclusions.** Two practical algorithms have been presented for the important, but little-treated problem of nonlinear least squares with general linear constraints. The methods use a Gauss–Newton–Marquardt approach with a dynamically-determined region of trust for the iterations. Numerical examples have been presented. Unlike the unconstrained problem, standard batteries of test examples do not yet exist; hence modifications to the unconstrained test examples were necessary to thoroughly exercise the code. A suggestion for future work is to establish a comprehensive set of test problems. Work is currently underway to produce an algorithm for the important special case where the Jacobian is large and sparse. The present codes are appropriate for small- to medium-sized problems, where the computation of the singular value decomposition of the reduced Jacobian is practical. Copies of the code are available from the authors.

**Acknowledgments.** One of us (S.J.W.) wishes to acknowledge the support of an Australian Postgraduate Research Award. We also wish to thank the referees of an earlier version of the paper, whose suggestions and criticism were appropriate and appreciated.

#### REFERENCES

- [1] J. R. BUNCH AND C. P. NIELSEN, *Updating the singular value decomposition*, Numer. Math., 31 (1978), pp. 111–129.
- [2] J. E. DENNIS, *Nonlinear least squares and equations*, The State of the Art in Numerical Analysis, D. Jacobs, ed., Academic Press, New York, 1977, pp. 269–312.
- [3] J. E. DENNIS, D. M. GAY, AND R. E. WELSCH, *An adaptive non-linear least squares algorithm*, ACM, Trans. Math. Software, 7 (1981) pp. 348–368.
- [4] R. Fletcher, *A modified Marquardt subroutine for non-linear least squares*, Harwell Report AERE-R.6799, 1972.

- [5] R. FLETCHER, *Practical Methods of Optimization, Volume 2: Constrained Optimization*, John Wiley, New York, 1981.
- [6] P. E. GILL AND W. MURRAY, *Algorithms for the solution of the nonlinear least squares problem*, SIAM J. Numer. Anal., 15 (1978), pp. 977–992.
- [7] P. E. GILL, W. MURRAY AND M. H. WRIGHT, *Practical Optimization*, Academic Press, New York, 1981.
- [8] M. D. HEBDEN, *An algorithm for minimization using exact second derivatives*, Harwell Report AERE TP515, 1973.
- [9] K. L. HIEBERT, *An evaluation of mathematical software that solves nonlinear least squares problems*, ACM Trans. Math. Software, 7 (1981) pp. 1–16.
- [10] W. HOCK AND K. SCHITTKOWSKI, *Test examples for nonlinear programming codes*, Lecture Notes in Economics and Mathematical Systems 187, Springer-Verlag, New York, 1981.
- [11] J. N. HOLT AND R. FLETCHER, *An algorithm for constrained non-linear least squares*, J. Inst. Maths Applics, 23 (1979), pp. 449–463.
- [12] L. KAUFMAN AND V. PEREYRA, *A method for separable, non-linear least squares problems with separable nonlinear equality constraints*, SIAM J. Numer. Anal., 15 (1978), p. 12–20.
- [13] C. L. LAWSON AND R. J. HANSON, *Solving Least Squares Problems*, Prentice-Hall, Englewood Cliffs, NJ, 1974.
- [14] P. LINDSTRÖM, *A working algorithm based on the Gauss–Newton method for non-linear least squares problems with non-linear constraints*, Report UMINF-79-80, Institute of Information Processing, University of Umea, 1980.
- [15] J. J. MORÉ, *The Levenberg–Marquardt algorithm: Implementation and Theory*, Numerical Analysis, Lecture Notes in Mathematics 630, G. Watson, ed., Springer-Verlag, New York, 1978, pp. 105–116.
- [16] L. NAZARETH, *Some recent approaches to solving large residual nonlinear least squares problems*, SIAM Rev. 22 (1980), pp. 1–11.
- [17] M. R. OSBORNE, *Some aspects of non-linear least squares calculations*, Numerical Methods for Non-Linear Optimisation, F. Lootsma, ed., Academic Press, New York, 1972.

## SOLVING ELLIPTIC DIFFERENCE EQUATIONS ON A LINEAR ARRAY OF PROCESSORS\*

Y. SAAD†, A. SAMEH‡ AND P. SAYLOR‡

**Abstract.** In this paper we consider the organization of three iterative methods for solving self-adjoint elliptic difference equations on a set of linearly connected processors. These algorithms are the cyclic Chebyshev semi-iterative scheme, a preconditioned conjugate gradient method, and a generalization of the Chebyshev method. We also compare their performance on this multiprocessor as a function of the cost of interprocessor communication.

**Key words.** elliptic difference equations, multiprocessors, conjugate gradient method, Chebyshev methods, parallel algorithms

**1. Introduction.** Numerical methods for solving elliptic partial differential equations have been among the most important applications made possible by the digital computer. The literature in this area is extensive and is still evolving, see for example [Schu81], in particular the articles by Rice and Young regarding the two software packages ELLPACK and ITPACK respectively. With the advent of array and vector computers such as the ILLIAC IV and the Cray-1, respectively, some studies have been performed to determine the speedup that may be achieved by direct and iterative numerical methods for solving elliptic difference equations on these computers ([Miur71], [Eric72], [BuGH77], and [GePV78]). In this paper we concern ourselves only with iterative methods for solving self-adjoint elliptic difference equations on a multiprocessor [Same81]. Specifically, we consider the cyclic Chebyshev semi-iterative scheme, the conjugate gradient method, and a generalization of the Chebyshev method which we call the block-Stiefel iteration.

We assume that our hypothetical multiprocessor consists of a number of linearly connected processors, where each processor can simultaneously transmit a previously computed floating-point number to either of its immediate neighbors, and receive another, while performing an arithmetic operation. We further assume that an arithmetic operation consumes one time step, while the cost of transmitting and receiving a floating-point number is  $\psi \geq 1$  time steps.

In order to illustrate our above assumption regarding overlapping the computation and communication (i.e. the fact that the linear array of processors is pipelined), we present the following example. See Fig. 1. Let each processor  $j$ ,  $1 \leq j \leq p-1$ , be responsible for computing the scalar-vector multiplication  $\mathbf{v}_j = \alpha_j \mathbf{u}_j$  and transmitting it to processor  $j+1$ . After processor  $j$  performs the multiplication  $v_1 = \alpha_j u_1$ , where  $u_i(v_i)$ ,  $1 \leq i \leq n$ , is the  $i$ th element of  $\mathbf{u}_j(\mathbf{v}_j)$ , it issues an instruction for transmitting  $v_1$  to processor  $j+1$ , and proceeds to compute  $v_2 = \alpha_j u_2$ . The quantity  $v_1$ , therefore, reaches processor  $j+1$   $\psi$  time steps after it has become available in processor  $j$ , and from now on  $v_i$ ,  $2 \leq i \leq n$ , becomes available in the local memory of processor  $j+1$  following each subsequent time step. Hence, the cost of computing  $\mathbf{v}_j = \alpha_j \mathbf{u}_j$  in each processor  $j$ , and transmitting it to processor  $j+1$  is  $(n + \psi + 1)$  time steps.

The main objective of this paper is twofold: we demonstrate the organization of the cyclic Chebyshev semi-iterative and the conjugate gradient schemes on a pipelined linear

---

\*Received by the editors November 9, 1982, and in revised form May 1, 1984. An early version of this paper appeared in the proceedings of CONPAR81, Springer-Verlag. This work has been partially supported by the National Science Foundation under grants MCS 79-18394 and MCS 81-17010.

†Department of Computer Science, Yale University, New Haven, Connecticut 06520.

‡Department of Computer Science, University of Illinois at Urbana-Champaign, Urbana, Illinois 61801.

.....	Proc. $j$	Proc. $j + 1$	.....
	$v_1$		
	$v_2$		
	$v_3$		
	$v_4$	$v_1$	
	$v_5$	$v_2$	
	$\vdots$	$\vdots$	
	$v_n$	$v_{n-3}$	
		$\vdots$	
		$v_n$	

FIG. 1. Example  $\psi = 2$  time steps.

array of processors, and introduce the block-Stiefel iteration. We show that the latter algorithm, with optimal parameters, can be superior to the conjugate gradient method on our hypothetical multiprocessor. This occurs when the (preconditioned) finite-difference operator has a certain eigenvalue distribution, or when  $\psi$  is much larger than 1.

There are several reasons to consider a linear array of processors for handling the above problem. Such arrays are relatively easy to construct and expand. They deliver high performance (speedup is roughly equal to the number of processors) by using both multiprocessing and pipelining. Furthermore, such a linear array of processors can be easily configured by connecting an equal number of nonfaulty cells on a wafer, e.g. see [AuCa78], [GaSW82], [FuVa82], and [LeLe83]. In this case, the wafer is attached to a host computer as an inexpensive peripheral device.

**2. A model problem.** In order to illustrate a suitable organization of each of the above three algorithms on our multiprocessor, we consider the following second order, self-adjoint elliptic problem on the unit square

$$(2.1) \quad -\frac{\partial}{\partial x} \left[ a(x, y) \frac{\partial u}{\partial x} \right] - \frac{\partial}{\partial y} \left[ b(x, y) \frac{\partial u}{\partial y} \right] + c(x, y)u = f(x, y)$$

for  $0 < x, y < 1$ , with Dirichlet boundary conditions. We assume that  $a(x, y)$  and  $b(x, y)$  are positive, and  $c(x, y)$  is nonnegative.

Superimposing a square grid over the unit square with a mesh size  $h = 1/(n + 1)$ , where  $n$  is an even integer, and using the five-point difference scheme with line red-black ordering [Youn71], we obtain a positive-definite linear system of order  $n^2$

$$(2.2) \quad \mathbf{C}u = \mathbf{f}$$

whose solution  $u$  yields an approximation of  $u(x, y)$  in the unit square. Further, the matrix  $\mathbf{C}$  is of the form

$$\mathbf{C} = \begin{bmatrix} \mathbf{T}_R & \mathbf{F} \\ \mathbf{F}^T & \mathbf{T}_B \end{bmatrix}$$





Here

$$(2.6) \quad \hat{\mathbf{L}}_R = \text{diag}\{\hat{\mathbf{L}}_{2i-1} = \mathbf{D}_{2i-1}^{-1/2} \mathbf{L}_{2i-1} \mathbf{D}_{2i-1}^{1/2}\},$$

$$\hat{\mathbf{L}}_B = \text{diag}\{\hat{\mathbf{L}}_{2i} = \mathbf{D}_{2i}^{-1/2} \mathbf{L}_{2i} \mathbf{D}_{2i}^{1/2}\}, \quad 1 \leq i \leq \frac{n}{2}$$

are unit lower bidiagonal matrices, and

$$(2.7) \quad \hat{\mathbf{F}} = \mathbf{D}_R^{-1/2} \mathbf{F} \mathbf{D}_B^{-1/2} = \begin{bmatrix} \hat{\mathbf{B}}_1 & & & & & \\ \hat{\mathbf{B}}_2 & \hat{\mathbf{B}}_3 & & & & \\ & \ddots & \ddots & & & \\ & & & \ddots & & \\ & & & & \hat{\mathbf{B}}_{n-2} & \hat{\mathbf{B}}_{n-1} \end{bmatrix}$$

in which

$$\hat{\mathbf{B}}_{2i-1} = \mathbf{D}_{2i-1}^{-1/2} \mathbf{B}_{2i-1} \mathbf{D}_{2i}^{-1/2}, \quad 1 \leq i \leq \frac{n}{2}$$

and

$$\hat{\mathbf{B}}_{2i} = \mathbf{D}_{2i+1}^{-1/2} \mathbf{B}_{2i} \mathbf{D}_{2i}^{-1/2}, \quad 1 < i \leq \frac{n}{2} - 1$$

remain diagonal matrices. Such a transformation of (2.3) is referred to as the Cuthill–Varga normalization technique [CuVa59], [HaYo81].

It is well known [Varg 62] that the block-Jacobi iterative scheme

$$(2.8) \quad \begin{bmatrix} \mathbf{w}_{k+1}^{(R)} \\ \mathbf{w}_{k+1}^{(B)} \end{bmatrix} = \begin{bmatrix} \mathbf{0} & -\mathbf{G} \\ -\mathbf{G}^T & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{w}_k^{(R)} \\ \mathbf{w}_k^{(B)} \end{bmatrix} + \begin{bmatrix} \mathbf{g}_R \\ \mathbf{g}_B \end{bmatrix}$$

converges to a solution of (2.4), barring roundoff error, for arbitrary  $\mathbf{w}_0^{(R)}$  and  $\mathbf{w}_0^{(B)}$ . In other words the spectral radius of

$$(2.9) \quad \mathbf{H} = \begin{bmatrix} \mathbf{0} & -\mathbf{G} \\ -\mathbf{G}^T & \mathbf{0} \end{bmatrix}$$

is less than 1.

**3. Preprocessing stage.** Throughout this paper we assume that the number of processors is  $p = n/2$ . The algorithms, however, may be reorganized in an obvious way if  $p < n/2$ . Using  $a(x, y)$ ,  $b(x, y)$ ,  $c(x, y)$ , and  $f(x, y)$  each processor  $j$  generates  $\mathbf{A}_{2j-1}$ ,  $\mathbf{A}_{2j}$ ; the corresponding portions of the right-hand side  $\mathbf{f}_{2j-1}$ ,  $\mathbf{f}_{2j}$ ; and  $\mathbf{B}_{2j-1}$ ,  $\mathbf{B}_{2j}$ . The cost of the preprocessing stage, i.e. the cost of obtaining  $\hat{\mathbf{F}}$ ,  $\hat{\mathbf{L}}_R$ ,  $\hat{\mathbf{L}}_B$ ,  $\mathbf{g}_R$ , and  $\mathbf{g}_B$ , may be outlined as follows. Each processor  $j$ ,  $1 \leq j \leq n/2$ , performs the following:

1. Obtains the factorization  $\mathbf{A}_{2j-1} = \mathbf{L}_{2j-1} \mathbf{D}_{2j-1} \mathbf{L}_{2j-1}^T$ , and  $\mathbf{A}_{2j} = \mathbf{L}_{2j} \mathbf{D}_{2j} \mathbf{L}_{2j}^T$ , at the cost of  $8(n - 1)$  time steps. (The contents of the local memories of the  $n/2$  processors, for  $n = 12$ , are shown in Fig. 2a.)
2. Forms the diagonal matrices  $\mathbf{D}_{2j-1}^{-1/2}$  and  $\mathbf{D}_{2j}^{-1/2}$ , and transmits  $\mathbf{D}_{2j-1}^{-1/2}$  to processor  $j - 1$ , at the cost of  $2n$  square roots, and  $(2n + \psi + 1)$  time steps.
3. Obtains  $\hat{\mathbf{L}}_{2j-1}$ , and  $\hat{\mathbf{L}}_{2j}$ , see (2.6), at the cost of  $4(n - 1)$  time steps.
4. Computes  $\hat{\mathbf{B}}_{2j-1} = \mathbf{D}_{2j-1}^{-1/2} \mathbf{B}_{2j-1} \mathbf{D}_{2j-1}^{-1/2}$  and  $\hat{\mathbf{B}}_{2j} = \mathbf{D}_{2j+1}^{-1/2} \mathbf{B}_{2j} \mathbf{D}_{2j}^{-1/2}$ , and transmits  $\hat{\mathbf{B}}_{2j}$  to processor  $j + 1$ , at a cost of  $(4n + \psi + 1)$  time steps. (The reason that we would like to store  $\hat{\mathbf{B}}_{2j-2}$ ,  $\hat{\mathbf{B}}_{2j-1}$ , and  $\hat{\mathbf{B}}_{2j}$  in processor  $j$  will become apparent later as we discuss the multiplication of  $\hat{\mathbf{F}}$  by a vector.)

Proc. 1	Proc. 2	Proc. 3	Proc. 4	Proc. 5	Proc. 6
$D_1, L_1$ $D_2, L_2$	$D_3, L_3$ $D_4, L_4$	$D_5, L_5$ $D_6, L_6$	$D_7, L_7$ $D_8, L_8$	$D_9, L_9$ $D_{10}, L_{10}$	$D_{11}, L_{11}$ $D_{12}, L_{12}$
$B_1$ $B_2$	$B_3$ $B_4$	$B_5$ $B_6$	$B_7$ $B_8$	$B_9$ $B_{10}$	$B_{11}$
$f_1, f_2$	$f_3, f_4$	$f_5, f_6$	$f_7, f_8$	$f_9, f_{10}$	$f_{11}, f_{12}$

FIG. 2a.

5. Obtains the vectors  $g_{2i-1} = D_{2i-1}^{-1/2} \hat{f}_{2i-1}$  and  $g_{2i} = D_{2i}^{-1/2} \hat{f}_{2i}$ , where  $\hat{f}_{2i-1}$  and  $\hat{f}_{2i}$  are the solutions of the systems  $L_{2i-1} \hat{f}_{2i-1} = f_{2i-1}$  and  $L_{2i} \hat{f}_{2i} = f_{2i}$ , respectively, at the cost of  $(6n - 4)$  time steps.

Hence the total cost of the preprocessing stage is roughly  $T_0 = 24n + 2\psi$  time steps, and  $2n$  square roots. As we shall see later, this cost is comparable to the cost of only one iteration of any of the three iterative schemes discussed in this paper. Figure 2b shows the contents of the  $n/2$  processors after the preprocessing stage ( $n = 12$ ).

Proc. 1	Proc. 2	Proc. 3	Proc. 4	Proc. 5	Proc. 6
$D_1, \hat{L}_1$ $D_2, \hat{L}_2$	$D_3, \hat{L}_3$ $D_4, \hat{L}_4$	$D_5, \hat{L}_5$ $D_6, \hat{L}_6$	$D_7, \hat{L}_7$ $D_8, \hat{L}_8$	$D_9, \hat{L}_9$ $D_{10}, \hat{L}_{10}$	$D_{11}, \hat{L}_{11}$ $D_{12}, \hat{L}_{12}$
$\hat{B}_1$ $\hat{B}_2$	$\hat{B}_3$ $\hat{B}_4$ $\hat{B}_2$	$\hat{B}_5$ $\hat{B}_6$ $\hat{B}_4$	$\hat{B}_7$ $\hat{B}_8$ $\hat{B}_6$	$\hat{B}_9$ $\hat{B}_{10}$ $\hat{B}_8$	$\hat{B}_{11}$  $\hat{B}_{10}$
$g_1, g_2$	$g_3, g_4$	$g_5, g_6$	$g_7, g_8$	$g_9, g_{10}$	$g_{11}, g_{12}$

FIG. 2b.

**4. The cyclic Chebyshev semi-iterative method (CCSI).** Golub and Varga [GoVa 61] have developed a scheme for accelerating the convergence of any iterative method of the form (2.8). It can be stated as follows. Let  $w_0^{(R)}$  be an arbitrary initial approximation of  $w_R$ , and  $w_1^{(R)} = (g_B - G^T w_0^{(R)})$ . Now, for  $k = 1, 2, 3, \dots$ , we construct the iterates

(4.1a) 
$$w_{2k}^{(R)} = w_{2k-2}^{(R)} + \alpha_{2k} \Delta w_{2k-2}^{(R)}$$

and

(4.1b) 
$$w_{2k+1}^{(B)} = w_{2k-1}^{(B)} + \alpha_{2k+1} \Delta w_{2k-1}^{(B)},$$

where

(4.1c) 
$$\Delta w_{2k-2}^{(R)} = (g_R - G w_{2k-1}^{(B)}) - w_{2k-2}^{(R)}$$

and

(4.1d) 
$$\Delta w_{2k-1}^{(B)} = (g_B - G^T w_{2k}^{(R)}) - w_{2k-1}^{(B)}.$$

The acceleration parameters  $\alpha_j$  are given by

$$\alpha_2 = \frac{2}{2 - \rho^2}, \quad \alpha_{j+1} = \frac{1}{1 - \rho^2 \alpha_j / 4}, \quad j \geq 2,$$

where  $\rho$  is the spectral radius of  $\mathbf{H}$  in (2.9).

We would like to point out that by the  $k$ th iteration we mean obtaining  $\mathbf{w}_{2k}^{(R)}$  and  $\mathbf{w}_{2k+1}^{(B)}$  from  $\mathbf{w}_{2k-2}^{(R)}$  and  $\mathbf{w}_{2k-1}^{(B)}$ , respectively. Let

$$\mathbf{w}_{2k-2}^{(R)} = (\mathbf{y}_1^{(R)T}, \mathbf{y}_2^{(R)T}, \dots, \mathbf{y}_{n/2}^{(R)T})^T$$

and

$$\mathbf{w}_{2k-1}^{(B)} = (\mathbf{y}_1^{(B)T}, \mathbf{y}_2^{(B)T}, \dots, \mathbf{y}_{n/2}^{(B)T})^T.$$

If the spectral radius  $\rho$  is known beforehand, and  $\rho^2$  and  $\rho^2/4$  are contained in the local memory of each processor  $j$ ,  $1 \leq j \leq n/2$ , together with  $\mathbf{y}_j^{(R)}$  and  $\mathbf{y}_j^{(B)}$ , each CCSI iteration may be efficiently organized on our multiprocessor, in the sense that the resulting speedup over its sequential counterpart is almost  $n/2$ .

An iteration consists of two main tasks:

- (i) given vectors  $\mathbf{a} = (\mathbf{a}_1^T, \dots, \mathbf{a}_{n/2}^T)^T$ ,  $\mathbf{b} = (\mathbf{b}_1^T, \dots, \mathbf{b}_{n/2}^T)^T$ , and a scalar  $\alpha$ , with  $\mathbf{a}_j, \mathbf{b}_j \in \mathbb{R}^n$ , and  $\alpha$  contained in the local memory of processor  $j$ , obtain  $\mathbf{c} = \mathbf{a} + \alpha \mathbf{b}$ ; and
- (ii) given a vector  $\mathbf{a} = (\mathbf{a}_1^T, \dots, \mathbf{a}_{n/2}^T)^T$ , with  $\mathbf{a}_j \in \mathbb{R}^n$  contained in processor  $j$ , compute  $\mathbf{d} = \mathbf{G} \mathbf{a}$  or  $\mathbf{e} = \mathbf{G}^T \mathbf{a}$ .

Task (i) requires no interprocessor communications and is accomplished at the cost of  $2n$  time steps. Task (ii), however, does require interprocessor communications. The computation of  $\mathbf{d} = \mathbf{G} \mathbf{a} = \hat{\mathbf{L}}_R^{-1} \hat{\mathbf{F}} \hat{\mathbf{L}}_B^{-T} \mathbf{a}$  can be divided into three stages:

1.  $\hat{\mathbf{L}}_B^T \mathbf{b} = \mathbf{a}$ , consists of solving  $n/2$  independent unit lower bidiagonal systems  $\hat{\mathbf{L}}_{2j}^T \mathbf{b}_j = \mathbf{a}_j$ ,  $1 \leq j \leq n/2$ , one per processor (cost =  $2(n-1)$  steps).
2.  $\mathbf{c} = (\mathbf{c}_1^T, \dots, \mathbf{c}_{n/2}^T)^T = \hat{\mathbf{F}} \mathbf{b}$ , where  $\mathbf{c}_j = \hat{\mathbf{B}}_{2j-2} \mathbf{b}_{j-1} + \hat{\mathbf{B}}_{2j-1} \mathbf{b}_j$ . Here, each processor  $j$  computes  $\hat{\mathbf{B}}_{2j-1} \mathbf{b}_j$  (cost =  $n$  steps), computes  $\hat{\mathbf{B}}_{2j} \mathbf{b}_j$  and transmits it to processor  $j+1$  (cost =  $n + \psi + 1$  steps), and finally  $\mathbf{c}_j$  is computed as the sum of the previous two vectors (cost =  $n$  steps) for a total cost of  $3n + \psi + 1$  time steps.
3.  $\hat{\mathbf{L}}_R \mathbf{d} = \mathbf{c}$  is solved as in stage 1 in  $2(n-1)$  steps.

Consequently, task (ii) consumes  $(7n + \psi - 3)$  time steps.

As a result, in iteration  $k$ ,  $\mathbf{w}_{2k}^{(R)}$  may be obtained in  $11n + \psi - 3$  time steps, if we assume that the acceleration parameter  $\alpha_{2k}$  is available in all  $n/2$  processors. Vector  $\mathbf{w}_{2k+1}^{(B)}$  is then computed in another  $11n + \psi - 3$  steps plus 3 additional steps to compute  $\alpha_{2k+1}$ . Hence, the cost of one iteration of CCSI is roughly  $T(\text{CCSI}) = 22n + 2\psi$  time steps, a speedup of approximately  $n/2$  over the sequential scheme.

**5. The conjugate gradient method (CG).** We consider the conjugate gradient algorithm applied to the reduced system

$$(5.1a) \quad (\mathbf{I} - \mathbf{G}^T \mathbf{G}) \mathbf{w}_B = (\mathbf{g}_B - \mathbf{G}^T \mathbf{g}_R),$$

or

$$(5.1b) \quad \mathbf{K} \mathbf{v} = \mathbf{b}.$$

This will be referred to as the (RS-CG) scheme; see [HaLY80] and [HaYo 81]. When  $\mathbf{K}$  is left in the form  $(\mathbf{I} - \mathbf{G}^T \mathbf{G})$ , where  $\mathbf{G}$  is as given by the first of equations (2.5), this algorithm

requires the additional preprocessing task of computing  $\mathbf{b} = \mathbf{g}_b - \mathbf{G}^T \mathbf{g}_r$ . As indicated in tasks (i) and (ii) in §4,  $\mathbf{b}$  may be obtained in approximately  $T'_0 = 8n + \psi$  time steps; here we are adopting the storage scheme shown in Fig. 2a.

The CG algorithm applied to (5.1b) is given by:

(a) *Initial step*

$\mathbf{v}_0$ : arbitrary

$$\mathbf{r}_0 = \mathbf{b} - \mathbf{K} \mathbf{v}_0$$

$$\mathbf{p}_0 = \mathbf{r}_0, \rho_0 = \mathbf{r}_0^T \mathbf{r}_0.$$

(b) *For  $j = 0, 1, 2, \dots$ , obtain the following*

1.  $\mathbf{q}_j = \mathbf{K} \mathbf{p}_j$
2.  $\eta_j = \mathbf{p}_j^T \mathbf{q}_j$
3.  $\alpha_j = \rho_j / \eta_j$
- (5.2) 4.  $\mathbf{v}_{j+1} = \mathbf{v}_j + \alpha_j \mathbf{p}_j$
5.  $\mathbf{r}_{j+1} = \mathbf{r}_j - \alpha_j \mathbf{q}_j$
6.  $\rho_{j+1} = \mathbf{r}_{j+1}^T \mathbf{r}_{j+1}$
7.  $\beta_j = \rho_{j+1} / \rho_j$
8.  $\mathbf{p}_{j+1} = \mathbf{r}_{j+1} + \beta_j \mathbf{p}_j.$

In addition to fundamental tasks (i) and (ii), discussed in §4, we see that inner products are necessary for the iterations (5.2):

- (iii) Given two vectors  $\mathbf{a} = (\mathbf{a}_1^T, \dots, \mathbf{a}_{n/2}^T)^T$  and  $\mathbf{b} = (\mathbf{b}_1^T, \dots, \mathbf{b}_{n/2}^T)$ , with  $\mathbf{a}_j$  and  $\mathbf{b}_j \in \mathbb{R}^n$  contained in processor  $j$ , the inner product  $\mathbf{a}^T \mathbf{b}$  is computed and made available in all processors in  $[(2.5n - 2) + (n - 2)\psi]$  time steps.

This task is realized as follows. Compute in each processor  $j$ ,  $1 \leq j \leq n/2$ , the inner product  $\theta_j = \mathbf{a}_j^T \mathbf{b}_j$  (cost =  $2n - 1$  steps). Sequentially obtain the partial sum  $v_i = \theta_1 + \theta_2 + \dots + \theta_i$  in processor  $i$  by shifting to the right and adding until  $v_{n/2} = \mathbf{a}^T \mathbf{b}$  is contained in processor  $n/2$  (cost =  $(n/2 - 1)(\psi + 1)$  steps). Finally, shift to the left until each processor contains  $\mathbf{a}^T \mathbf{b}$  (cost =  $(n/2 - 1)\psi$ ).

Now, from the cost of each of the three fundamental tasks discussed earlier, it can be shown that each iteration (5.2) consumes approximately  $T(\text{CG}) \approx 26n + 2n\psi$  time steps. In the RS-CG scheme, however, the preprocessing cost  $T_0(\text{CG})$  is the sum of  $T_0$  (§3),  $T'_0$  (cost of forming the right-hand side of (5.1b)), and the cost of computing  $\mathbf{r}_0$ , i.e., roughly  $15n + 2\psi$ . Hence,  $T_0(\text{CG}) \approx 47n + 5\psi$  time steps. (The preprocessing cost for CCSI is slightly less, i.e.,  $T_0(\text{CCSI}) = T_0 + 7n + \psi = 29n + 3\psi$ .)

Note that, on our multiprocessor, the time needed by one CG iteration is approximately  $(1.18 + .09\psi)$  times more costly than one CCSI iteration. For large  $\psi$ , and an unfavorable (for CG) distribution of the eigenvalues of  $\mathbf{K}$ , CCSI with optimal parameters may be competitive with the RS-CG scheme.

**6. The block-Stiefel algorithm (BST).** In this section we consider a generalization of the Stiefel iteration applied to the reduced system (5.1b)

$$\mathbf{K} \mathbf{v} = \mathbf{b}.$$

Let  $\nu$  and  $\mu$  be estimates of the smallest and largest eigenvalues of  $K$ , i.e.

$$0 < \nu \leq \mu_1 \leq \mu_2 \leq \dots \leq \mu_m \leq \mu < 1.$$

The classical Steifel iteration [Stie58] may be given as follows:

(a) *Initial step*

$\mathbf{v}_0$ : arbitrary

$$\mathbf{r}_0 = \mathbf{b} - \mathbf{K}\mathbf{v}_0$$

$$\mathbf{v}_1 = \mathbf{v}_0 + \gamma^{-1}\mathbf{r}_0; \mathbf{r}_1 = \mathbf{b} - \mathbf{K}\mathbf{v}_1.$$

(b) For  $j = 1, 2, 3, \dots$  obtain the following:

$$(6.1) \quad \begin{aligned} 1. \quad & \Delta \mathbf{v}_j = \omega_j \mathbf{r}_j + (\gamma \omega_j - 1) \Delta \mathbf{v}_{j-1}, \\ 2. \quad & \mathbf{v}_{j+1} = \mathbf{v}_j + \Delta \mathbf{v}_j, \\ 3. \quad & \mathbf{r}_{j+1} = \mathbf{b} - \mathbf{K}\mathbf{v}_{j+1}. \end{aligned}$$

Here,  $\gamma = \beta/\alpha$ , in which

$$\alpha = \frac{2}{\mu - \nu}, \quad \beta = \frac{\mu + \nu}{\mu - \nu}$$

and

$$\omega_j = \left( \gamma - \frac{1}{4\alpha^2} \omega_{j-1} \right)^{-1}, \quad j \geq 1$$

with  $\omega_0 = 2/\gamma$ .

This iterative scheme produces residuals  $\mathbf{r}_j$  that satisfy the relation

$$(6.2) \quad \mathbf{r}_j = P_j(\mathbf{K})\mathbf{r}_0$$

where  $P_j(\lambda)$  is a polynomial of degree  $j$  given by

$$(6.3) \quad P_j(\lambda) = \frac{\tau_j(\beta - \alpha\lambda)}{\tau_j(\beta)}, \quad \nu \leq \lambda \leq \mu,$$

in which the Chebyshev polynomial  $\tau_j(\xi)$  is defined by

$$\tau_j(\xi) = \begin{cases} \cos(j \cos^{-1} \xi), & |\xi| \leq 1, \\ \cosh(j \cosh^{-1} \xi), & \xi \geq 1. \end{cases}$$

As a result

$$\frac{\|\mathbf{r}_j\|_2}{\|\mathbf{r}_0\|_2} \leq \tau_j^{-1}(\beta).$$

In the optimal case  $\nu = \mu_1$  and  $\mu = \mu_m$ .

Suppose now that  $\nu$  is an estimate of an interior eigenvalue  $\mu_{s+1}$ , where  $s$  is a small integer

$$0 < \mu_1 \leq \dots \leq \mu_s < \nu \leq \mu_{s+1} \leq \dots \leq \mu_m \leq \mu < 1.$$

Consequently, if  $\mathbf{r}_0 = \sum_{i=1}^m \eta_i \mathbf{z}_i$ , where  $\mathbf{z}_i$  is the eigenvector of  $\mathbf{K}$  corresponding to  $\mu_i$ , then  $\mathbf{r}_j$  can be expressed as

$$(6.4) \quad \mathbf{r}_j = \sum_{i=1}^s \eta_i P_j(\mu_i) \mathbf{z}_i + \sum_{i=s+1}^m \eta_i P_j(\mu_i) \mathbf{z}_i = \mathbf{r}'_j + \mathbf{r}''_j.$$

While  $\mathbf{r}_j''$  is damped out quickly as  $j$  increases, i.e., for  $\beta = (\mu + \nu)/(\mu - \nu)$

$$(6.5) \quad \frac{\|\mathbf{r}_j''\|_2}{\|\mathbf{r}_0''\|_2} \leq \tau_j^{-1}(\beta),$$

the term  $\mathbf{r}_j'$  is damped out at a much slower rate,

$$\frac{\|\mathbf{r}_j'\|_2}{\|\mathbf{r}_0'\|_2} \leq \frac{\tau_j(\beta - \alpha\mu_1)}{\tau_j(\beta)}.$$

The basic strategy of the block-Stiefel algorithm is to annihilate the contributions of the eigenvectors  $\mathbf{z}_1, \mathbf{z}_2, \dots, \mathbf{z}_s$  to the residuals  $\mathbf{r}_j$  so that eventually  $\|\mathbf{r}_j\|_2$  approaches zero as  $\zeta_j = 1/\tau_j[(\mu_m + \mu_{s+1})/(\mu_m - \mu_{s+1})]$  rather than  $\kappa_j = 1/\tau_j[(\mu_m + \mu_1)/(\mu_m - \mu_1)]$  as in the classical Stiefel iteration [Ruti 59]. Let  $\mathbf{Z} = [\mathbf{z}_1, \mathbf{z}_2, \dots, \mathbf{z}_s]$  be the orthonormal matrix consisting of the  $s$ -smallest eigenvectors. Then, from the fact that  $\mathbf{r}_j = -\mathbf{K}(\mathbf{v}_j - \mathbf{v})$  a projection process [Hous64] produces the improved iterate

$$\hat{\mathbf{v}}_j = \mathbf{v}_j + \mathbf{Z}(\mathbf{Z}^T\mathbf{K}\mathbf{Z})^{-1}\mathbf{Z}^T\mathbf{r}_j,$$

for which the corresponding residual  $\hat{\mathbf{r}}_j = \mathbf{b} - \mathbf{K}\hat{\mathbf{v}}_j$  has zero projection onto  $\mathbf{z}_i$ ,  $1 \leq i \leq s$ , i.e.,  $\mathbf{Z}^T\hat{\mathbf{r}}_j = 0$ . Note that  $\mathbf{Z}^T\mathbf{K}\mathbf{Z} = \text{diag}(\mu_1, \dots, \mu_s)$ .

The preprocessing cost for this algorithm is identical to that of the RS-CG scheme in §5. We are assuming for the time being that we also have the optimal parameters  $\mu = \mu_m$ ,  $\nu = \mu_{s+1}$  where  $s$  is a small integer (2 or 3) such that  $\mu_s < \mu_{s+1}$ , and  $\zeta_k \ll \kappa_k$  for  $k$  not too large, together with a reasonable approximation of the eigenpairs  $\mu_i$  and  $\mathbf{z}_i$   $1 \leq i \leq s$ . From earlier discussions it can be verified that a single iteration (6.1) consumes  $T(\text{BST}) \approx 20n + 2\psi$  time steps, again realizing a speedup of approximately  $n/2$  over the corresponding sequential scheme. In order to avoid the computation of inner products in each iteration, we adopt an unconventional stopping criterion. Once  $\zeta_i$ , the right-hand side of (6.5), drops below a given tolerance we consider  $\|\mathbf{r}_i''\|_2$  to be sufficiently damped and start the projection step. Therefore, once  $\mathbf{v}_{l+1}$  and  $\mathbf{r}_{l+1}$  are obtained, the improved iterate  $\mathbf{v}_{l+1}$  is computed by

$$(6.6) \quad \hat{\mathbf{v}}_{l+1} = \mathbf{v}_{l+1} + \sum_{i=1}^s \mu_i (\mathbf{z}_i^T \mathbf{r}_{l+1}) \mathbf{z}_i.$$

If each eigenvector is  $\mathbf{z}_i = (\mathbf{z}_1^{(i)T}, \dots, \mathbf{z}_{n/2}^{(i)T})^T$ , with  $\mathbf{z}_j^{(i)}$  residing in the local memory of processor  $j$ , (6.6) may be computed at the cost of approximately  $T'(\text{BST}) = (4.5 + \psi)ns$  time steps.

While it is reasonable to expect that the optimal parameters  $\mu$  and  $\nu$  ( $\mu_m \leq \mu < 1$ ,  $\mu_s \leq \nu < \mu_{s+1}$ ) are known, for example as a result of previously solving problem (5.1b) with a different right-hand side using the CG algorithm, it may not be reasonable to assume that  $\mathbf{Z}$  is known a priori. In this case the projection step (6.6) may be performed as follows.

Let  $k = l - s + 1$ , where  $l$  is determined as before, i.e. so that  $\tau_k(\beta)$  is large enough to assure that  $\|\mathbf{r}_k''\|_2$  is negligible compared to  $\|\mathbf{r}_k'\|_2$ . Now, from (6.2) and (6.4)

$$\mathbf{r}_k \approx P_k(\mathbf{K})\mathbf{Z}\mathbf{y},$$

where  $\mathbf{y}^T = (\eta_1, \eta_2, \dots, \eta_s)$ , or

$$\mathbf{r}_k \approx \mathbf{Z}\mathbf{w}_k$$

in which

$$\mathbf{w}_k^T = (\eta_1 P_k(\mu_1), \dots, \eta_s P_k(\mu_s)).$$

Consequently,

$$\mathbf{R}_l = [\mathbf{r}_{l-s+1}, \mathbf{r}_{l-s+2}, \dots, \mathbf{r}_l] \approx \mathbf{Z}[\mathbf{w}_{l-s+1}, \mathbf{w}_{l-s+2}, \dots, \mathbf{w}_l] = \mathbf{Z}\mathbf{W}_l.$$

Let

$$(6.7) \quad \mathbf{R}_l = \mathbf{Q}_l \mathbf{U}_l$$

be the orthogonal factorization of  $\mathbf{R}_l$  where  $\mathbf{Q}_l$  has orthonormal columns and  $\mathbf{U}_l$  is upper triangular of order  $s$ . As a result

$$\mathbf{Q}_l \approx \mathbf{Z}\mathbf{\Theta}_l$$

in which  $\mathbf{\Theta}_l$  is an orthogonal matrix of order  $s$ , and

$$(6.8) \quad \hat{\mathbf{v}}_{l+1} = \mathbf{v}_{l+1} + \mathbf{Q}_{l-s+1} (\mathbf{Q}_{l-s+1}^T \mathbf{K} \mathbf{Q}_{l-s+1})^{-1} \mathbf{Q}_{l-s+1}^T \mathbf{r}_{l+1}$$

has the desired property that  $\mathbf{Z}^T \hat{\mathbf{r}}_{l+1} \approx 0$ . Note that the eigenvalues of  $\mathbf{Q}_{l-s+1}^T \mathbf{K} \mathbf{Q}_{l-s+1}$  are good approximations of  $\mu_1, \dots, \mu_s$ .

The projection stage consists of the six steps shown below. The cost of each step can be readily estimated, based on the cost of each of the three fundamental tasks (i)–(iii) considered earlier.

1. The modified Gram–Schmidt factorization  $\mathbf{R}_l = \mathbf{Q}_l \mathbf{U}_l$ , where

$$\mathbf{R}_l = [\mathbf{r}_{l-s+1}^{(1)}, \dots, \mathbf{r}_l^{(1)}], \quad \mathbf{Q}_l = [\mathbf{q}_{l-s+1}, \dots, \mathbf{q}_l], \quad \mathbf{U}_l = \begin{bmatrix} \rho_{11} & \rho_{12} & \dots & \rho_{1s} \\ & \rho_{22} & \dots & \rho_{2s} \\ & & & \rho_{ss} \end{bmatrix},$$

is given by

for  $i = 1, 2, \dots, s$

$$\rho_{ii} = \|\mathbf{r}_{l-s+i}^{(i)}\|_2$$

$$\mathbf{q}_i = (1/\rho_{ii}) \mathbf{r}_{l-s+i}^{(i)}$$

for  $j = i + 1, \dots, s$

$$\rho_{ij} = \mathbf{q}_i^T \mathbf{r}_{l-s+j}^{(i)}$$

$$\mathbf{r}_{l-s+j}^{(i+1)} = \mathbf{r}_{l-s+j}^{(i)} - \rho_{ij} \mathbf{q}_i.$$

Since each residual  $\mathbf{r}_i \equiv \mathbf{r}_i^{(1)}$  is computed so that its first  $n$  elements are in processor 1, its second  $n$  elements in processor 2, and so on, it can be shown that  $\mathbf{Q}_l$  is determined in approximately

$$[(1.25 + 0.5\psi)s^2n + (4.25 + 0.5\psi)sn]$$

time steps with each  $\mathbf{q}_i$  replacing  $\mathbf{r}_i$  in the local memories of the  $n/2$  processors.

2. Obtain the  $s$  inner-products  $\gamma_i = \mathbf{q}_i^T \mathbf{r}_{l+1}$ ,  $l - s + 1 \leq i \leq l$ , and store them in processor 1, [cost  $\approx (2.5 + 0.5\psi)sn$  time steps].

3. Compute the upper triangular part of the symmetric matrix  $\mathbf{S} = \mathbf{Q}_l^T \mathbf{K} \mathbf{Q}_l$  and store it in processor 1, [cost  $\approx (1.25 + 0.25\psi)s^2n + (16 + 0.25\psi)sn$  time steps].

4. Solve the linear system  $\mathbf{S}\mathbf{d} = \mathbf{c}$ , where  $\mathbf{c}^T = (\gamma_{l-s+1}, \dots, \gamma_l)$ , sequentially in processor 1 in  $O(s^3)$  time steps.

5. Transmit each element  $\delta_i = \mathbf{e}_i^T \mathbf{d}$  to all processors, where  $\mathbf{e}_i$  is the vector with 1 as the  $i$ th component and zero elsewhere, and compute  $\mathbf{d}' = \mathbf{Q}_l \mathbf{d} = \sum_{i=l-s+1}^l \mathbf{q}_i \delta_i$ , [cost  $\approx (n/2 + s)\psi + 2sn$  time steps]. Note that, while processor  $j$  is transmitting  $\delta_1$  to processor



$j + 1$ , processor  $i, j - s + 1 \leq i \leq j - 1$ , is transmitting  $\delta_{j-i+1}$  to processor  $i + 1$ , ( $s \ll n/2$ ).

6. Finally,  $\hat{v}_{i+1} = v_{i+1} + d'$  is obtained in  $n$  time steps.

For small  $s$ , 2 or 3, the cost of this projection stage, following the BST iterations (6.1), is given by

$$T''(\text{BST}) \approx (2.5 + 0.75\psi)s^2n + (23 + 1.25\psi)sn$$

time steps.

**7. Some remarks.** The three methods discussed above have been implemented on the CDC CYBER 175 at the University of Illinois, for which the arithmetic precision is roughly 14 decimal digits. They were used to solve problem (2.1) with  $a(x, y) = b(x, y) = 1, c(x, y) = f(x, y) = 0.1, h = 1/33$  (i.e.,  $n = 32$ ), and with the Dirichlet boundary conditions  $u(x, y) = 0$ . The results are shown in Figs. 3-5, where the base 10 logarithm of the relative error in the solution appears as a function of the number of iterations for each algorithm. An initial iterate that does not favor any one algorithm was used for all three schemes. For the block-Stiefel algorithm, we have used  $\mu_2 < \nu \approx .044518 < \mu_3$ , i.e.  $s = 2$ , and  $\mu = \mu_m$  (the spectral radius of  $K \approx .999746$ ). Figure 5 shows the results of four experiments; terminating the iteration stage by performing a projection stage, (6.8), after  $j = 40, 50, 60$ , and 70 iterations, for which  $\tau_j(\beta) = 10^7, 10^9, 10^{11}$ , and  $10^{13}$ .

Performing the projection stage in BST after  $j = 70$  iterations, i.e. after  $\|r_j\|_2 / \|r_0\|_2 \approx 10^{-13}$ , we achieve a relative error in the solution of  $10^{-9}$ . The CCSI and CG schemes require 90 and 30 iterations, respectively, to achieve the same level of the relative error. On a sequential machine, the CG algorithm would certainly be the scheme that requires the least number of time steps for the iteration stage for this problem. The same holds true

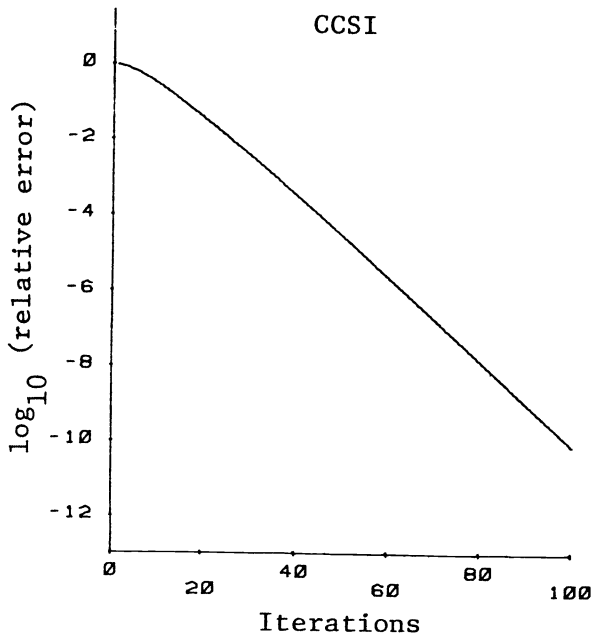


FIG. 3.

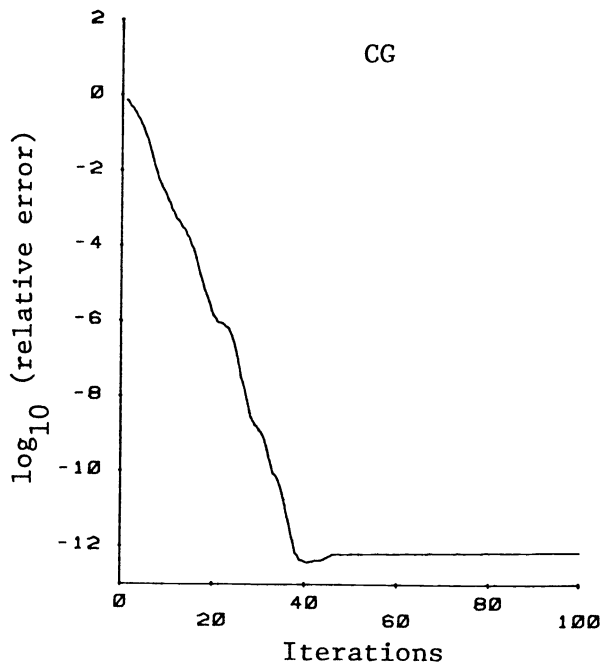


FIG. 4.

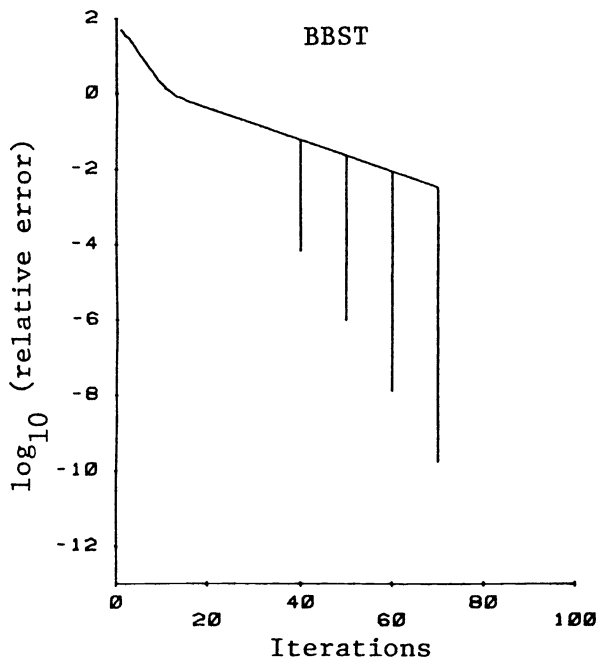


FIG. 5.

on our multiprocessor with  $\psi = 1$ . This, however, may not be the case as the value of  $\psi$  increases.

Now, in order to achieve a relative error of  $10^{-9}$  or less in the solution of the above model problem ( $n = 32$ ), the cost of each algorithm is given by:

- (i) cyclic Chebyshev semi-iterative (with optimal parameters)

$$T_{\text{total}}(\text{CCSI}) = 90 * T(\text{CCSI}),$$

- (ii) conjugate gradient (reduced system)

$$T_{\text{total}}(\text{CG}) = 30 * T(\text{CG}),$$

- (iii) block-Stiefel iterations (with optimal parameters and projection step via (6.8))

$$T''_{\text{total}}(\text{BST}) = 70 * T(\text{BST}) + T''(\text{BST}),$$

where for the sake of summary, for  $n = 32$  and  $s = 2$ , we have

$$T(\text{CCSI}) \approx 704 + 2\psi,$$

$$T(\text{CG}) \approx 832 + 64\psi,$$

$$T(\text{BST}) \approx 640 + 2\psi,$$

$$T''(\text{BST}) \approx 1792 + 176\psi.$$

Hence, for this problem, when  $s = 2$ , CCSI and BST become superior to RS-CG on linear array of processors only if  $\psi$  is greater than 22 and 14, respectively. Such large values of  $\psi$  are not unrealistic in certain designs, e.g. see [CoFi83].

**8. Further comments on CG and BST.** In this section we explore a little further the relative merits of the CG and BST methods. We shall assume that the (two) smaller eigenvalues are either known or else known to be isolated from the larger eigenvalues. In such a case, when good information is available concerning the spectrum, the BST method performs favorably. This should be expected. The CG method requires no such information but, for this advantage, a price is paid in computing two inner products at each step.

We have compared the performance of the conjugate gradient and the block-Stiefel algorithms for solving a system of linear equations, where the eigenvalues of the symmetric positive-definite matrix of coefficients,  $\mathbf{K}$ , have the following distribution,

$$0 < \mu_1 < \mu_2 < \mu_3 \leq \dots \leq \mu_m < 1.$$

In our experiments we have taken  $m = 512$ ,  $\mu_1 = .01$ , and  $\mu_2 = .02$ . The rest of the 510 eigenvalues are uniformly distributed in  $[\mu_3, \mu_m]$  with  $\mu_m = 0.9998$  and  $\mu_3$  taking the values 0.1998, 0.5998, and 0.8998. In these experiments the matrix of coefficients is diagonal.

TABLE 1

$\mu_3$	$\delta \approx$	No. of iterations (CG)	$\ r\ _2^2$
0.1998	$16 \times 10^{-4}$	26	$2 \times 10^{-13}$
0.5998	$8 \times 10^{-4}$	14	$4 \times 10^{-13}$
0.8998	$2 \times 10^{-4}$	9	$4 \times 10^{-3}$

$\delta \equiv$  eigenvalue spacing within the cluster  $[\mu_3, \mu_{512}]$ .

TABLE 2

$\mu_3$	No. of iterations (BST) & $\ r\ _2^2$			
	Projection step via (6.6)		Projection step vis (6.8)	
0.1998	16	$9 \times 10^{-13}$	22	$5 \times 10^{-12}$
0.5998	8	$2 \times 10^{-14}$	10	$7 \times 10^{-13}$
0.8998	5	$3 \times 10^{-17}$	6	$3 \times 10^{-16}$

(Since behavior of convergence depends only on the spectrum, this assumption is justified.) Table 1 shows the number of iterations required by the conjugate gradient method to ensure that the 2-norm of the residual is  $O(10^{-6})$ . Table 2 shows the corresponding results for the block-Stiefel scheme with the optimal parameters  $\nu = \mu_3$  and  $\mu = \mu_m$ , and both options of the projection step, i.e. via (6.6) when the eigenvectors corresponding to  $\mu_1$  and  $\mu_2$  are available, and via (6.8).

Were the eigenvalues of  $\mathbf{K}$  in (5.1b) distributed as in the above example (note that  $m = n^2/2 = 512$ , i.e.  $n = 32$ , and  $s = 2$ ), the BST scheme would have a definite advantage over the conjugate gradient method. If

$$T'_{\text{total}}(\text{BST}) = (\text{no. of iterations}) * T(\text{BST}) + T'(\text{BST}),$$

denotes the cost of the block-Stiefel iterations with the projection step performed via (6.6), where  $T'(\text{BST}) = 64(4.5 + \psi)$ , then the average of the ratio  $T'_{\text{total}}(\text{BST})/T_{\text{total}}(\text{CG})$  for the three choices of  $\nu = \mu_3$  is given by 0.45 for  $\psi = 1$ , and 0.30 for  $\psi = 10$ . While with the projection step (6.8), the average of the ratio  $T'_{\text{total}}(\text{BST})/T_{\text{total}}(\text{CG})$  is given by 0.70 and 0.50, for  $\psi = 1$  and 10, respectively.

We avoid the question of whether any matrix would have this distribution. We illustrate only what would occur if such information were available.

**9. Summary.** We have studied the iterative solution of elliptic difference equations on a simple multiprocessor, namely, a linearly connected array of processors. If the matrix is in line red-block form, we show how to perform a matrix vector multiply efficiently on such a multiprocessor. We consider three iterative methods, CCSI, CG, and BST. We determine the work required for each method in terms of interprocessor communication time,  $\psi$ , and the number of unknowns, and observe that for large values of  $\psi$ , but, nevertheless for values inherent in reasonable designs, the CG method may be at a disadvantage compared to the other two methods.

At the end of the paper, we briefly study an important feature of the BST method. The method exploits information about the spectrum, and we consider an example to illustrate this. If a few small eigenvalues are isolated, then this property improves the performance of the BST method as compared to the CG method.

**Acknowledgments.** We wish to thank Dr. J. Grcar and Dr. E. Kamgnia for their help in performing some of the numerical experiments. We would also like to thank the referees for their suggestions.

#### REFERENCES

- [AuCa78] R. AUBBUSON AND I. CATT, *Wafer-scale integration-fault-tolerant procedure*, IEEE J. Solid-State Circuits, SC-13 (1978), pp. 339-344.

- [BuGh77] B. BUZBEE, G. GOLUB AND J. HOWELL, *Vectorization for the CRAY-1 of some methods for solving elliptic difference equations*, High Speed Computer and Algorithm Organization, D. Kuck, D. Lawrie and A. Sameh, eds., Academic Press, New York, 1977, pp. 255–272.
- [CoFi83] ROBERT COOK, RAPHAEL FINKEL, DAVID DEWITT, LAWRENCE LANDWEBER AND THOMAS VIRGILIO, *The crystal nugget*, Computer Sciences Technical Report No. 499, Computer Sciences Dept., Univ. Wisconsin, Madison, April 1983.
- [CuVa59] E. CUTHILL AND RICHARD S. VARGA, *A method of normalized block iteration*, J. Assoc. Comput. Mach., 6 (1959), pp. 236–244.
- [Eric72] J. ERICKSEN, *Iterative and direct methods for solving the Poisson's equation and their adaptability to Illiac IV*, CAC document No. 60, Univ. Illinois at Urbana-Champaign, December 1972.
- [FuVa82] D. FUSSELL AND P. VARMAN, *Fault-tolerant wafer-scale architecture for VLSI*, Proc. 9th Annual Symposium on Computer Architecture, IEEE Computer Society Press, 1982, pp. 190–198.
- [GaSW82] D. GAJSKI, A. SAMEH AND J. WISNIEWSKI, *Iterative algorithms for tridiagonal matrices on a WSI-multiprocessor*, Proc. 1982 International Conference on Parallel Processing, Batchler, Meilander, and Potter, eds., IEEE Computer Society Press, 1982, pp. 82–89.
- [GePV78] A. GEORGE, W. POOLE AND R. VOIGT, *Analysis of dissection algorithms for vector computers*, Comp. Math. Appl., 4 (1978), pp. 287–304.
- [GoVa61] G. GOLUB AND R. VARGA, *Chebyshev semi-iterative methods, successive over-relaxation methods, and second order Richardson iterative methods, Parts I and II*, Numer. Math., 3 (1981), pp. 147–168.
- [HaLY80] L. HAGEMAN, F. LUK AND D. YOUNG, *On the equivalence of certain iterative acceleration methods*, SIAM J. Numer. Anal., 17 (1980), pp. 852–873.
- [HaYo81] LOUIS A. HAGEMAN AND DAVID M. YOUNG, *Applied Iterative Methods*, Academic Press, New York, 1981.
- [Hous64] A. HOUSEHOLDER, *The Theory of Matrices in Numerical Analysis*, Blaisdell, Waltham, MA, 1964.
- [LeLe83] F. LEIGHTON AND C. LEISERSON, *Wafer-scale integration of systolic arrays*, TM-236, Laboratory for Computer Science, Massachusetts Institute of Technology, Cambridge, 1983.
- [Miur71] K. MUIRA, *The block-iterative method for Illiac IV*, CAC document No. 41, Univ. Illinois at Urbana-Champaign, July 1971.
- [Ruti59] H. RUTISHAUSER, *Refined iterative methods for computation of the solution and the eigenvalues of self-adjoint boundary value problems*, Theory of Gradient Methods, M. Engli, Th. Ginsburg, H. Rutishauser and E. Stiefel, eds., Springer-Verlag, Heidelberg, 1959.
- [Same81] A. SAMEH, *Parallel algorithms in numerical linear algebra*, CREST Conference, Bergamo, Italy, July 1981.
- [Schu81] M. SCHULTZ, ed., *Elliptic Problem Solvers*, Academic Press, New York, 1981.
- [Stie58] E. STIEFEL, *Kernel polynomials in linear algebra and their numerical applications*, Nat. Bur. Standards, Appl. Math. Series 49, 1958, pp. 1–22.
- [Varg62] R. VARGA, *Matrix Iterative Analysis*, Prentice-Hall, Englewood Cliffs, NJ, 1962.
- [Youn71] D. YOUNG, *Iterative Solution of Large Linear Systems*, Academic Press, New York, 1971.